Computers & Graphics 46 (2015) 130–137

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

SMI 2014

Sketch-based 3-D modeling for piecewise planar objects in single images ☆

Changqing Zou [a,b,c], Xiaojiang Peng [a], Hao Lv [b], Shifeng Chen [b,*], Hongbo Fu [d], Jianzhuang Liu [e,f]

[a] Department of Physics and Electronic Information Science, Hengyang Normal University, Hengyang, China
[b] Shenzhen Key Lab of C.V.P.R., Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China
[c] Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, China
[d] City University of Hong Kong, Hong kong
[e] The Chinese University of Hong Kong, Hong Kong
[f] Media Laboratory, Huawei Technologies Co., Ltd., Shenzhen, China

ARTICLE INFO

Article history:
Received 8 July 2014
Received in revised form
26 September 2014
Accepted 27 September 2014
Available online 13 October 2014

Keywords:
3-D modeling
Piecewise planar objects
Single images
Sketch-based

ABSTRACT

3-D object modeling from single images has many applications in computer graphics and multimedia. Most previous 3-D modeling methods which directly recover 3-D geometry from single images require user interactions during the whole modeling process. In this paper, we propose a semi-automatic 3-D modeling approach to recover accurate 3-D geometry from a single image of a piecewise planar object with less user interaction. Our approach concentrates on these three aspects: (1) requiring rough sketch input only, (2) accurate modeling for a large class of objects, and (3) automatically recovering the invisible part of an object and providing a complete 3-D model. Experimental results on various objects show that the proposed approach provides a good solution to these three problems.

1. Introduction

In recent years, 3-D applications such as 3-D TV and movies, 3-D city, and virtual reality are in rapid development. For these applications, the creation of 3-D models is still one of the main bottlenecks. Generally, tools used to create 3-D models can be classified into two categories: (1) hardware tools (e.g., 3-D cameras) and (2) computer-aided design tools. Compared with the former approach, the latter is less costly and has been extensively studied.

3-D object reconstruction from single images provides a significant interface for a 3-D model design software, which has attracted considerable attention recently, though this topic is challenging due to the ill-posed nature of the problem. In this scenario, intensive interactive 3-D modeling methods [4,7,8,10,13,18,21,22,24,25] have been proposed to reconstruct 3-D objects in single images. In general, some methods where user's interactions are needed throughout the whole modeling process (e.g., [8], and [24]) obtain good modeling results at the cost of intensive user interactions. Some other methods need less user interaction but only focus on reconstructing some special objects or object categories. For instance, the method in [22] reconstructs inflectionally symmetric objects in single images, based on a few user marks which are used to label the symmetric information; the method in [7] assumes that the desired object can be modeled as a polyhedron where the coordinates of the vertices can be expressed as a linear function of a dimension vector. Some other methods achieve efficient 3-D modeling but sacrifice or neglect the accuracies of the results. For example, reconstruction errors might accumulate easily during face-by-face propagation in [18]. The method by Xu et al. [21] only require reconstructed 3-D shapes to be consistent with human perception. 3-D models generated by the 3-sweep technique [4] are only rough approximations of target objects.

In this work, we aim to provide an efficient tools to reconstruct up-to-a-scale piecewise planar objects from single images (for concision, we use “objects” to denote “piecewise planar objects” in the remainder of this paper). In general, there are three main challenges in this topic. The first one is how to provide a rapid modeling of a desired object, i.e., the modeling system must be easy to use for a common user who has less background knowledge. The second one is how to precisely recover a up-to-a-scale 3-D model. The last one is how to obtain a complete 3-D model for an object with invisible parts in an image.

☆The work described in this paper was partially supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project no. CityU 113513), Guangdong Innovative Research Team Program (No. 201001D0104648280), and the Construct Program of the Key Discipline in Hunan Province.

