

A Sketch-Based User Interface for Reconstructing Architectural Drawings

Sangwon Lee¹, David Feng², Cindy Grimm³ and Bruce Gooch⁴

¹Northwestern University, Evanston, IL USA
s-lee@cs.northwestern.edu

²University of North Carolina, Chapel Hill, NC USA
d.feng1@gmail.com

³Washington University in St. Louis, MO USA
cmg@wustl.edu

⁴University of Victoria, BC, CANADA
brucegooch@gmail.com

Abstract

We present a framework for interactive sketching that allows users to create three-dimensional (3D) architectural models quickly and easily from a source drawing. The sketching process has four steps. (1) The user calibrates a viewing camera by specifying the origin and vanishing points of the drawing. (2) The user outlines surface polygons in the drawing. (3) A 3D reconstruction algorithm uses perceptual constraints to determine the closest visual fit for the polygon. (4) The user can then adjust aesthetic controls to produce several stylistic effects in the scene: a smooth transition between day and night rendering, a horizon knockout effect and entourage figures. The major advantage of our approach lies in the combination of perception-based techniques, which allow us to minimize unnecessary interactions, and a hinging-angle scheme, which shows significant improvement in numerical stability over previous optimization-based 3D reconstruction algorithms. We also demonstrate how our reconstruction algorithm can be extended to work with perspective images, a feature unavailable in previous approaches.

Keywords: visual perception, single-view reconstruction

CR Categories: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Visible line/surface algorithms; I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Modeling packages

ACM CCS: I.3.5 [Computational Geometry and Object Modelling]: Modelling packages

1. Introduction

Constructing three-dimensional (3D) models that match a given two dimensional (2D) drawing enables navigation of the 2D line-drawing world, viewing of pictures and paintings from different angles, and adjustment of architectural design based on 3D model walkthrough. Manual construction of such models is a tedious, involving CAD software training; CAD has a complex interface that requires the user to impose 3D information on the model in ways that include manual matching between the drawing and predefined 3D primitive shapes, specification of correspondences between multiple

pictures, and use of various guiding lines to trace landing positions of strokes.

A similar problem exists when designing a 3D model from scratch. The user interface of most CAD systems focuses on structural processes of 3D model construction. However, drawing, a skill architects spend 7–10 years mastering, concerns itself foremost with a building's visual impact. Changing a few lines in hand drawing requires little work and can produce dramatic changes in building structure; creating an analogous change in a CAD model may require extensive, time-consuming adjustments.

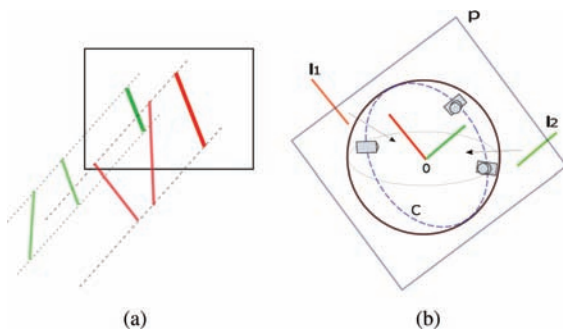


Figure 1: (a) A 2D line can be a projection of infinitely many 3D lines. (b) Two arbitrary, non-parallel lines l_1 and l_2 in 3D space are translated to the center of a sphere. The plane P containing these two lines intersects the sphere along the circle C . The only cameras that lie on this circle and aim the center of the sphere represent the possible camera directions that produces parallel lines on 2D camera plane from l_1 and l_2 . Considering that cameras can have arbitrary position on the sphere, this is a significantly limited range.

In order to address this problem, we propose an intuitive sketching interface that retains the familiarity of hand drawing in conjunction with a perceptually-based 3D reconstruction algorithm for inferring a 3D depth information. Given a 2D line drawing, we construct a cost function using several known visual constraints based on principles of human visual perception that, upon optimization, results in a 3D model. In this way, we achieve an integrated interface for modelling and reconstruction, with minimal user annotation for assigning 3D positions to the model. While our results focus on reconstruction from existing drawings, the same procedure can be applied to generate 3D models from scratch. Note that our primary goal is creation of a smart and intuitive user interface for modelling and reconstruction, rather than generating 3D models from rough, conceptual design sketches.

The cost function of the optimization algorithm is intimately related to determining depths of geometric primitives forming the 3D model. This is an under constrained problem: a line in 2D has infinitely many possible projections in 3D [Figure 1(a)]. The architectural drawing domain, however, affords us several conventions we can leverage to simplify the problem. For example, architectural drawings generally contain many sets of parallel lines, and lines that are parallel in 2D architectural drawings are likely also to be parallel in 3D. While there are cases when this assumption does not hold, such accidental viewpoints only occur under very restricted camera positions and avoided by architects [Figure 1(b)].

Previous methods have used the parallel line assumption to find the optimal positions of all *vertices* in a 3D model from a graph of lines in a 2D image [MKL05]. Such approaches is that they require minimization of a multi-dimensional non-

linear function where the dimension of the problem space is determined by the number of vertices in the scene. This is problematic because algorithms for optimizing such functions tend to get stuck in local minima. Furthermore, the runtime can become impractical as the number of vertices increases. Observing that architectural structures are highly planar, we instead use *polygons* as our fundamental optimization primitive, resulting in a more robust, single parameter per-polygon optimization that scales well with complex models and that, due to its speed and compatibility with hand-drawn sketches, supports the drawing stage of architectural design well.

