Example-Based 3D Object Reconstruction from Line Drawings

Tianfan Xue[1], Jianzhuang Liu[1,2,3], and Xiaoou Tang[1,2]
[1] Department of Information Engineering, The Chinese University of Hong Kong
[2] Shenzhen Key Lab for CVPR, Shenzhen Institutes of Advanced Technology, China
[3] Media Lab, Huawei Technologies Co. Ltd., China
{xtf09, jzliu, xtang}@ie.cuhk.edu.hk

Abstract

Recovering 3D geometry from a single 2D line drawing is an important and challenging problem in computer vision. It has wide applications in interactive 3D modeling from images, computer-aided design, and 3D object retrieval. Previous methods of 3D reconstruction from line drawings are mainly based on a set of heuristic rules. They are not robust to sketch errors and often fail for objects that do not satisfy the rules. In this paper, we propose a novel approach, called example-based 3D object reconstruction from line drawings, which is based on the observation that a natural or man-made complex 3D object normally consists of a set of basic 3D objects. Given a line drawing, a graphical model is built where each node denotes a basic object whose candidates are from a 3D model (example) database. The 3D reconstruction is solved using a maximum-a-posteriori (MAP) estimation such that the reconstructed result best fits the line drawing. Our experiments show that this approach achieves much better reconstruction accuracy and are more robust to imperfect line drawings than previous methods.

1. Introduction and Related Work

A line drawing is a 2D projection of the wireframe of a 3D object. Reconstructing a 3D object from a 2D line drawing is an important and challenging task in computer vision. The applications of this work include: interactive 3D modeling from images [5], [9], [11], a flexible 2D sketch query interface for 3D object retrieval [3], [10], a user-friendly interface in CAD systems where a designer can sketch a 2D line drawing of a 3D model on paper or on the screen of a tablet PC [15], [19], and automatic 3D database generation from images with user sketches [1], [7].

Line drawing interpretation is one of the traditional topics in computer vision. The earliest work is line labeling [6], [23]. It searches for a set of consistent labels such as convex, concave, and occluding from a line drawing to test its correctness and/or realizability, but line labeling itself cannot recover the 3D shape from a line drawing.

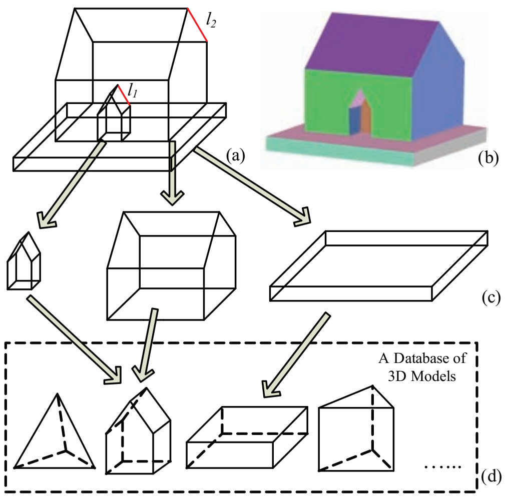


Figure 1. Illustration of example-based 3D reconstruction. (a) Input 2D line drawing. (b) Recovered 3D shape. (c) Separated 2D line drawings. (d) A database of 3D models.

The main purpose of line drawing interpretation is to reconstruct the 3D shape from a 2D line drawing. However, this reconstruction problem is intrinsically ill-posed due to the missing of one dimension. In order to circumvent this ill-posed problem, some researchers develop interactive methods that use additional information from the user. In [9] and [12], parametric 3D models are used as references for 3D reconstruction. In [21], a set of gestures is provides by the user to indicate the geometric relationship between parts. In [22], the user specifies parallelism and perpendicularity of lines. Usually, these interactive methods are only suitable for users with strong technical background in 3D geometry. Besides, the interaction is often manually intensive.

Rule-based automatic reconstruction from single 2D line drawings is a popular approach and has been studied extensively. Since there are an infinite number of 3D objects whose projections are the same line drawing, 3D object reconstruction from a 2D line drawing is to find the most plausible 3D object that is consistent with our visual system on the 3D interpretation of the line drawing. Finding appropriate rules for reconstruction greatly affects the reconstructed result. In previous methods, heuristic rules summarized from human visual perception are used to construct an objective function, the 3D object is obtained by minimizing this function [13], [14], [16], [17], [18], [24], [4]. One rule is to force all the angles at the vertices of a line drawing to be the same so that a 3D object can be inflated from the 2D line drawing. Line parallelism is another rule that two parallel lines in a 2D line drawing indicate that they are also parallel in 3D space. Other rules include face planarity, isometry, polyhedron symmetry, line verticality, skewed facial orthogonality, skewed facial symmetry, face perpendicularity, corner orthogonality, etc. Although the previous methods obtain good results in their experiments, those heuristic rules are not always satisfied in many cases. For example, in Figure 1(a), although lines l_1 and l_2 are nearly parallel in the line drawing, they are not parallel in 3D space. Besides, imperfect line drawings or sketch errors often cause the rules to be little useful, and there is no principled way of tuning the parameters that balance the heuristic rules.

In this paper, we propose a novel automatic approach called example-based 3D object reconstruction from line drawings. As previous related works, we consider planar-faced objects with all edges visible. The assumption in our approach is that a complex 3D object can be separated into simpler basic 3D models. This is true for most complex objects, especially man-made objects. For example, the 3D object shown in Figure 1(b) can be divided into three parts, two pentagonal prisms and one cuboid (see Figure 1(c)). Based on this assumption, we build a database of basic 3D models, as shown in Figure 1(d). A complex 2D line drawing is first decomposed into multiple smaller line drawings (Figure 1(c)), and multiple candidates (also called examples) are selected from the 3D model database for each small line drawing. Then an undirected graphical model is built where each node denotes a small line drawing. Based on this graphical model, the 3D reconstruction is solved using a maximum-a-posteriori (MAP) estimation that selects the best candidates (examples) so that the reconstructed result best fits the line drawing.