* Corresponding author.
E-mail address: shifeng.chen@siat.ac.cn (S. Chen).

http://dx.doi.org/10.1016/j.cag.2014.09.031
0097-8493/

To address the above three problems, several previous methods called *semi-automatic* methods, such as [7,10,13,15], first manually obtain the 2-D wireframe of a desired object and then automatically reconstruct the complete 3-D object via recovering the 3-D depths (z coordinates) of the vertices in the corresponding 2-D wireframe. Given a 2-D wireframe of a desired object, the 3-D reconstructions in semi-automatic methods are completely automatic. Therefore, semi-automatic methods greatly simplify the work of the users, compared to those methods where user interactions are needed throughout the whole modeling process. However, there are still several disadvantages for these methods. Firstly, drawing a complete and precise 2-D wireframe of an object is not very tractable and takes time; secondly, the positions of the invisible vertices drawn by the user are imperfect in terms of precision, and they are treated as constants, i.e., the 2-D positions of the invisible vertices are fixed during the reconstruction, which results in a consequent deterioration in reconstruction quality.

We propose a sketch-based 3-D modeling approach to address all these three problems in a unified framework. The pipeline of our method is shown in Fig. 1. Given an input image, the user first roughly sketches the visible vertices and edges of an object of interest. Based on the manually drawn sketch and the edge information from a line segment detector, the system first generates an initial 2-D wireframe of the visible part of the desired object (Section 2). Then an optimization algorithm is proposed to reconstruct the precise 3-D wireframe of the visible part (see Section 3). Further, the invisible part of the object is automatically recovered based on the 3-D structure of the visible part (Section 4).

Our approach also bears close resemblance to the previous semi-automatic methods in [7,10,13,15], but is advantageous in two aspects. First, in terms of the efficiency of user interface, our method automatically derives initial 2-D wireframes from the input users' sketches and detected line segments. Therefore, it is more convenient to use and lessens the user interactions. Second, in terms of the reconstruction accuracy, the good performance of the proposed method mainly comes from two facts:

(i) the 2-D positions of the vertices in the 2-D wireframe whose precise locations are uncertain are treated as variables, and they are jointly optimized together with the depths of all the visible vertices during the 3-D reconstruction of the visible part. Therefore, our approach can reconstruct a more precise 3-D visible part than those reconstructing the 3-D visible part from an inaccuracy 2-D wireframe where the x- and y-coordinates of all the vertices are fixed;

(ii) after the invisible topological structure is obtained, the 3-D positions of the invisible vertices can usually be computed based on the recovered 3-D visible part directly. Therefore, our method usually produces a more precise 3-D invisible part than previous methods.

The rest of this paper is organized as follows: Section 2 details how to generate an initial 2-D wireframe W_{2di} based on users' sketches. Sections 3 and 4 present the algorithms which reconstruct the visible part M_{v3dw} and the invisible part M_{h3dw} of a desired object, respectively. Section 5 gives the experimental results and some discussions.

2. Generation of initial 2-D wireframe

2.1. User's sketches and the initial 2-D wireframe

We require the user to provide three types of interaction information (as shown in Fig. 2) to generate the initial 2-D wireframe W_{2di}. The first type includes (approximately) closed contours, we call them *sketch vertices*. These sketch vertices encircle the possible locations of the visible vertices, and are drawn by the user with one stroke. The second type consists of the lines connecting two sketch vertices, called *sketch lines*, which are used to represent the connectivity of the vertices. The last type includes the artificial lines used to indicate the coplanar relationship of two sketch lines (it can be extended to cover more geometrical relationship). Note that the artificial lines are not the components of the 2-D wireframe. The sketch vertices and sketch lines are automatically recognized from the following criteria:

(i) (approximately) closed contours finished by one stroke are treated as sketch vertices;