2. Related work

Our user interface and 3D reconstruction algorithm are comparable with previous work in two areas: 3D model recovery from a single image and interactive 3D sketching systems.

Shesh et al. [SC04] and Lipson et al. [MKL05] both use optimization-based techniques for 3D model construction from an orthographic projection. These methods have two restrictions: they can only handle drawings that show and annotate all hidden lines, and they do not support perspective drawings. Our tool removes these limitations by allowing reconstruction from perspective images without hidden lines.

Hoiem et al. [HEM] suggest a completely automatic method for popping up a scene by assuming that all objects in the scene are vertical to a ground plane. Although their method shows that a simple structure can be effective for viewing a picture in 3D, the approximate nature of the corresponding reconstruction process prohibits editing or compositing and restricts final models to vertical billboards. We impose no such restrictions, and our interface gives the user the option to edit the resulting geometry.

In Facade [DTM96], the user assembles geometry in a CAD-style user interface by choosing among a pre-defined set of primitives. The advantage of this approach is the robust geometric accuracy achieved by manual assignment of 3D position. However, it restricts the number and form of allowable shapes to those in the primitive set. Our interface is not restricted to a primitive set and can generate any realizable polygonal shape.

Because any user-drawn stroke can be interpreted in a multiple ways, a sketching interface must acquire additional information to correctly infer the 3D model's desired geometry.

The Chateau [IH01] system is a polygonal model drawing system that lets the user quickly turn sketched lines into polygons and 3D shapes. Chateau addresses reconstruction ambiguity by building several candidate reconstructions and letting the user select the intended one.

Zelevnik et al. [ZHH96] first introduced Sketch, a system for rapidly creating and editing a 3D scene using a purely

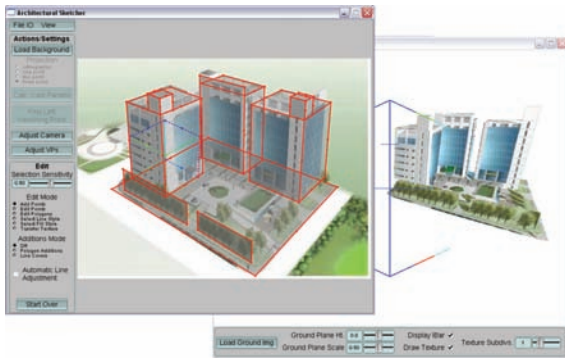


Figure 2: A snapshot of the complete user interface windows, consisting of a sketching window (top) and a 3D navigation window (bottom).

gestural interface. The system supports a variety of primitives (including any shape that can be extruded vertically from a planar curve, such as cones, spheres, etc.) along with basic solid modelling operators. Sketch could be used to create most of the models shown here. However, extrusion-based modelling is conceptually very different from drawing. By retaining a simple drawing interface, our system is better suited to complement designers' and architects' training.

Sketchup [Ske] is a commercial software package that addresses the drawbacks of many CAD tools. Although the reconstruction process is completely manual, and requires many user-specified lines for tracking 3D positions, its resemblance to traditional drafting procedure makes 3D modelling far easier and faster for non-expert users. Shesh et al. [SC05] shares a goal similar to ours; an intuitive interface for progressively constructing 3D model. Although their modelling procedure employs an optimization method for intelligent reconstruction, perceptual constraint selection is manual.

3. User Interface

Our user interface consists of two windows: a sketching window and a model-view window (Figure 2). The sketching window provides the user with a simple, intuitive means of drawing lines and shapes over an existing drawing or photograph. The model-view window supports arbitrary model viewpoints including textured and flat-shaded. Before polygons can be drawn, the user calibrates the scene camera by annotating several on top of the 2D drawing. In orthographic projection, the user draws three lines which correspond to projection of the three unit-length principal axes. For perspective images, the user refines camera parameters by sketching lines which converge on the vanishing points of the scene (Figure 3(a)), or several edges that represents a portion of the unit cube in 3D space projected onto the camera plane (Figure 3(b)). We discuss our camera calibration techniques in section 4.

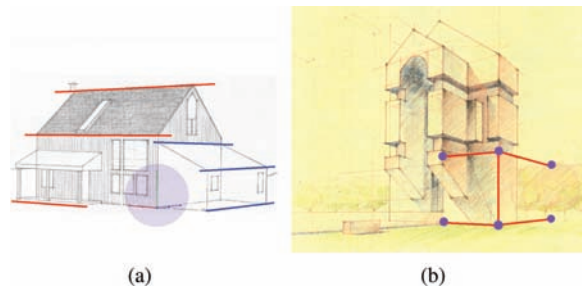


Figure 3: Two suggested methods for camera calibration. a) Specifying principal axes followed by vanishing lines. b) Drawing I shape corresponding to a unit cube in 3D space.

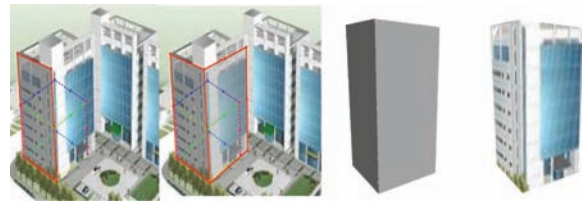


Figure 4: From left to right: a) Sketching the first polygon relative to the coordinate system. b) Sketching a new polygon adjacent to an existing polygon. c) The resulting 3D geometry. d) The texture mapped geometry.