Compared with previous rule-based automatic methods, our approach has the following advantages: 1) It does not use any heuristic rules, and thus avoids the tuning of the parameters that balance the rules. The parameter tuning can be quite tricky; while they are suitable for one set of objects, they may cause many failures for another set of objects. 2) Our approach is more robust to sketch errors. The rules in previous methods are based on the local information of vertices and edges in a 2D line drawing, and imperfect line drawings may render many rules useless. Our approach, however, is based on both the 3D models from a database and a global optimization that chooses the best examples for the reconstruction.

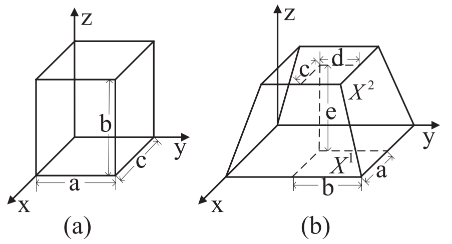


Figure 2. Two examples of 3D models in the database. (a) Cuboid. (b) Frustum of pyramid.

2. 3D Models in the Database

In this paper, a bold upper-case letter (say, $\mathbf{X}$) denotes the 3D coordinate of a point, and its 2D projection on the line drawing plane is denoted by the corresponding bold lower-case letter $\mathbf{x}$. A 2D line drawing is represented by $L = (\{\mathbf{x}^v\}, G)$, where $\mathbf{x}^1, \mathbf{x}^2, \ldots, \mathbf{x}^m$ are the 2D coordinates of the vertices of the line drawing, and G is an undirected graph indicating that which two vertices are connected. A recovered 3D shape from L is represented by $S = (\{\mathbf{X}^i\}, G)$, where $\mathbf{X}^1, \mathbf{X}^2, \ldots, \mathbf{X}^m$ are the 3D coordinates of vertices.

We manually build a database of 3D models for reconstruction. To increase the generalization ability of the database, the 3D shape of each model is controlled by a set of parameters. For example, the shape of the model *cuboid* is determined by three parameters: length a, width b and height c (shown in Figure 2(a)).

We assume that the 3D coordinate of each vertex of a model can be expressed as a linear function of a parameter vector. Formally, if a model has m vertices and n parameters, there is a set of $3 \times n$ matrices $\{A^1, A^2, \ldots, A^m\}$ such that the 3D Euclidean coordinate of the *i*-th vertex is $A^i\boldsymbol{\alpha}$, where $\boldsymbol{\alpha}$ is an n-dimensional vector containing all the parameters. For example, in the model *frustum of pyramid* (see Figure 2(b)), vertex $\mathbf{X}^1$ and vertex $\mathbf{X}^2$ are represented as:

$$\mathbf{X}^1 = \begin{pmatrix} 2a \\ 2b \\ 0 \end{pmatrix} = \begin{pmatrix} 2 & 0 & 0 & 0 & 0 \\ 0 & 2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} \boldsymbol{\alpha}, \quad (1)$$

$$\mathbf{X}^2 = \begin{pmatrix} a+c \\ b+d \\ e \end{pmatrix} = \begin{pmatrix} 1 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \boldsymbol{\alpha}, \quad (2)$$

where $\boldsymbol{\alpha} = (a, b, c, d, e)^\top$.

Each 3D model is defined by $M = (\{A^v\}, G)$, where A^v, $v = 1, 2, \ldots, m$, are the linear coefficient matrices defined above and G is an undirected graph denoting its topology. A 3D model represents a group of 3D objects. An

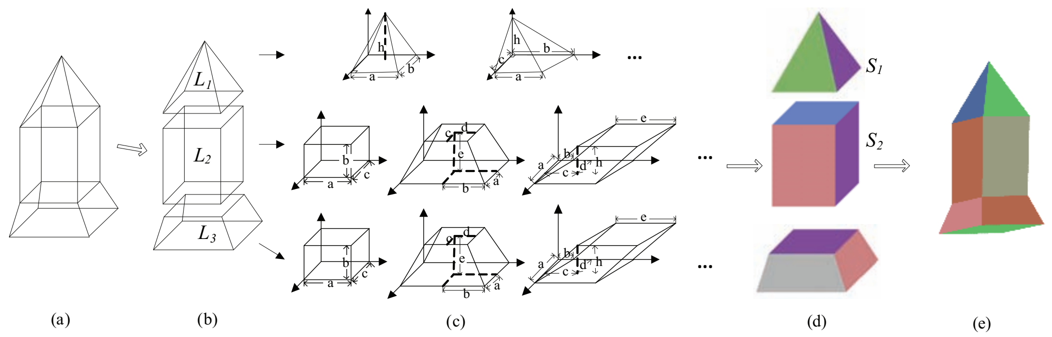


Figure 3. Reconstruction procedure. (a) Inputted 2D line drawing. (b) Basic line drawings separated from (a). (c) 3D model candidates. (d) Recovered 3D parts from the basic line drawings. (e) Final 3D object by combining the 3D parts together.

instance of this model is a 3D object $S = (\{\mathbf{X}_i\}, G)$ determined by a parameter vector $\boldsymbol{\alpha}$, a 3D rotation matrix R, and a 3D translation vector $\mathbf{t}$, where the 3D coordinate of the *i*-th vertex is $\mathbf{X}^i = RA^i\boldsymbol{\alpha} + \mathbf{t}$.

The current database contains 72 models. Since these models are parameterized, so model corresponds to innumerous 3D variations, thus they can represent most of basic 3D shapes. If a line drawing has a part that the database does not cover, our algorithm can automatically detect it and then it is added to the database.

3. Example-Based Reconstruction

The procedure of our approach is shown in Figure 3. First an input line drawing is decomposed into several basic line drawings (Figure 3(b)). Then for each basic line drawing, a set of 3D models which have the same topology as the basic line drawing are retrieved from the database (Figure 3(c)). After that, the best fitted part model for each basic line drawing, as well as corresponding parameters R, $\mathbf{t}$, and $\boldsymbol{\alpha}$ are estimated (Figure 3(d)). The final object is obtained by combining these small 3D objects, as shown in Figure 3(e). More details of these steps are described as follows.

Line drawing decomposition. We use the method proposed in [17] to separate a complex line drawing into multiple simple ones.