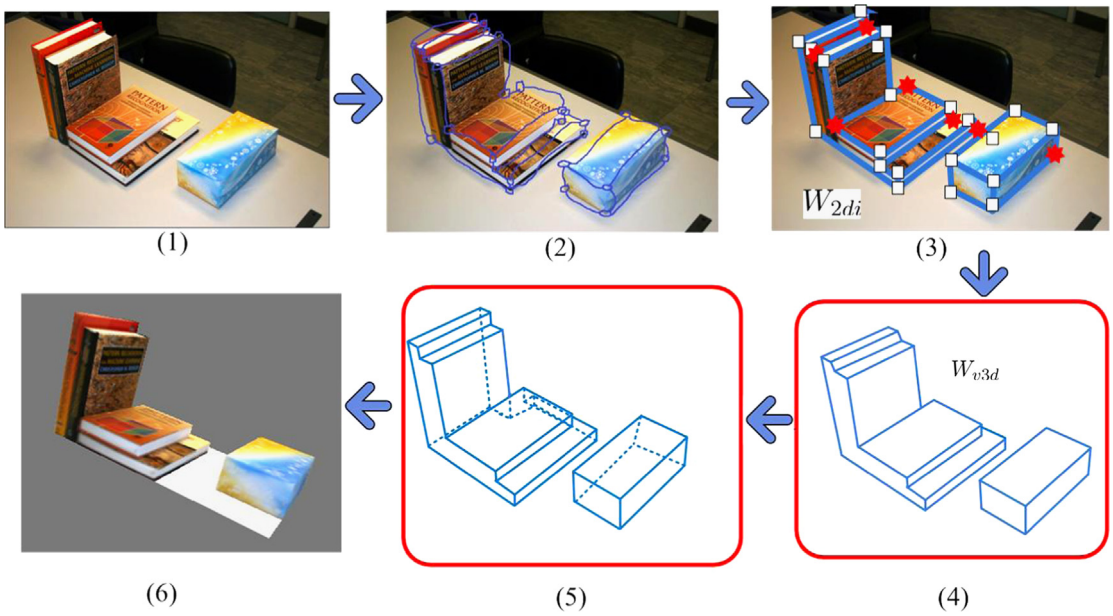

Fig. 1. Flow chart of the proposed approach. (1) The original image. (2) Roughly sketched vertices and edges for the visible part of the desired object. (3) Automatically generated 2-D initial wireframe W_{2di}. Note that some vertices in W_{2di} have no precise locations, which are highlighted with red stars. (4) Precise reconstruction of 3-D wireframe M_{v3dw} from W_{2di}. (5) Inference of the invisible 3-D geometries (marked with dotted lines). (6) Complete 3-D objects. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

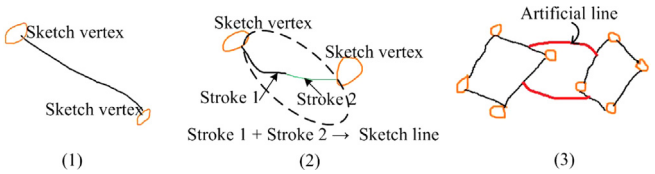

Fig. 2. (1) A sketch line which connects two sketch vertices. It consists of one stroke. (2) Another sketch line consisting of two strokes (note that a sketch line may consist of multiple strokes). (3) Two artificial lines used to indicate the coplanar relationship of two sketch lines.

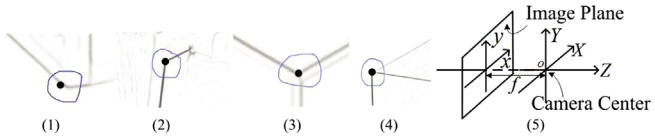

Fig. 3. Four examples showing inferring the precise position of one vertex by two or more distinct intersecting segments.

(ii) a stroke which does not form a sketch vertex and connects two sketch vertices is treated as a sketch line;
(iii) a stroke which does not form a sketch vertex (i.e., approximately closed contour) and only connects one sketch vertex, called a *broken sketch line*, is treated as one part of a sketch line;
(iv) a stroke which does not form a sketch vertex but connects a broken sketch line is merged into the broken sketch line to generate a new broken sketch line or sketch line;
(v) a stroke connecting two (broken) sketch lines belongs to an artificial line;
(vi) a stroke which neither forms a sketch vertex nor connects any sketch vertex or broken sketch line is treated as an invalid input.

After the user finishing the sketch vertices and sketch lines, the system obtains the connectivity of the vertices, i.e., the topology information, and the coarse locations of the visible vertices. Then segments are detected by a fast line segment detector algorithm [20] from the contours generated by [1] (in this step the contour generation algorithm in [23] is also a good alternative). From these detected segments the system automatically computes the precise locations of the vertices (encircled by sketch vertices) which are the intersections of two or more distinct segments (some example cases in our experiments are shown in Fig. 3(1–4)).