After camera calibration, the user starts at the origin (Figure 4), indicating a new polygon by tracing its silhouette; the system finds the most likely 3D polygonal reconstruction and adds the new 3D polygon to the model. Our algorithm requires that each new polygon either share an edge with an existing polygon (e.g. adjoining walls) or be connected to the surface of an existing polygon. This process parallels hand drawing in that the basic shape of the structure must be outlined before additional details can be added.

Because line drawings by user may not perfectly match the desired 3D model, our tool provides the option of rectifying the polygon edges so that they align to vanishing points (or to perfect vertical lines, in case y axis is up direction). Rectification of a 2D polygon is achieved by iterating through the edges, forcing each to snap to nearby *important lines*. The rectified polygon is used to generate 3D geometry, and the original 2D polygon is used to create the texture map.

4. Specifying the Camera

When finding the 3D model that best represents the given 2D drawing, 3D vertex positions can be represented as an x and y coordinate lying in the camera plane, and a z coordinate denoting depth from the camera plane. The depth depends on camera parameters that define the shape of the projection frustum. The projection frustum determines the vector from the eye position to the 2D point on the camera plane, on

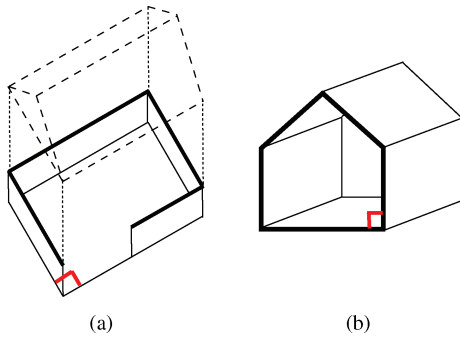


Figure 5: Example of orthographic drawings: planview (a) and sectionview (b). Note that the target plane is unskewed (shown with red angle) and the perpendicular side planes are drawn diagonally. In standard orthographic projection with 1:1 aspect ratio, side planes should be invisible in the drawing. If it were to show side planes, the angle in red should be larger (in (a)), or smaller (in (b)). We calibrate these scenes by allowing to adjust the aspect ratio of the camera.

which the 3D position in world coordinate should lie. We find camera parameters through user annotation and existing camera calibration techniques. Once the scene camera of the scene is found, we can represent a 3D model as a collection of depths on vertex points.

In perspective projection, we can infer camera parameters from various cues like point correspondences, corner orthogonality or vanishing points locations. These cues indicate the relationship between 3D and 2D coordinates, suggesting the shape of the frustum and location of the camera in 3D space. In orthographic projection, however, it is less obvious that we need a calibrated camera since the frustum's shape remains perpendicular to the camera plane and depths of vertices do not change regardless of the external camera parameters, as long as the internal parameter has 1:1 aspect ratio. Therefore previous methods for 3D model reconstruction for orthographic projections do not include a camera calibration step [IH01] [LS96] [SC04]. However, architectural drawings frequently include different variations of orthographic projection which are devised from the architect's point of view [Chi95]. For example, a house plan may be drawn from a camera view projecting perpendicular to a target plan and have the side walls raised as diagonal lines in 2D (Figure 5). In ordinary orthographic projections with 1:1 aspect ratio, when the plan is parallel to the camera plane, sides of walls may not be shown with the frontal wall drawn unskewed. However, this technique is commonly used among architects because it allows viewing of the overall 3D structure at the same time that plans are drawn to scale without foreshortening. Another drawing technique uses three principal axes foreshortened at an arbitrary rate (trimetric projections). In order to represent these drawings correctly, we have to find the unique internal as well as external camera parameters through calibration under orthographic as well as perspective projection.

We include two methods for camera calibration: the principal axis method and the unit cube method. These two methods compliment each other in their ability to handle different scene types and projection modes.

The basis of principal axis camera calibration is the Simplex solver [CSB*05], a general-purpose gradient descent solver. Given default camera settings, the Simplex solver attempts to iteratively adjust those settings until they match those of the camera represented in the input image. This mainly consists of two stages of user interaction. First, a user draws three calibration lines that correspond to the camera's unit-length principal axes as represented in the input image [Figure 3(a)]. To avoid shape ambiguity, we require users to draw these axes on top of a convex corner. Second, the solver uses endpoints of calibration lines as point constraints for the Simplex solver's error function, which is minimized by adjusting camera parameters. Let P_i be the user-drawn origin and axis points, and q_i their desired screen-space position. The error function is:

$$[uvw]_i^T = CP_i \quad (1)$$

$$E = \sum_i ||q_i - (u_i/w_i, v_i/w_i)||^2 \quad (2)$$

Having approximately located the principal axes in image space, we can further calibrate the camera by taking into account the location of the image's vanishing points. Given lines drawn from the left and right vanishing points, we can use the Simplex solver to enforce line parallelism constraints [Figure 3(a)]. Once these constraints have been applied, the camera is generally well-defined. Specification of vanishing points is obviously unnecessary for orthographic images.