Candidate 3D model generation. For each basic line drawing $L_i = (\{\mathbf{x}_i^v\}, G_i)$, we find a set of candidate 3D part models that share the same topology as L_i. Specifically, for each part model $M = (\{A^v\}, G')$, if G_i and G' are isomorphic, then M is a candidate for L_i. We use the algorithm in [8] to check whether two graphs are isomorphic. Then for each basic line drawing L_i, there are n_i candidate 3D models: $M_{i,1}, \ldots, M_{i,n_i}$.

3D reconstruction. In this step, the best fitted part model for each basic line drawing is selected, as well as corresponding parameters R, $\mathbf{t}$, and $\boldsymbol{\alpha}$.

3D part combination. The 3D coordinate of each vertex of the combined 3D model is calculated as follows: if a vertex only belongs to one 3D part, its 3D coordinate is the coordinate in this part; if a vertex is shared by two or more 3D parts, its coordinate is the average of the corresponding coordinates in these parts.

The key step of this algorithm is the 3D reconstruction. We mainly focus on this step in the rest of this section.

3.1. Camera Model

In this paper, similar to most related work, orthogonal projection is assumed, whose projection matrix is

$$K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}. \tag{3}$$

Actually, our formulation in the next two sections is valid for other projections such as perspective projection. Only the inference in Section 3.4 needs to be changed if a different projection is used.

3.2. Problem Definition

The task of the 3D reconstruction is to estimate the shape of a 3D part corresponding to each basic line drawing L_i. We assume that each 3D part S_i is determined by a set of random variables $q_i = \{\mathbf{c}_i, R_{i,1}, \mathbf{t}_{i,1}, \boldsymbol{\alpha}_{i,1}, \ldots, R_{i,n_i}, \mathbf{t}_{i,n_i}, \boldsymbol{\alpha}_{i,n_i}\}$, where n_i is the number of candidate 3D models for the *i*-th basic line drawing, $\mathbf{c}_i$ is an n_i-dimensional indicator vector whose elements are defined as

$$c_i(k) = \begin{cases} 1, & \text{if the } k\text{-th candidate model is selected,} \\ 0, & \text{otherwise,} \end{cases}$$

and $R_{i,k}$, $\mathbf{t}_{i,k}$ and $\boldsymbol{\alpha}_{i,k}$ are the rotation matrix, the translation vector and the parameter vector for the k-th candidate model, respectively. The final recovered 3D object is obtained by combining these basic 3D objects together. Directly estimating q_i from L_i is an ill-posed problem, and so the following two constraints are imposed:

1) Projection constraint. The projection of each 3D part $S_i = (\{\mathbf{X}_i^v\}, G_i)$ on the 2D line drawing plane should be consistent with the corresponding decomposed line drawing $L_i = (\{\mathbf{x}_i^v\}, G_i)$. Considering sketch errors in the 2D line drawing, the 2D projections of the 3D object vertices $\mathbf{X}_i^v$ need not be strictly equal to $\mathbf{x}_i^v$, but as close to as possible.

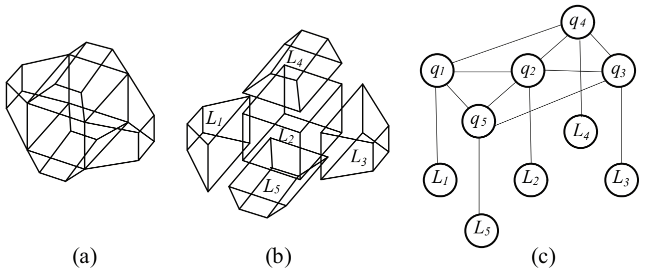


Figure 4. (a) Inputted 2D line drawing. (b) Decomposed basic line drawings L_{1-5}. (c) Graphical model of these line drawings. Observed nodes L_{1-5} are marked by shadow.

2) Construction constraint. The common 3D vertices of two neighboring 3D parts should be as close as possible. For example, in Figure 3(d), the bottom of S_1 and the top of S_2 have four common vertices. Their corresponding vertices should be as close as possible.

3.3. Undirected Graphical Model of Reconstruction

Given a line drawing $L = \{L_i\}$, using the MAP estimation, the best choice of $\{q_i\}$ should maximize the posteriori probability $P(\{q_i\}|L) \propto P(L|\{q_i\})P(\{q_i\})$. To formulate this probability, we assume there is a Markov property in $\{q_i\}$ and build an undirected graphical model as shown in Figure 4(c). Each observed node L_i denotes a basic line drawing L_i and each latent node q_i denotes the corresponding 3D part S_i. There are two kinds of edges in the graphical model. One is the edge connecting L_i and q_i, which ensures the projection constraint. The other is the edge connecting two neighboring 3D parts q_i and q_j, which ensures the construction constraint. With this graphical model and Markov property, we have $P(L|\{q_i\}) = \prod_i P(L_i|q_i)$ and $P(\{q_i\}) = \left(\prod_i \phi_i(q_i)\right)\left(\prod_{(i,j)\in Ne} \psi_{i,j}(q_i, q_j)\right)$, where Ne is the set of edges among $\{q_i\}$, and $\phi_i(\cdot)$ and $\psi_{i,j}(\cdot,\cdot)$ are potential functions [2]. Then the posterior probability is reformulated as

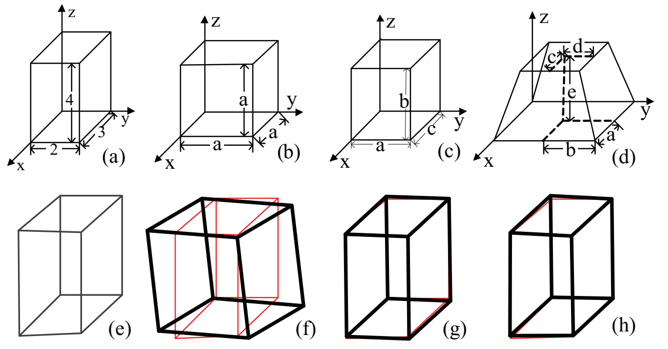
$$P(\{q_i\}|L) \propto \prod_i P(L_i|q_i) \cdot \prod_{(i,j)\in Ne} \psi_{i,j}(q_i, q_j) \cdot \prod_i \phi_i(q_i). \quad (4)$$