If there are multiple intersections inside a single sketch vertex, we select the intersection closer to the centroid of the sketch vertex as the vertex location. For a vertex which is not passed by two or more segments, its initial location is set to the centroid of the corresponding sketch vertex. If two detected line segments have endpoints which are close but not quite intersect within the encircled range of a sketch vertex, we first check if the distance of these two endpoints d is shorter than the approximate radius r of the sketch vertex. If $d < r$, we stretch both of these endpoints and set the vertex location to their intersection. Otherwise, the vertex location is set to the centroid of the sketch vertex. Once the system obtains the positions in the image for all the visible vertices of the object, the segments of 2-D wireframe which connects two vertices are generated according to the topological information from sketch lines. Finally, the system generates the initial 2-D wireframe W_{2di} whose partial vertices have inaccurate positions (2-D coordinates in the image).

In the next section, we describe how to jointly infer two kinds of information: (1) the precise 2-D locations of the vertices with rough positions (i.e., the vertices whose locations are set to the centers of the corresponding sketch vertices) and (2) the precise depths for all the vertices in W_{2di}.

3. 3-D modeling of visible parts

3.1. Target problem, challenges, and motivation

The task that automatically reconstructs a 3-D object from its 2-D wireframe is challenging, even if the x- and y-coordinates of all the vertices in this 2-D wireframe are fixed. The classical solutions for this problem are to transform the 3-D reconstruction problem into an optimization problem [7,21] or a problem solving a system of equations [13] (to some extent, these two solutions are similar since a system of equations is usually solved by an optimization algorithm). In these solutions, geometrical and topological constraints are built from 2-D cues, based on image regularities, to construct an objective function or a system of equations. For example, Li et al. [13] used the constraints of perspective symmetry and face connectivity to construct a system of linear equations with a closed-form solution.

In general, the success of the reconstructions in these solutions depends on whether enough *valid constraints* can be built for objective functions or systems of equations. Actually, it is very difficult to build enough constraints that are completely valid for a large classes of objects because of two reasons. First, most constraints only suit some special categories of objects, e.g., the constraints of perspective symmetry in [13] only work for objects with a certain number of bilateral symmetrical faces. Second, some constraints may introduce outliers, e.g., it is possible that a detected perspective symmetrical face in a 2-D wireframe corresponds to an asymmetric planar face in space.

The target problem in this section is to reconstruct a precise up-to-a-scale piecewise 3-D shape from the initial 2-D wireframe W_{2di}, which is more challenging compared to the similar problem in [13]. That is because the positions of some vertices in W_{2di} are variable, while the positions of all the vertices in [13] are fixed. We utilize a new RANSAC-like optimization scheme to solve the 3-D reconstruction problem from the initial 2-D wireframe. To handle a wider range of objects, we develop more geometrical properties to build the objective function of the underlying optimization problem. The pipeline of this scheme is as follows: Given W_{2di}, we first reconstruct a rough 3-D shape O_{3dr}. We then analyze O_{3dr} and detect all valid constraints for the next optimization process (the detection process of valid constraints here is similar to the selection of applicable shape regularities that are used to lift a sketch curve network into 3-D in True2Form [21]). Finally, we jointly optimize the x- and y-coordinates of the vertices that have no precise 2-D positions and the depths of all the vertices. In addition to handling more object classes, our method can generate a more precise 3-D object than the one in [13], since only valid constraints are utilized.

3.2. Assumptions, calibration, and normal vector

We use a simplified camera model in this paper. The camera has zero skew and does not have radial distortion, and the aspect ratio of the pixel equals 1. In this model, the projection matrix is

$$\mathbf{k} = \begin{pmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{pmatrix}, \tag{1}$$

where f is the focal length of the camera. We first align the world frame with the camera frame as shown in Fig. 3(5), where the image plane is $Z = -f$, and the projection matrix takes a simple

form $\mathbf{P}=[\mathbf{K}|\mathbf{0}]$. We obtain the intrinsic parameters of the camera model in the same procedure as [13].