The second method of camera calibration uses Direct Linear Transformation (DLT) [HZ03]. The user draws an I shape which matches several edges of the unit cube in 3D space [Figure 3(b)]. This cube provides enough 3D–2D point correspondences to recover the camera projection matrix. In our experiment, use of cube edges' mid-points and end-points improved the solver's numerical stability. We use Singular Value Decomposition to remove point correspondence redundancy and compute the optimal camera matrix. We apply QR decomposition to the camera matrix to decompose it into a product of the internal and external camera matrices. The DLT method is more deterministic than the Simplex-based method and generally performs more accurately when calibrating 2D perspective images. Interestingly, this unit-length matching technique gives the user freedom to control scale factors in x, y and z directions in 3D space.

5. Polygon Reconstruction

The reconstruction process consists of annotating the source image with lines. These lines form a planar graph of 2D intersection points. To define polygonal regions, we need to

detect the smallest of any cycles in the graph that result from drawing a line. Starting from the newest drawn edge in the graph, we traverse from one endpoint in both a clockwise and counter-clockwise direction, stopping when the other endpoint of the line is reached. If the other endpoint isn't visited during traversal, then no new polygon exists. If the other endpoint is visited, the path indicates two cycles: the desired minimal cycle and the cycle enclosing the whole planar graph. A simple polygon inside-outside test picks the desired cycle.

A new polygon may be connected to an existing polygon by either a shared *edge* or a shared *surface*. Polygons that have more than one neighbour already have sufficient information to completely determine the plane of the new polygon. If the polygon has only one neighbour, then it has only one degree of freedom: rotation along the connecting edge. We optimize this angle by *minimizing* a sum of weighted functions which are based on three perceptual constraints: axis alignment, parallelism and symmetry. These constraints are discussed individually below.

We use Brent's [Bre73] algorithm to find the optimal angle at which the two polygons should be hinged. It is a robust technique for one-dimensional functions and does not require derivative calculations. The domain of the angle $[0,180)$ is discretized to integers, and we compare the local minima in subdomains to get global minimum. For each subdomain, approximately three iterations were sufficient to compute the global minimum value. The cycle's cost function is:

$$f_{total}(\theta) = f_{axis}(\theta) + f_{symmetry}(\theta) + f_{parallel}(\theta) \quad (3)$$

The primary advantage of this approach is that the cost function optimizes over single variable, an angle from 0 to 180 degrees, and so is amenable to brute force search followed by a refining optimization step. This avoids getting stuck in local minima assuming that the cost function remains smooth enough within the given integer sampling domain. This is an important achievement because parameterization using depths are usually unrestricted, which makes finding the correct global minimum a difficult task.

If the newly drawn polygon does not share an edge with any existing polygons in the 2D edge-vertex graph, we attempt to project that polygon onto background surfaces. We first identify this situation by noticing if the new polygon shares no edges or vertices with the current 2D vertex-edge graph but does lie in the interior of an existing projected polygon. If this is the case, the most recently drawn line is projected onto the existing polygon and uses this line as the hinging edge (see Figure 6).

5.1. Axis alignment

For any image, the only edges which supply immediately useful 3D information are those which are parallel to principal

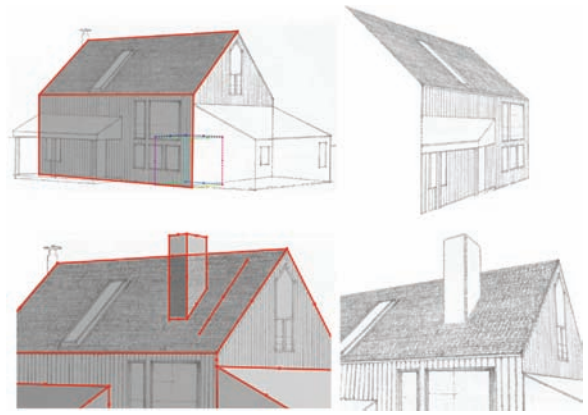


Figure 6: Examples showing hinging angle scheme. The new polygon is attached to an existing polygon by an edge (top). The edges of new polygons can lie inside existing polygons. Note that the chimney does not exist in the original drawing. The detached stroke on roof extracts color information and defines the style of the chimney's sketching lines (bottom).

axes, as such lines are likely to be parallel in 3D as well. The range of camera positions under which two non-parallel 3D lines are parallel in the projected 2D plane is limited; a slight change of camera view direction reveals this accidental view and is not usually considered to be the best view for architectural objects. We enforce lines parallel to principal axes to be parallel also in 3D by assigning a θ that minimizes the function:

$$f_{axis}(\theta) = \sum_{e \in cycle} w_e \|I_e^\theta \times P_{x|y|z}\| \quad (4)$$

where $P_{x|y|z}$ is principal axis, that is most parallel to the edge being examined. The magnitude of the cross product between current edge and $P_{x|y|z}$ favors a θ that projects 2D axis-aligned lines to 3D lines parallel to the principal axis. For orthographic images, we calculate the weight of each edge w_e as:

$$w_e = \max_{i \in \{x,y,z\}} [G_A(\text{angle}(I_e, P_i))]. \quad (5)$$

Equation 5 calculates the degree of being parallel between current edge and the principal axis most parallel to the edge. A Gaussian distribution function G_A returns the normalized value of the angle difference; 1 means alignment while 0 means perpendicularity. This method provides far smoother normalization than thresholding, where an angle slightly larger than the threshold is assigned a full weight and an angle slightly smaller than the threshold has zero weight. In perspective projection, an edge's parallelism to a principal axis is calculated as the distance from the edge to each vanishing point:

$$w_e = \max_{i \in \{x,y,z\}} [G_L(\text{dist}(I_e, V_i))]. \quad (6)$$

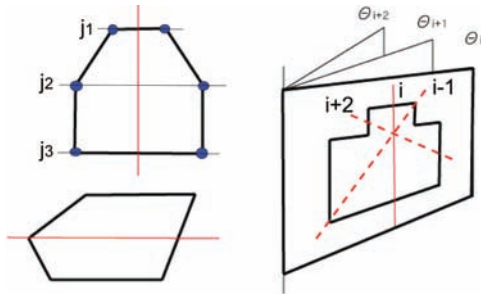


Figure 7: Left: Detecting symmetry with orthographic drawings showing vertex pairs j with given best mirror axis. Right: Symmetry in perspective drawing. We examine all possible mirror axis i with different hinging angle θ .

Here, the unit of deviation in Gaussian G_L is length instead of angle. In one and two point perspective drawings, the calculation of w_e uses both Equations 5 and 6 since orthographic and perspective style vanishing lines coexist.

5.2. Symmetry

The human visual system also has a strong tendency to detect symmetry within polygons. Our system detects symmetry by modifying a method suggested by Lipson and Shpitlani [LS96]. We parameterize the problem using hinging angle and extend it to perspective projection. In essence, we first determine the best mirror axis for each possible hinging angle, and then select the hinging angle corresponding to the best mirror axis from the set of best mirror axes.

Computing the mirror axis is far easier in orthographic projection than perspective projection; for all axes that divide the polygon in half, we examine the deviations of angles and distances between the axis and vertices along the axis. In perspective images, finding candidate axes is difficult since the mid-points of 2D edges do not correspond exactly to mid-points in 3D due to perspective distortion. We remove this distortion by projecting edges to planes with integer hinging angles θ . Then we find the best mirror axis for each integer hinging angle (see Equations 7–9). For angle θ , we pick the candidate mirror axis i that divides the polygon in half, and examine pairs of vertices j with respect to the chosen mirror axis as shown in Figure 7.

$$w_s = G_{0.1} \left[\min_{\theta=0..180} \left[\min_{i=1..n} (\sigma(\text{perp}_i[j]) + \sigma(\text{sym}_i[j])) \right] \right] \quad (7)$$

where

$$\text{perp}_i[j] = \|I_{(v_1, v_2)} \cdot I_{(v_3, v_4)}\| \quad (8)$$

$$\text{sym}_i[j] = \|\text{dist}(I_{(v_1, v_2)}, v_3) - \text{dist}(I_{(v_1, v_2)}, v_4)\| \quad (9)$$

$$\begin{aligned} v_1 &= v\left(\frac{i}{2}\right), & v_2 &= v\left(\frac{i+n}{2}\right) \\ v_3 &= v\left(\frac{i+j}{2}\right), & v_4 &= v\left(\frac{i-j}{2}\right) \end{aligned}$$

For a candidate axis $I_{(v_1, v_2)}$, vertices in the mirror direction (v_3, v_4) should have a similar distance to the candidate axis ($\text{sym}_i[j]$) and the edge connecting these edges $I_{(v_3, v_4)}$ should also be perpendicular to the candidate axis ($\text{perp}_i[j]$). Optimizing the following cost function generates the actual 3D reconstruction:

$$f_{\text{symmetry}}(\theta) = w_s \sum_{i=k, j=1..n} \|I_{(v_1, v_2)}^\theta \cdot I_{(v_3, v_4)}^\theta\| \quad (10)$$

where k is the optimal i found from Equations 7–9. The dot product forces a 90 degree angle between the mirror axis and edges with vertex pairs along the mirror axis. Note that the weight w_s is common to all edges within the current cycle.

5.3. Parallelism

A drawing can contain sets of parallel lines that are not parallel to a principal axis. For each edge in the current cycle, we search for edges in the partially constructed 3D model that are likely to be parallel. We find these edges and apply parallelism constraint by minimizing the function:

$$f_{\text{parallel}}(\theta) = \sum_{e \in \text{cycle}} w_e \|I_e^\theta \times I_e^{\text{parallel}}\| \quad (11)$$

where I_e^{parallel} is a direction in 3D to which edge e is likely to be parallel. This is the direction of the 3D edge that returns the best weight w_e . In orthographic projection, w_e for each edge is:

$$w_e = \max_{i \in \{3D \text{ edges}\}} [G_A(\text{angle}(I_e, I_i))]. \quad (12)$$

In perspective projection, parallel lines are parallel in 3D if they merge at a single vanishing point in 2D. We compute minimal distance from the edge being tested to the vanishing points for w_e where V_i is a vanishing point of parallel lines in the image.

$$w_e = \max_{i \in \{\text{vanishing points}\}} [G_L(\text{dist}(I_e, V_i))]. \quad (13)$$

6. Texture Mapping

The texture mapping process consists of using 2D image vertices as texture mapping coordinates and adjusting the subsequent texture map to account for perspective foreshortening in the 2D image. We correct perspective distortion by introducing polygon subdivision (Figure 8). The subdivision algorithm recursively divides n -sided polygons into n distinct polygons by introducing edge mid-points and a centroid point as vertices (i.e., Catmull–Clark subdivision). We subdivide polygons in realtime before rendering each frame to simplify mesh intersection. For modern machines, using fewer

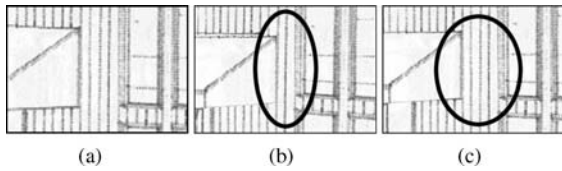


Figure 8: Adjusting for perspective distortion. Note the number of stripes between the porch roof and the window.

than four subdivision iterations does not seem to noticeably reduce frame rates.