Let $E(L_i|q_i) = -logP(L_i|q_i)$, $E(q_i, q_j) = -log\psi_{i,j}(q_i, q_j)$, and $E(q_i) = -log\phi(q_i)$. Then maximizing (4) is equivalent to

$$\min_{\{q_i\}} \left(\sum_i E(L_i|q_i) + \sum_{\{i,j\}\in Ne} E(q_i, q_j) + \sum_i E(q_i) \right). \quad (5)$$

The first term $E(L_i|q_i)$ in (5) is the negative log likelihood term corresponding to the projection constraint, which is defined as

$$E(L_i|q_i) = \lambda_p \sum_{k=1}^{n_i} \left(c_i(k) \sum_{v\in V_i} ||K\mathbf{X}_{i,k}^v - \mathbf{x}_i^v||^2 \right), \quad (6)$$



$E(L\|q_i)$	11.54	1.86	1.75
$E(q_i)$	1	3	5
$E(L\|q_i)+E(q_i)$	12.54	4.86	6.75

Figure 5. (a) A 3D object. (b)–(d) Three 3D models. (e) An imperfect line drawing. (f)–(g) Best fitted results of the 3D models in (b)–(d) to the line drawings in (e). The 2D line drawing is drawn in red, and the best fitted results are drawn in bold. The table shows the projection errors, the negative priors, and their sums corresponding to the 3D models in (b)–(d).

where V_i is the set of vertices in the basic line drawing L_i, $\mathbf{X}_{i,k}^v = R_{i,k}A_{i,k}^v\boldsymbol{\alpha}_{i,k} + \mathbf{t}_{i,k}$ is the 3D coordinate of the vertex v after rotation and translation, λ_p is the weight for this term, $A_{i,k}^v$ is the matrix of the candidate 3D model $M_{i,k}$ that corresponds to the vertex v, and $\mathbf{x}_i^v$ is the 2D coordinate of the vertex v in the input line drawing.

The second term $E(q_i, q_j)$ in (5) corresponds to the construction constraint defined as

$$E(q_i, q_j) = \lambda_c \sum_{k=1}^{n_i} \sum_{l=1}^{n_j} \left(c_i(k)c_j(l) \sum_{v\in V_i\cap V_j} ||\mathbf{X}_{i,k}^v - \mathbf{X}_{j,l}^v||^2 \right), \quad (7)$$

where λ_c is the weight for this term. Notice that λ_p and λ_c are the only two parameters in this algorithm. Ideally the vertices shared by two parts should have exactly the same coordinates in these two parts, but here we only force them to be as close as possible in order to tolerate sketch errors.

The third term $E(q_i)$ denotes the negative prior of q_i. Different 3D models should have different prior probabilities. For example, the 3D shape shown in Figure 5(a) can be represented by the 3D model *cuboid* shown in Figure 5(c) with $a = 2$, $b = 4$, and $c = 3$, or be represented by *frustum of pyramid* shown in Figure 5(d) by setting $a = c = 2$, $b = d = 4$, and $e = 3$. However, human beings interpret this object as a cuboid other than a frustum of pyramid, meaning that cuboid has a higher prior probability. According to Gestalt psychology, one of the most influential theories with a long history, asserts that human beings are innately driven to perceive objects as simple as possible [20]. Therefore, it is reasonable to define $E(q_i)$ by the number of

Algorithm 1 Calculating the initial values of R, $\mathbf{t}$ and $\boldsymbol{\alpha}$

```
Initialization: Randomly generate initial value $R^{(0)}$, $\mathbf{t}^{(0)}$ and $\boldsymbol{\alpha}^{(0)}$; $i \leftarrow 0$.
  1. Fix $R^{(i)}$, find the optimal values of $\mathbf{t}$ and $\boldsymbol{\alpha}$ by
     solving $\begin{cases} f'_{\boldsymbol{\alpha}}(R,\mathbf{t},\boldsymbol{\alpha}) = 0, \\ f'_{\mathbf{t}}(R,\mathbf{t},\boldsymbol{\alpha}) = 0, \\ t(3) = 0, \end{cases}$
     and assign the solution to $\mathbf{t}^{(i+1)}$ and $\boldsymbol{\alpha}^{(i+1)}$.
  2. Fix $\mathbf{t}^{(i+1)}$ and $\boldsymbol{\alpha}^{(i+1)}$, and find the optimal value of $R^{(i+1)}$ using the algorithm in [25].
  3. If $|f(R^{(i)},\mathbf{t}^{(i)},\boldsymbol{\alpha}^{(i)}) - f(R^{(i+1)},\mathbf{t}^{(i+1)},\boldsymbol{\alpha}^{(i+1)})| < \epsilon$, then $i \leftarrow i+1$ and go to step 1.
Return $R^{i+1}$, $\mathbf{t}^{i+1}$ and $\boldsymbol{\alpha}^{i+1}$.
```

parameters $\eta_{i,k}$ in the model $M_{i,k}$ as

$$E(q_i) = \sum_{k=1}^{n_i} c_i(k)\eta_{i,k}. \tag{8}$$

The negative prior term $E(q_i)$ ensures the robustness of the algorithm. For example, given an imperfect line drawing as shown in Figure 5(e), the best fitted model should be the cuboid in Figure 5(c) as it has a small projection error and a high prior. The model cube in Figure 5(b) cannot fit this line drawing well as it has a large projection error although the prior is high. The model frustum of pyramid in Figure 5(d) is also a bad choice, because although it achieves an even slightly smaller projection error, it has a low prior. If no prior is considered, the model frustum of pyramid will be selected and the resulted 3D object will overfit to the sketch errors in the line drawing.