In the rest of this paper, a bold upper-case letter (say, $\mathbf{X}=(X,Y,Z,1)^{\mathrm{T}}$) denotes the homogeneous coordinate of a 3-D point, and its 2-D projection on the image plane is denoted by the corresponding bold lower-case letter $\mathbf{x}=(x,y,1)^{\mathrm{T}}$ (homogeneous coordinate). Since the image plane is put at $Z=-f$, the 3-D Euclidean coordinate of $\mathbf{x}$ is $(x,y,-f)^{\mathrm{T}}$, denoted by $\tilde{\mathbf{x}}$. The 2-D position of $\tilde{\mathbf{x}}$ in the image plane (i.e., (x,y)) is presented as $\mathbf{x}'$.

A plane $n_x x+n_y y+n_z z+1=0$ is represented by $\boldsymbol{\pi}=(n_x, n_y, n_z, 1)^{\mathrm{T}}=(\mathbf{n}^{\mathrm{T}},1)^{\mathrm{T}}$, where $\mathbf{n}=(n_x,n_y,n_z)^{\mathrm{T}}$ is the normal of the plane (face). Thus, a vertex in 3-D space lying on the plane $n_x x+n_y y+n_z z+1=0$ can be formulated as $\boldsymbol{\pi}^{\mathrm{T}}\mathbf{X}=0$. We use the algorithm in [12] to identify all the faces in W_{2di}, and similar to [13], represent the desired (partial) objects in an image with a vector consisting of all the normals of the faces of the objects (i.e., $\mathbf{q}=(\mathbf{n}_1^{\mathrm{T}},\mathbf{n}_2^{\mathrm{T}},\ldots,\mathbf{n}_{N_f}^{\mathrm{T}})^{\mathrm{T}}$, where N_f is the number of faces). This vector $\mathbf{q}$ is called the normal vector of a desired object in this paper. Next, we discuss the relation between the normal vector $\mathbf{q}$ and several constraints.

3.3. Geometric and topological constraints

In this subsection, we discuss two new geometric constraints, which will be used together with the constraints *connectivity* (C_{co}), *constraint by fixing one vertex* (C_{fv}), and *perspective symmetry* (C_{ps}) in [13] to recover the 3-D geometries of the 2-D wireframes of wider objects.

Line parallelism (C_{lp}): We first develop a constraint to show the relation of a face $\mathbf{f}$ with normal vector $\mathbf{n}$ and a pair of 3-D parallel lines which are parallel to the plane that $\mathbf{f}$ lies on.

Proposition 1. *If two lines, $\mathbf{L}_1$ and $\mathbf{L}_2$, and a plane $\boldsymbol{\pi}$ are parallel in 3-D space, where $\mathbf{l}_1$ and $\mathbf{l}_2$ are the projections of $\mathbf{L}_1$ and $\mathbf{L}_2$ in the image, respectively, then the normal $\mathbf{n}$ of $\boldsymbol{\pi}$ satisfies*

$$\mathbf{n}^{\mathrm{T}}(\mathbf{K}^{-1}(\mathbf{l}_1\times\mathbf{l}_2))=0. \tag{2}$$

Proof. Let $\mathbf{v}$ be the vanishing point of $\mathbf{L}_1$, the direction $\mathbf{R}$ of $\mathbf{L}_1$ satisfies $\mathbf{R}=\mathbf{K}^{-1}\mathbf{v}$ [6]. Then we have $\mathbf{v}=\mathbf{l}_1\times\mathbf{l}_2$, since $\mathbf{L}_1$ and $\mathbf{L}_2$ are parallel. We further have $\mathbf{n}\perp\mathbf{R}$ since $\mathbf{L}_1$ and $\boldsymbol{\pi}$ are parallel, and thus $0=\mathbf{n}^{\mathrm{T}}\mathbf{R}=\mathbf{n}^{\mathrm{T}}\mathbf{K}^{-1}(\mathbf{l}_1\times\mathbf{l}_2)$.