7. Results

7.1. Interactive drawing

In Figures 10 and 11, we analyse the polygon reconstruction process by comparing cost function graphs. Recall that the minimum cost hinging angle is used for 3D model reconstruction. In Figure 10(a), the axis alignment cost function is dominant, since all edges in polygon A are parallel to principal axes. Figure 10(b) shows a case where symmetry dominates because edges parallel to principal axes have little effect on hinge angle cost. In Figure 10(c), parallelism is the most important cost since the new polygon was drawn unaligned intentionally such that non-axis aligned edges are parallel to edges in polygon B and the effect of symmetry cost function is minimal.

Figure 11 shows cost function graphs in orthographic projection. In Figure 11(a), symmetry and parallelism are suppressed since newly added lines are not parallel to existing lines. In Figure 11(b), axis alignment and parallelism stay constant because newly added lines remain aligned regardless of the hinging angle in orthographic projection. These three constraints have similar minimum hinging angle as shown in Figure 11(c).

We show 2D drawings and photographs with their corresponding 3D model reconstructions in Table 1. The fifth example, a line drawing of a barn, demonstrates the system's ability to quickly and accurately generate a 3D model, and easily create additions to its structure. Notice the fine details of the windows and skylight, and how the texture-painting mechanism has been used to hide hidden lines in the outputted 3D model. The system can also handle architectural photographs, as demonstrated by images in Table 2. These images are considerably more complex than the barn drawing.

Architects and designers from seven architectural schools and firms evaluated our system's user interface for usefulness in their early design processes. Most evaluators note that our system does not suffer from conceptual discrepancy and stifled creativity that commonly occur when shifting from 2D sketches to 3D models using traditional software. The evaluators suggested several possible applications of our tool:

Table 1: Notation used in the equations.

$cycle$	set of edges in the new polygon
n	number of vertices in the $cycle$
P_x	principal axis x
V_x	vanishing point of lines parallel to axis x
I_e	2D line of edge e
I_e^θ	3D line of I_e projected to plane with hinge angle θ
$I_{(v_1, v_2)}$	2D line of edge connecting vertex v_1 and v_2
$I_{(v_1, v_2)}^\theta$	3D line of $I_{(v_1, v_2)}$ projected to plane with hinge angle θ
v_n	a n_{th} vertex point
$v_{n+1/2}$	mid-point of vertex v_n and v_{n+1}
$dist(I, v)$	distance from 2D line I to vertex v
$angle(I_1, I_2)$	angle between two 2D lines I_1 and I_2
$G_A(x), G_L(x)$	a Gaussian distribution function whose mean value is zero and deviation depends on whether the unit of x is angle (G_A) or length (G_L)
$\sigma(x_i)$	standard deviation of series x_i

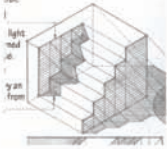
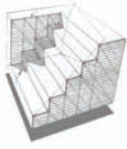
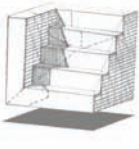
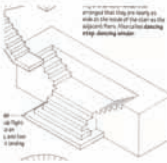
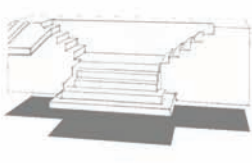
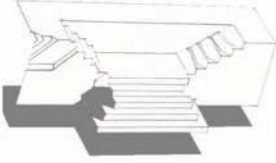
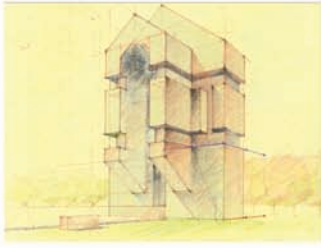



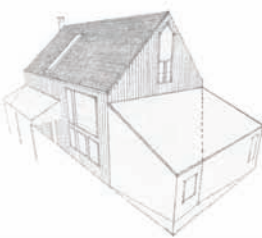


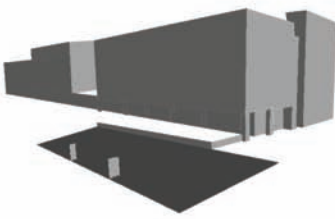




measuring total site area, judging how well a 3D model fits in a location and measuring object volume from a single image. The mathematics required to perform these tasks are already well-understood; through our user interface we provide simple, easy access to these functions. Several evaluators expressed concerns about the ability to construct hidden polygons, but found this less problematic once they understood our tool's construction mechanism.