3.4. Inference in the Graphical Model

Finding the optimal solution to (5) is not a trivial problem, as it is subject to two non-convex constraints: the binary constraint $c_i(k) \in \{0,1\}$ and the orthogonal constraint $R_{i,k}^{\top}R_{i,k} = I_{3\times3}$. Besides, the objective function is a six-order polynomial. First, we relax the binary constraint to be a continuous linear inequality constraint $0 \le c_i(k) \le 1$. Then we design an alternative minimization algorithm to solve the problem. Next, we first discuss how to find a good initialization and then present the algorithm.

Initial value of $\mathbf{c_i}$. $\mathbf{c}_i$ has the weights for the candidate 3D models. We simply give an equal weight for all the candidates, i.e., to set the initial value $c_i(k)$ to $1/n_i$, where n_i is the number of candidate models for the basic line drawing L_i.

Initial values of $\mathbf{R_{i,k}}$, $\mathbf{t_{i,k}}$ and $\boldsymbol{\alpha}_{\mathbf{i,k}}$. We calculate the initial values of $R_{i,k}$, $\mathbf{t}_{i,k}$ and $\boldsymbol{\alpha}_{i,k}$ in each candidate 3D model ($k = 1,2,\ldots,n_i$) corresponding to L_i by minimizing a projection error:

$$\min f(R,\mathbf{t},\boldsymbol{\alpha}) = \sum_{v\in V} ||K(RA^v\boldsymbol{\alpha} + \mathbf{t}) - \mathbf{x}^v||^2,$$
$$\text{subject to:} \qquad R^{\top}R = I, \tag{9}$$

where the subscripts i and k for R, A^v, $\boldsymbol{\alpha}$ and $\mathbf{t}$, and the subscript i for $\mathbf{x}^v$ and V, are omitted for conciseness, V is the set of the vertices in L_i, $\mathbf{x}^v$ denotes the 2D coordinates of the vertices in L_i, and I is the identity matrix.

Equation (9) is also minimized using an alternative minimization algorithm. The algorithm is summarized in Algorithm 1. In step 1, when R is fixed, $f(R,\mathbf{t},\boldsymbol{\alpha})$ becomes a quadratic function of $\mathbf{t}$ and $\boldsymbol{\alpha}$. Its minimal value is achieved from the $\mathbf{t}$ and $\boldsymbol{\alpha}$ that are the solution to the two partial derivatives setting to 0. Notice that since the projection is along the z-axis, the translation along the z-axis $t(3)$ is irrelevant to the objective value $f(\cdot)$, where $t(3)$ denotes the third component of $\mathbf{t}$. So it is set to 0. Then in step 2, we fix $\mathbf{t}$ and $\boldsymbol{\alpha}$ and update R using the method in [25], which uses a gradient descent algorithm to find the minimizer of a differentiable objective function subject to an orthogonal constraint. Since the object value $f(R,\mathbf{t},\boldsymbol{\alpha})$ is decreasing in each run, a convergence is always guaranteed. To achieve a good initialization, Algorithm 1 runs multiple times, and the best result is automatically selected as the initial value for the following steps.

Initial values of translation along the z-direction. After running Algorithm 1 for every candidate 3D model $M_{i,k}$, we have initial $\mathbf{c}_i$, $R_{i,k}$, $\boldsymbol{\alpha}_{i,k}$, $t_{i,k}(1)$ and $t_{i,k}(2)$, $i = 1,\ldots,N$, $k = 1,\ldots,n_i$, where N is the number of decomposed line drawings, and $t_{i,k}(1)$ and $t_{i,k}(2)$ are the first two components of $\mathbf{t}_{i,k}$. Then we use (5) to estimate the initial $t_{i,k}(3)$ with these known $\mathbf{c}_i$, $R_{i,k}$, $\boldsymbol{\alpha}_{i,k}$, $t_{i,k}(1)$ and $t_{i,k}(2)$. Since the objective function in (5) is a quadratic function of $t_{i,k}(3)$, the optimal solution is obtained by setting its derivatives with respect to $t_{i,k}(3)$ to 0 and solving the resulting linear equations.