According to Proposition 1, for a given plane $\boldsymbol{\pi}_i$, we have

$$\mathbf{C}_{\boldsymbol{\pi}_i}\mathbf{q}=\mathbf{0}, \tag{3}$$

by putting the constraints from all the pairs of parallel lines of $\boldsymbol{\pi}_i$ together. In our implementation, we select the pairs of lines (say $\mathbf{l}_1$ and $\mathbf{l}_2$) in an image as the projections of the parallel lines in the underlying 3-D object, if $\mathbf{l}_1$ and $\mathbf{l}_2$ pass through a common vanishing point (the method in [9] is used to automatically detect all the vanishing points in an image). We use the truth one plane $\boldsymbol{\pi}$ is parallel to a line $\mathbf{L}$ in 3-D space if $\boldsymbol{\pi}$ does not pass through $\mathbf{L}$ and $\boldsymbol{\pi}$ has at least one parallel line of $\mathbf{L}$, to find all the parallel pairs which satisfy Proposition 1. □

Orthogonal corner (C_{oc}): We use the constraint C_{oc} in [11] to detect possible pairs of mutually perpendicular faces in a 2-D wireframe. This constraint can be formulated as

$$\mathbf{n}_j^{\mathrm{T}}\mathbf{n}_k=g_{(j,k)}(\mathbf{q})=0, \tag{4}$$

where $\mathbf{n}_j$ and $\mathbf{n}_k$ are the normals of a pair of mutually perpendicular faces.

3.4. Reconstruction of the rough 3-D shape

In this subsection, we state how to reconstruct a rough 3-D shape O_{3dr} from W_{2di}. The basic idea of this step is that we first build a system of equations according to C_{co}, C_{ps}, C_{fv}, C_{lp}, and C_{oc}, and then solve the system using quasi-Newton search algorithm [17], to obtain $\mathbf{q}$, finally compute O_{3dr} according to $\mathbf{q}$.

The proposed system of equations, which includes five sets of equations:

$$\begin{cases}\mathbf{A}^{(0)}\mathbf{q}=\mathbf{0} & (a)\\ \mathbf{B}^{(0)}\mathbf{q}=\mathbf{0} & (b)\\ \mathbf{E}^{(0)}\mathbf{q}=f/Z_0 & (c)\\ \mathbf{C}^{(0)}\mathbf{q}=\mathbf{0} & (d)\\ \sum_{i=1}^{n} g_i(\mathbf{q})=0 & (e)\end{cases} \tag{5}$$

where the linear equations (5a), (5b), and (5c) are respectively built from C_{co}, C_{ps}, and C_{fv} (see Eqs. (7), (10), and (13) in [13]), linear equations (Eq. (5d)) are obtained by grouping the constraints C_{lp} described in Eq. (3) together, quadratic equations (Eq. (5e)) include all the constraints C_{oc} described in Eq. (10).

In Eq. (5e), n denotes the pair number of mutually perpendicular faces. Note that matrixes $\mathbf{A}^{(0)}$, $\mathbf{B}^{(0)}$, $\mathbf{C}^{(0)}$, $\mathbf{E}^{(0)}$ are computed from W_{2di} and their values are known (i.e, we treat the vertices whose initial locations are set to the centers of the corresponding sketch vertices, as fixed vertices in this step).

3.5. Constraint refinement and joint optimization

After obtaining a rough 3-D shape O_{3dr} from W_{2di}, in return we use O_{3dr} to detect all valid constraints in Eq. (5) for W_{2di}. The detected valid constraints is further used to jointly infer the precise 2-D locations of the vertices who have not been fixed, and the normal vector $\mathbf{q}$.

In our implementation, the threshold angle θ_{thp} of the parallel relationship is empirically set to 7°, and the threshold angle θ_{tho} of the orthogonal relationship is empirically set to 83° (e.g., two faces correspond to a valid constraint described by C_{oc} if their intersection angle θ satisfies $83^\circ\le\theta\le90^\circ$). The bilateral symmetry of a face is judged by the average, say θ_{as}, of the intersection angles of the symmetry axis and the lines connecting pairs of symmetric vertices (i.e., a valid face of bilateral symmetry should satisfy $83^\circ\le\theta_{as}\le90^\circ$).