7.2. Aesthetic improvements

A key component of architectural design is being able to see the structure in its destined environment. Doing this can improve a viewer's sense of scale by incorporating human figures and trees into the drawing and viewing the structure under various lighting conditions. In our system, users can place both figures and trees in their image with a simple point-and-click interface and can change time of day by adjusting a slider (Figure 9). As the slider moves from day to night, the sky changes from a bright blue to a dark blue sunset while the hue of the structure changes accordingly. Additionally, polygons that represent windows can be rendered with a soft yellow glow upon selection. The user may also choose to load a ground texture image if the default texture is not satisfactory; both height and scale of ground texture are easily adjustable. The video accompanying this paper demonstrates the usage of these features.

In order to give rendered images a stronger visual effect, our tool's default setting is to display the 3D environment with an effect commonly referred to as 'knockout,' or giving the structure a soft glow along the horizon so as to make it

Table 2: Left: original images. Right: resulting 3D model. Photo rendering in fifth row courtesy of Davis Brody Bond, LLP.

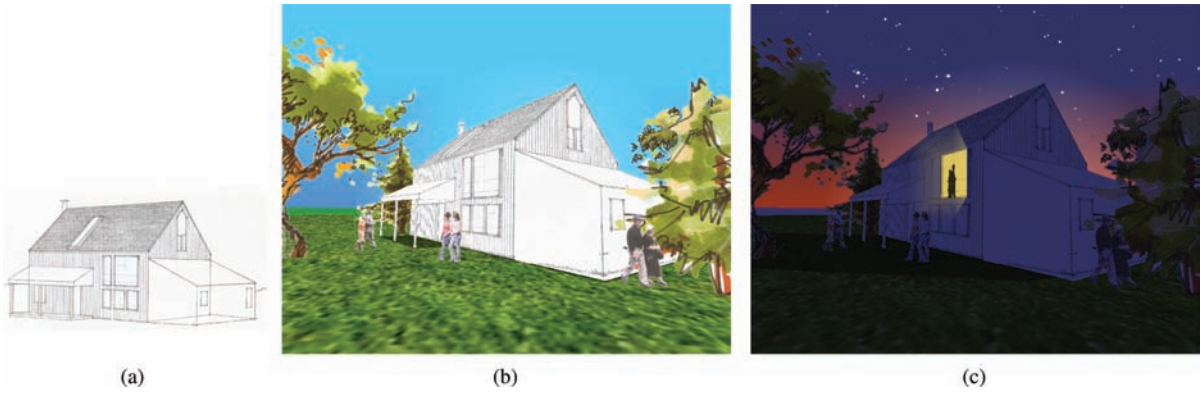


Figure 9: Day and night views of a line drawing by Francis Ching. Left: the original drawing. Center: a day time view including trees, figures and ground texturing. Right: a night time view with a starry sky, structural knockout and glowing windows.

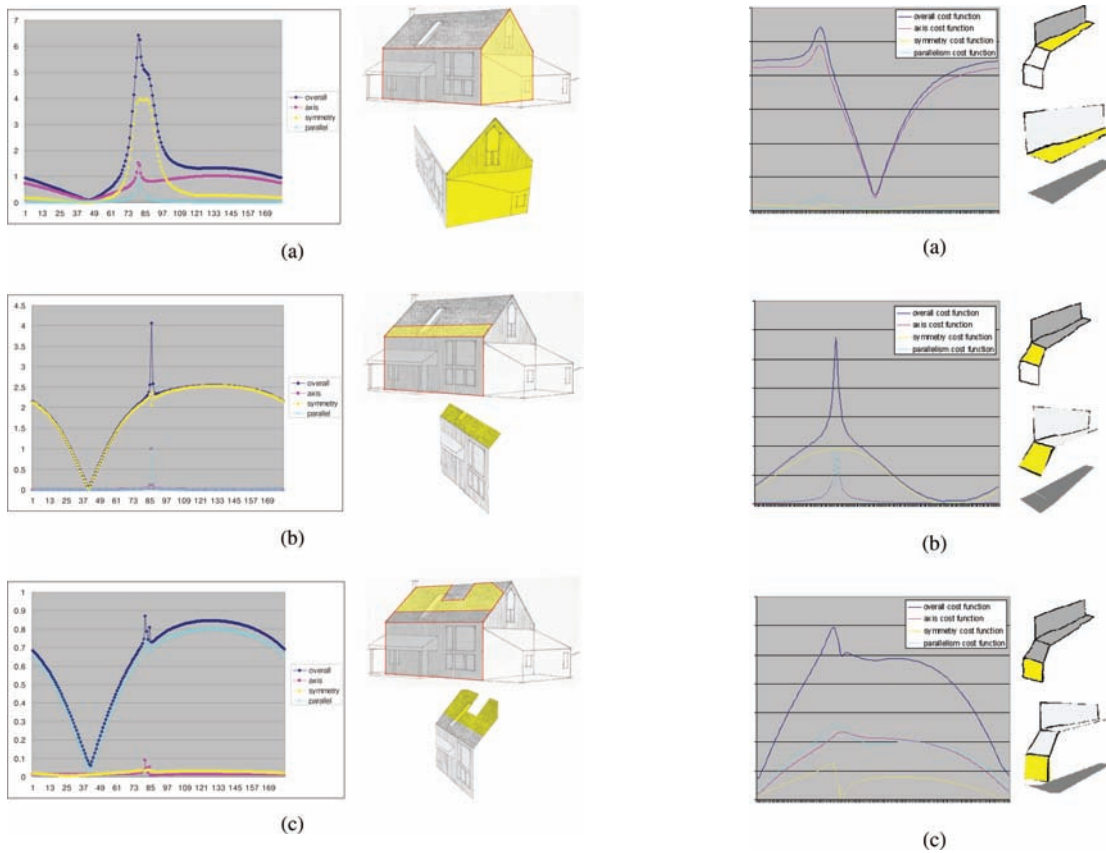


Figure 10: Results showing change of the cost function with respect to hinging angle under perspective projection. In (a), all constraints agree on a single hinging angle. Note that the new polygon in (b) has symmetry condition as its dominant constraint, while in (c), the prevailing constraint is parallelism. The polygon in (c) was drawn with the intention that the symmetry constraint would be suppressed.