Solution to (5). After initialization, the solution to (5) is found as follows. For ease description, let $\widetilde{\mathbf{c}} = (\mathbf{c}_1,\ldots,\mathbf{c}_N)$, $\widetilde{R} = diag(R_{1,1},\ldots,R_{N,n_N})$, $\widetilde{\mathbf{t}} = (\mathbf{t}_{1,1},\ldots,\mathbf{t}_{N,n_N})$ and $\widetilde{\boldsymbol{\alpha}} = (\boldsymbol{\alpha}_{1,1},\ldots,\boldsymbol{\alpha}_{N,n_N})$, and denote the objective function in (5) as $g(\widetilde{\mathbf{c}},\widetilde{R},\widetilde{\mathbf{t}},\widetilde{\boldsymbol{\alpha}})$. Although $g(\cdot)$ is a six-order polynomial, it is a quadratic function if any three of $\widetilde{\mathbf{c}}$, $\widetilde{R}$, $\widetilde{\mathbf{t}}$, and $\widetilde{\boldsymbol{\alpha}}$ are fixed. Using this property, we design an alternative minimization algorithm listed in Algorithm 2. In step 1, by fixing $\widetilde{R}$, $\widetilde{\mathbf{t}}$, and $\widetilde{\boldsymbol{\alpha}}$, (5) becomes a quadratic programming problem. In step 2, by fixing $\widetilde{\mathbf{c}}$, $\widetilde{\mathbf{t}}$, and $\widetilde{\boldsymbol{\alpha}}$, (5) becomes a problem of minimizing a quadratic objective function with an orthogonal constraint. In step 3, by fixing $\widetilde{\mathbf{c}}$ and $\widetilde{R}$, (5) becomes a quadratic optimization problem with no constraints and is analytically solvable by setting the partial derivatives of the objective function with

```
Algorithm 2  Finding the optimal solution $\widetilde{\mathbf{c}}$, $\widetilde{R}$, $\widetilde{\mathbf{t}}$, and $\widetilde{\boldsymbol{\alpha}}$
Input: Initial values $\widetilde{\mathbf{c}}^{(0)}$, $\widetilde{R}^{(0)}$, $\widetilde{\mathbf{t}}^{(0)}$, and $\widetilde{\boldsymbol{\alpha}}^{(0)}$; $i \leftarrow 0$.
  1. Fix $\widetilde{R}^{(i)}$, $\widetilde{\mathbf{t}}^{(i)}$, and $\widetilde{\boldsymbol{\alpha}}^{(i)}$, and find $\widetilde{\mathbf{c}}^{(i+1)}$ by solving the quadratic programming problem:
       $\min_{\widetilde{\mathbf{c}}} \; g(\widetilde{\mathbf{c}}, \widetilde{R}^{(i)}, \widetilde{\mathbf{t}}^{(i)}, \widetilde{\boldsymbol{\alpha}}^{(i)})$
       subject to: $\mathbf{0} \le \widetilde{\mathbf{c}} \le \mathbf{1}, \sum_{k=1}^{n_i} c_i(k) = 1.$
  2. Fix $\widetilde{\mathbf{c}}^{(i+1)}$, $\widetilde{\mathbf{t}}^{(i)}$, and $\widetilde{\boldsymbol{\alpha}}^{(i)}$, and find $\widetilde{R}^{(i+1)}$ by solving
       $\min_{\widetilde{R}} \; g(\widetilde{\mathbf{c}}^{(i+1)}, \widetilde{R}, \widetilde{\mathbf{t}}^{(i)}, \widetilde{\boldsymbol{\alpha}}^{(i)}),$
       subject to: $\widetilde{R}\widetilde{R}^\top = I,$
     using the algorithm in [25].
  3. Fix $\widetilde{\mathbf{c}}^{(i+1)}$ and $\widetilde{R}^{(i+1)}$, and find $\widetilde{\mathbf{t}}^{(i+1)}$ and $\widetilde{\boldsymbol{\alpha}}^{(i+1)}$ by solving the linear equations:
       $\begin{cases} g'_{\widetilde{\mathbf{t}}}(\widetilde{\mathbf{c}}^{(i+1)}, \widetilde{R}^{(i+1)}, \widetilde{\mathbf{t}}, \widetilde{\boldsymbol{\alpha}}) = 0, \\ g'_{\widetilde{\boldsymbol{\alpha}}}(\widetilde{\mathbf{c}}^{(i+1)}, \widetilde{R}^{(i+1)}, \widetilde{\mathbf{t}}, \widetilde{\boldsymbol{\alpha}}) = 0. \end{cases}$
  4. If $|g(\widetilde{\mathbf{c}}^{(i)}, \widetilde{R}^{(i)}, \widetilde{\mathbf{t}}^{(i)}, \widetilde{\boldsymbol{\alpha}}^{(i)}) - g(\widetilde{\mathbf{c}}^{(i+1)}, \widetilde{R}^{(i+1)}, \widetilde{\mathbf{t}}^{(i+1)}, \widetilde{\boldsymbol{\alpha}}^{(i+1)})| < \delta$, then $i \leftarrow i+1$ and go to step 1.
Return $\widetilde{\mathbf{c}}^{(i+1)}$, $\widetilde{R}^{(i+1)}$, $\widetilde{\mathbf{t}}^{(i+1)}$, and $\widetilde{\boldsymbol{\alpha}}^{(i+1)}$.
```

respect to $\widetilde{\mathbf{t}}$ and $\widetilde{\boldsymbol{\alpha}}$ to 0. Same as algorithm 1, the convergence is always guaranteed. Since the total length of the hidden variables is linear to the number of parts, the computational efficiency of the algorithm does not decrease significantly when a line drawing grows complex.

After solving (5) in the continuous relaxation version, we obtain the binarized vector $\mathbf{c}_i$ by setting its maximum component to 1 and all the other components to 0.

4. Experiments

In this section, we demonstrate the effectiveness of our example-based 3D (E3D) reconstruction algorithm on various line drawings. We also compare it with two state-of-the-art methods, the plane-based optimization (PBO) in [16] and the divide-and-conquer (DAC) in [17]. Both [16] and [17] are rule-based reconstruction methods, and [17] can handle most complex objects than other previous methods.

We first test our algorithm on all the line drawings in [17] and find that it can reconstruct all the 3D objects successfully. Some line drawings in [17] are shown in Figure 6 (line drawings (a)–(d)), together with four new line drawings (e)–(h). From the relatively simple line drawing (a), all the three algorithms obtain good results. However, when a line drawing becomes complex with sketch errors, DAC

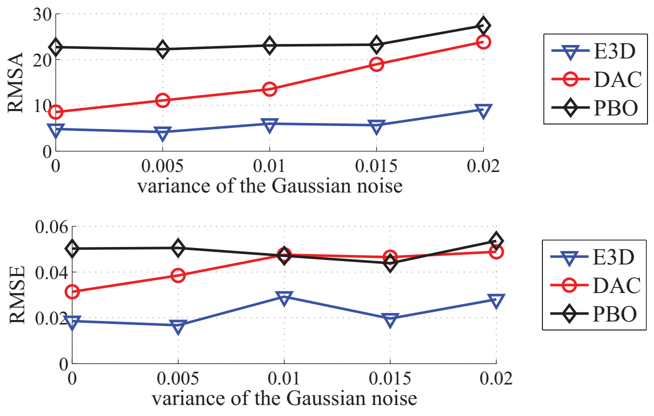


Figure 7. Performance comparison among E3D, DAC and PBO when different sketch errors are considered. All the line drawings are scaled into a rectangle with its diagonal equal to 1.

and PBO often generate bad results. The reconstructed 3D objects (b)–(h) in the last column of Figure 6 by PBO are all failed. DAC performs better than PBO, but its results (b)–(h) in the fourth column are distorted seriously (better viewed on the screen from the colored faces of the objects). In contrast, our algorithm E3D recovers much better objects from all the line drawings (see the second and third columns of Figure 6).