The precise 3-D reconstruction task from W_{2di} is formulated as a constrained optimization problem. The objective function is defined as

$$\begin{aligned}\min\quad & F(\mathbf{q},\mathbf{x}'_{\mathbf{r}})=|\mathbf{A}_{\mathbf{u}}\mathbf{q}|^2+|\mathbf{B}_{\mathbf{u}}\mathbf{q}|^2+|\mathbf{C}_{\mathbf{u}}\mathbf{q}|^2+\sum_{i=1}^{m}|g_i(\mathbf{q})|^2,\\ \text{s.t.}\quad & \mathbf{A}_{\mathbf{d}}\mathbf{q}=\mathbf{0},\quad \mathbf{B}_{\mathbf{d}}\mathbf{q}=\mathbf{0},\quad \mathbf{C}_{\mathbf{d}}\mathbf{q}=\mathbf{0},\\ & \mathbf{E}\mathbf{q}=f/Z_0,\quad |\mathbf{x}'_{r_i}-\mathbf{x}'_{r_{i_0}}|^2<t_{r_i}^2,\ r_i\in\mathbf{r},\end{aligned} \tag{6}$$

where $\mathbf{r}$ is the set of the vertex indexes corresponding to rough 2-D positions in W_{2di}, $\mathbf{x}'_{\mathbf{r}}$ denotes the set of the (x,y) coordinates of the vertices whose indexes are in $\mathbf{r}$, $\mathbf{x}'_{r_i}$ with initial value $\mathbf{x}'_{r_{i_0}}$ set to the centers of sketch vertices as discussed in Section 2, denotes the 2-D position of its r_i th elements. $\mathbf{A}_{\mathbf{d}}$, $\mathbf{B}_{\mathbf{d}}$, $\mathbf{C}_{\mathbf{d}}$, and $\mathbf{E}$ only consist of the constraints contributed by the fixed 2-D vertices (i.e., these matrixes are known), $\mathbf{A}_{\mathbf{u}}$, $\mathbf{B}_{\mathbf{u}}$, and $\mathbf{C}_{\mathbf{u}}$ denotes the constraints contributed by the unfixed 2-D vertices (i.e., these matrixes have unknown variables in $\mathbf{x}'_{\mathbf{r}}$), t_{r_i} is the approximate radius of the corresponding sketch vertex (i.e., $t_{r_i}=l/(2\pi)$, and l is the perimeter of the r_i th sketch vertex), $g_i(\mathbf{q})$ denotes a pair of valid orthogonal faces in W_{2di}, and m denotes the number of pairs of valid mutually perpendicular faces.

Eq. (6) is minimized using an alternate minimization algorithm. The algorithm is summarized in Algorithm 1. In the algorithm, the initial values $\mathbf{q}^{(0)}$ and $\mathbf{x}'_{\mathbf{r}}(0)$ are taken from O_{3dr}. For the two variables $\mathbf{q}$ and $\mathbf{x}'_{\mathbf{r}}$, we alternately fix one variable and optimize

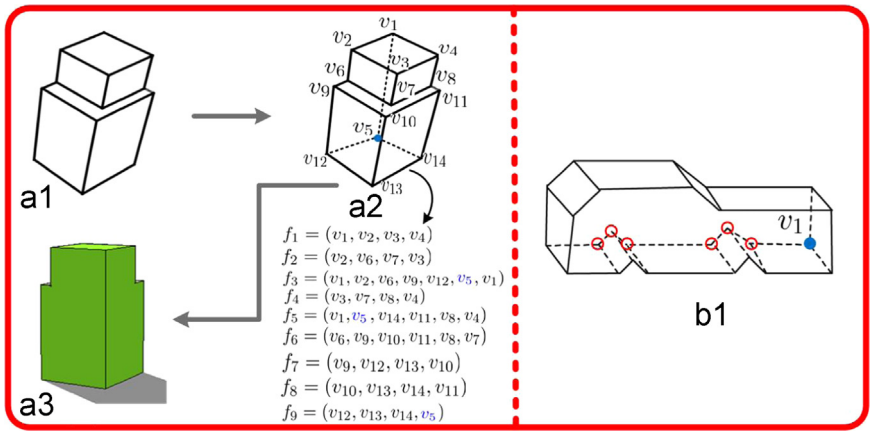

Fig. 4. (a1) 3-D wireframe M_{v3dw} of the visible part of a test object O_t. (a2) Invisible topological structure M_{h3dw} (dotted lines) obtained by the algorithm in [3]. Note that the invisible vertex in blue only has initial *x*-, *y*-, and *z*-coordinates. (a3) 3-D invisible structure reconstructed using Steps 1 and 2 of Algorithm 2. (b1) An object whose invisible part cannot be completely reconstructed using Steps 1 and 2 of Algorithm 2. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