Figure 11: Results showing change of the cost function under respect to the hinging angle with orthographic projection. (a) Axis alignment is dominant. (b) Symmetry determines the hinging angle. (c) All constraints agree on a single angle. In each figure, right top image is original 2D drawing and right bottom one is the reconstructed 3D model.

stand out from the background. We implement this by attaching several alpha-blended polygons to the silhouette of the structure in screen space.

8. Discussion and Future Work

We provide a complete drawing system in which users can interactively reconstruct architectural images in 3D space. The system uses perceptual constraints to automatically suggest plausible 3D placement of polygons. Unlike previous optimization-based approaches, we significantly improve speed and robustness using our polygon-based hinging-angle scheme. Also, our system requires little interaction beyond the drawing process itself. We had expert 3D modellers construct a model from one of our test drawings in Maya to gauge effectiveness of our solution. While it took modellers on the order of an hour to complete the task in Maya, model creation time with our system was 5–10 minutes including the sketching itself. In practice, users did not encounter any problem drawing on an existing image. The camera calibration step, however, was relatively more difficult and occasionally required repeated efforts especially when the Simplex solver failed to find the optimal global camera parameters.

One disadvantage of our system is that it only reconstructs visible portions of surfaces in a source image. This is a consequence of our initial assumption regarding input: a single view containing no hidden lines. However, because our system supports sketching from any view point, users can remedy incomplete polygons in the 3D model.

Reconstruction quality depends on accuracy of the 2D line drawing. For example, when a hinging axis line is significantly misaligned in the original 2D drawing, the resulting 3D placement might have the polygon facing in the wrong direction or accumulate error in subsequent optimizations. We minimize this problem by snapping the hinging axis to principal axes whenever appropriate. The degree of insensitivity to the inaccurate drawing generally depends on two factors: the parameters of Gaussian functions which control alignment degree, and overall alignment of the newly drawn polygon. If there is any single constraint applicable, the angle optimization procedure will find the right hinging angle for the whole polygon.

Compared with previous approaches, we provide a technique that is more stable than vertex-based optimization, more accurate than completely automatic reconstruction methods, and more automated than purely interactive methods. Our system provides a novel framework to efficiently utilize several important perceptual constraints and enables users to create a fast 3D model prototype based on a single image.

References

- [Bre73] BRENT R. P.: *Algorithms for Minimization without Derivatives*. Prentice Hall, 1973.
- [Chi95] CHING F.: *A visual Dictionary of Architecture*. Wiley, 1995.
- [CSB*05] COLEMAN P., SINGH K., BARRETT L., SUDARSANAM N., GRIMM C.: 3d screen-space widgets for non-linear projection. *GRAPHITE* (2005).
- [DTM96] DEBEVEC P. E., TAYLOR C. J., MALIK J.: Modelling and rendering architecture from photographs: A hybrid geometry- and image-based approach. *ACM TOG*, 3 (1996), 11–20.
- [HEM] HOIEM D., EFROS A. A., HERBERT M.: Automatic photo pop-up.
- [HZ03] HARTLEY R., ZISSERMAN A.: *Multiple View Geometry in computer vision*. Cambridge press, 2003.
- [IH01] IGARASHI T., HUGHES J.: A suggestive interface for 3d drawing. *Proceedings of UIST* (2001).
- [LS96] LIPSON H., SHPITALNI M.: Optimization-based reconstruction of a 3d object from a single freehand line drawing. *Computer-Aided Design* 28, 8 (1996), 651–663.
- [MKL05] MASRY M., KANG D., LIPSON H.: sketching interface for progressive construction of 3d objects. *Journal of Computers and Graphics* 29 (2005), 563–575.
- [SC04] SHESH A., CHEN B.: Smartpaper: An interactive and user friendly sketching system. *Computer Graphics Forum* 23, 3 (2004), 301–310.
- [SC05] SHESH A., CHEN B.: Peek-in-the-pic: Architectural scene navigation from a single picture using line drawing cues. *Pacific Graphics* (2005).
- [Ske] <http://www.sketchup.com>.
- [ZHH96] ZELEZNIK R. C., HERNDON K. P., HUGHES J.: Sketch: An interface for sketching 3d scenes. *ACM TOG*, 3 (1996), 163–170.

Supplementary Material

The following supplementary material is available for this article:

Video Clip S1.

This material is available as part of the online article from: <http://www.blackwell-synergy.com/doi/abs/10.1111/j.1467-8659.2007.01098.x>

(This link will take you to the article abstract).

Please note: Blackwell Publishing are not responsible for the content or functionality of any supplementary materials supplied by the authors. Any queries (other than missing material) should be directed to the corresponding author for the article.