To further demonstrate the robustness of E3D over PBO and DAC, we test the three algorithms on line drawings with different levels of sketch errors. First, using AutoCAD, we manually build the 3D objects corresponding to the line drawings (b)–(h) in Figure 6. These objects are used as the ground truth. Then we simulate the sketch errors by adding Gaussian noise with zero mean and different variances to the 2D coordinates of the vertices of the projected line drawings from the ground truth. Finally, we compare the 3D objects reconstructed by the three algorithms with the ground truth objects. Two measurements are used to judge the reconstruction accuracy. One is the root mean square of angle differences (RMSA) between the ground truth object O_G and the reconstructed 3D object O_R, defined as

$$RMSA(O_G, O_R) = \sqrt{\frac{1}{N_a}\sum_{i=1}^{N_a}(\theta_G^i - \theta_R^i)^2}, \qquad (10)$$

where N_a is the number of angles, each of which is made by any two edges meeting at a vertex, and θ_G^i and θ_R^i are the i-th angles from O_G and O_R, respectively. The second measurement is the root mean square of Euclidean distances (RMSE) of corresponding vertices in O_G and O_R. Note that before computing $RMSE(O_G, O_R)$, the reconstructed object is aligned to O_G such that $RMSE(O_G, O_R)$ is minimized. From Figure 7, we can see that E3D performs much better than PBO and DAC. Even when the noise level is strong, it still has small RMSA and RMSE. By the way, the

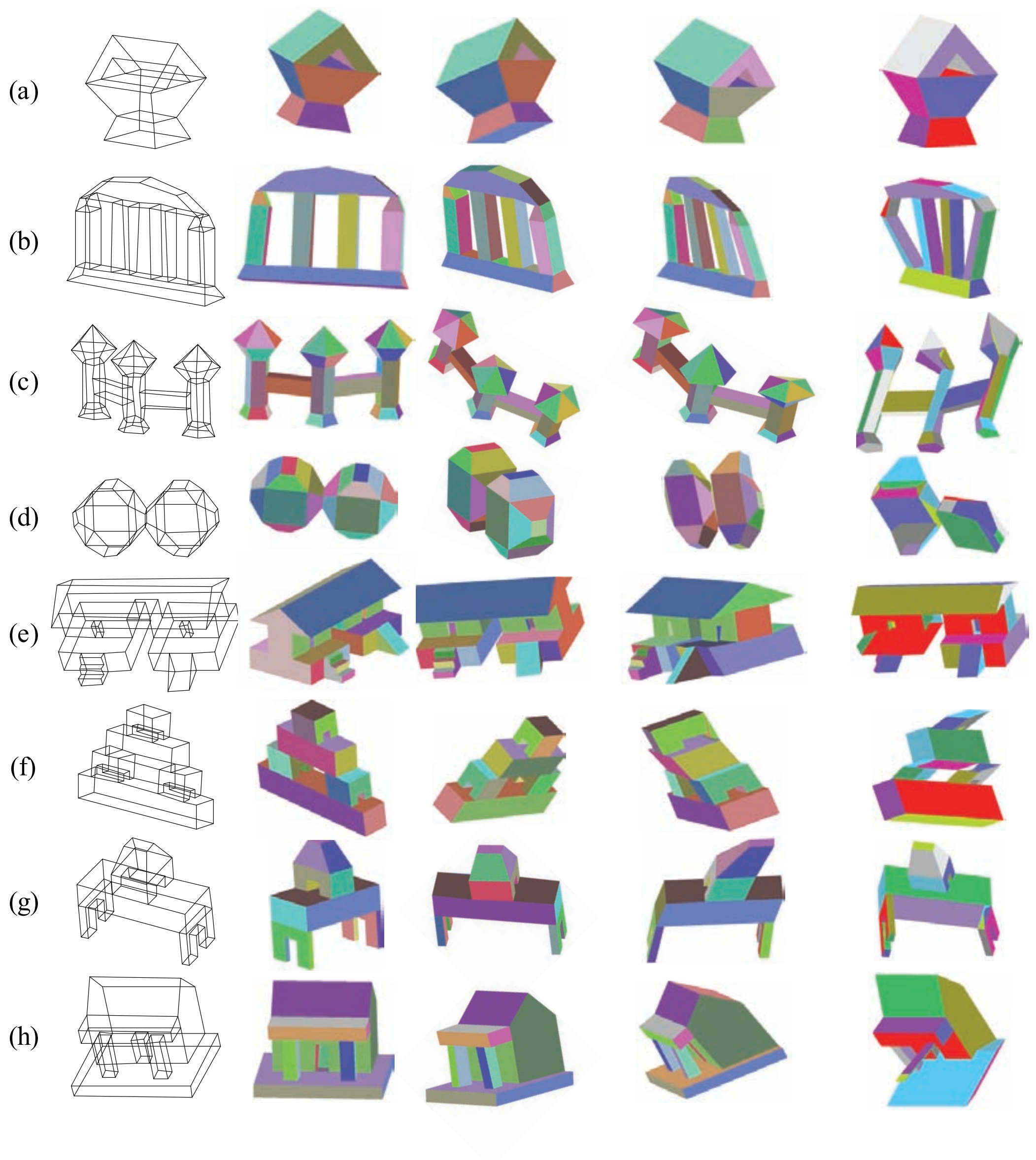


Figure 6. Experimental results on a set of complex line drawings (a)–(h) with sketch errors. The second and third columns show two views of the reconstructed objects by our algorithm E3D, the fourth column shows the objects recovered by DAC, and the last column shows the objects recovered by POA. Different colors are used to denote the faces. The results are better viewed on the screen.

line drawings (b)–(h) in Figure 6 are those with Gaussian noise of variance $= 0.01$ added.

In Figure 8, we show an application of our algorithm to 3D modeling from single images. Given an image, the user first sketches a line drawing along the visible edges and roughly guessed hidden edges of the objects in the image, as shown in Figure 8(b). Then the 3D shape is recovered by E3D, as shown in Figure 8(c) and Figure 8(d). Our algorithm is implemented in Matlab. On a PC with 2.4GHz Core2 CPU, it takes 12 minutes for each line drawing, and the initialization consumes the majority of the time.

Finally, it should be mentioned that it is possible that in the 3D model database, there is no 3D model whose topology is the same as a decomposed basic line drawing. For example, for a 3D object that cannot be composed into simple topologies, like icosahedron or footballene, it has no match in the database. However, in this case, we can still use PBO to recover the 3D part from this line drawing, and then merge this part with the other parts reconstructed by E3D.

5. Conclusion and Future Work

We have proposed a novel example-based 3D reconstruction algorithm to recover the 3D geometry from a 2D line drawing. In this approach, a complex line drawing is

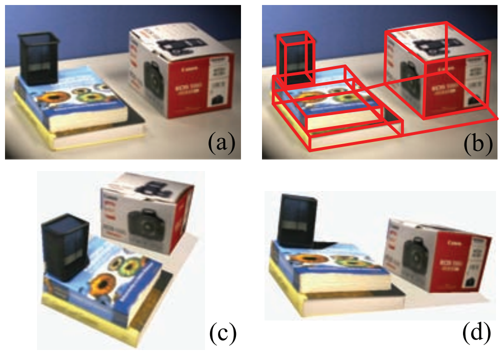


Figure 8. 3D modeling from an image. (a) Input image. (b) A line drawing sketched along the edges of the objects. (c) (d) Recovered 3D shape shown in two views.

first decomposed into multiple basic line drawings. Then a graphical model is built where each node denotes a basic object whose candidates are from a 3D model database. The 3D object is reconstructed using the MAP estimation. Our experiments show that this algorithm performs much bettern than two state-of-the-art algorithms.

Future work includes (i) the extension of this work to dealing with objects with curved faces, (ii) 3D modeling from single images when perspective projection is considered, and (iii) accelerating the inference step.

6. Acknowledgement

This work was supported by grants from Natural Science Foundation of China (60975029, 61070148), Science, Industry, Trade, Information Technology Commission of Shenzhen Municipality, China (JC200903180635A, JC201005270378A, ZYC201006130313A), and Guangdong Province through Introduced Innovative R&D Team of Guangdong Province 201001D0104648280.

References

[1] S. Agarwal, W. Waggenspack, et al. Decomposition method for extracting face topologies from wireframe models. *Computer-Aided Design*, 24(3):123–140, 1992.

[2] C. Bishop. *Pattern Recognition and Machine Learning*. Springer New York, 2006.

[3] L. Cao, J. Liu, and X. Tang. 3D object retrieval using 2D line drawing and graph based relevance feedback. *Proc. ACM Multimedia*, pages 105–108, 2006.

[4] L. Cao, J. Liu, and X. Tang. What the back of the object looks like: 3d reconstruction from line drawings without hidden lines. *IEEE Trans. PAMI*, 30(3):507–517, 2008.

[5] X. Chen, S. Kang, Y. Xu, J. Dorsey, and H. Shum. Sketching reality: Realistic interpretation of architectural designs. *ACM Transactions on Graphics*, 27(2):11, 2008.

[6] M. Clowes. On seeing things. *Artificial Intelligence*, 2:79–116, 1971.

[7] M. Cooper. Wireframe projections: physical realisability of curved objects and unambiguous reconstruction of simple polyhedra. *IJCV*, 64(1):69–88, 2005.

[8] D. Corneil and C. Gotlieb. An efficient algorithm for graph isomorphism. *Journal of the ACM*, 17(1):51–64, 1970.

[9] P. Debevec, C. Taylor, and J. Malik. Modeling and rendering architecture from photographs: A hybrid geometry-and image-based approach. *Proc. ACM SIGGRAPH*, pages 11–20, 1996.

[10] T. Funkhouser, P. Min, M. Kazhdan, J. Chen, A. Halderman, D. Dobkin, and D. Jacobs. A search engine for 3D models. *ACM Transactions on Graphics*, 22(1):83–105, 2003.

[11] F. Han and S. Zhu. Bayesian reconstruction of 3d shapes and scenes from a single image. *Proc. IEEE International Workshop on Higher-Level Knowledge in 3D Modeling and Motion Analysis*, pages 12–20, 2003.

[12] D. Jelinek and C. Taylor. Reconstruction of linearly parameterized models from single images with a camera of unknown focal length. *IEEE Trans. PAMI*, 23(7):767–773, 2001.

[13] Y. Leclerc and M. Fischler. An optimization-based approach to the interpretation of single line drawings as 3D wire frames. *IJCV*, 9(2):113–136, 1992.

[14] H. Lipson and M. Shpitalni. Optimization-based reconstruction of a 3D object from a single freehand line drawing. *Computer-Aided Design*, 28(8):651–663, 1996.

[15] H. Lipson and M. Shpitalni. Correlation-based reconstruction of a 3d object from a single freehand sketch. *Proc. ACM SIGGRAPH courses*, 2007.

[16] J. Liu, L. Cao, Z. Li, and X. Tang. Plane-based optimization for 3D object reconstruction from single line drawings. *IEEE Trans. PAMI*, 30(2):315–327, 2008.

[17] J. Liu, Y. Chen, and X. Tang. Decomposition of complex line drawings with hidden lines for 3d planar-faced manifold object reconstruction. *IEEE Trans. PAMI*, 33(1):3–15, 2011.

[18] T. Marill. Emulating the human interpretation of line-drawings as three-dimensional objects. *IJCV*, 6(2):147–161, 1991.

[19] M. Masry, D. Kang, and H. Lipson. A freehand sketching interface for progressive construction of 3d objects. *Proc. ACM SIGGRAPH 2007 courses*, 2007.

[20] S. Palmer. Vision Science: Photons to Phenomenology. *The MIT Press*, 1999.

[21] A. Shesh and B. Chen. Smartpaper: An interactive and user friendly sketching system. *Computer Graphics Forum*, 23(3):301–310, 2004.

[22] A. Shesh and B. Chen. Peek-in-the-pic: Flying through architectural scenes from a single image. *Computer Graphics Forum*, 27(8):2143–2153, 2008.

[23] K. Sugihara. Machine Interpretation of Line Drawings. *The MIT Press*, 1986.

[24] Y. Wang, Y. Chen, J. Liu, and X. Tang. 3D reconstruction of curved objects from single 2d line drawings. *CVPR*, 2009.

[25] Z. Wen and W. Yin. A feasible method for optimization with orthogonality constraints. Technical report, Rice University, 2010.