the other one until the objective function converges. In our implementation, we solve the optimization problems using the *lsqnonlin* function in MATLAB. After the execution of Algorithm 1, we compute all the Z-coordinate Z_i by $Z_i = f/(\mathbf{n}_i^{\mathrm{T}}\tilde{\mathbf{x}})$ for all vertices $(\tilde{\mathbf{x}} = (x, y, -f)^{\mathrm{T}})$, if it is on *i*th face with normal $\mathbf{n}_i$. We take the average as the final Z-coordinate Z if there are multiple Z_i. Then we use $X = -Zx/f, Y = -Zy/f$ to compute the X-coordinate and Y-coordinate of the vertex. After the above steps, the system obtains the precise 3-D wireframe of the visible part of the desired object. In the next section, we discuss how to reconstruct the invisible part based on this 3-D wireframe.

Algorithm 1. Finding the optimal solution $\mathbf{q}$ and $\mathbf{x}'_{\mathbf{r}}$.

Input: Initial values $\mathbf{q}^{(0)}$ and $\mathbf{x}'_{\mathbf{r}}(0)$; $i \leftarrow 0$.

(i) Fix $\mathbf{x}'_{\mathbf{r}}(i)$, and find $\mathbf{q}^{(i+1)}$ by solving the linear-constrained quadratic programming problem:

$$\min_{\mathbf{q}} \quad F(\mathbf{q}) = |\mathbf{A}_{\mathbf{u}}^{(i)}\mathbf{q}|^2 + \mathbf{B}_{\mathbf{u}}^{(i)}\mathbf{q}|^2 + |\mathbf{C}_{\mathbf{u}}^{(i)}\mathbf{q}|^2 + \sum_{j=1}^{m} |g_i(\mathbf{q})|^2,$$

$$\text{s.t.} \quad \mathbf{A}_{\mathbf{d}}\mathbf{q} = \mathbf{0}, \quad \mathbf{B}_{\mathbf{d}}\mathbf{q} = \mathbf{0}, \quad \mathbf{C}_{\mathbf{d}}\mathbf{q} = \mathbf{0}, \quad \mathbf{E}\mathbf{q} = f/Z_0.$$

(ii) Fix $\mathbf{q}^{(i+1)}$, and find $\mathbf{x}'_{\mathbf{r}}(i+1)$ by solving

$$\min_{\mathbf{x}'_{\mathbf{r}}} \quad F(\mathbf{x}'_{\mathbf{r}}) = |\mathbf{A}_{\mathbf{u}}\mathbf{q}^{(i+1)}|^2 + \mathbf{B}_{\mathbf{u}}^{(i)}\mathbf{q}|^2 + |\mathbf{C}_{\mathbf{u}}\mathbf{q}^{(i+1)}|^2,$$

$$\text{s.t.} \quad |\mathbf{x}'_{r_i} - \mathbf{x}'_{r_{i_0}}|^2 < t_{r_i}, \quad r_i \in \mathbf{r}.$$

(iii) if $|F(\mathbf{q}^{(i+1)}, \mathbf{x}'_{\mathbf{r}}(i+1)) - F(\mathbf{q}^{(i)}, \mathbf{x}'_{\mathbf{r}}(i))| < \delta$, then $i \leftarrow i+1$ and go to step (i).

Return: $\mathbf{q}^{(i+1)}$ and $\mathbf{x}'_{\mathbf{r}}(i+1)$

4. 3-D modeling of invisible parts

Based on the visible 3-D structure M_{v3dw} of a desired object, we next take two steps to recover the invisible 3-D structure M_{h3dw}. We first infer the topological structure of M_{h3dw}, and then derive the 3-D locations of all the vertices in M_{h3dw}.

4.1. Inference of the invisible topological structure

The algorithms deriving the invisible structure of a natural line drawing in the work [3,5,19] can be borrowed for the inference of the invisible topological structure of M_{v3dw}. In both [5,19], the steps for the inference of the invisible structure have to be conducted based on the results of a line labeling step. Different from these two methods, the authors in [3] directly derive the invisible structure of a natural line drawing. They first construct an initial invisible structure according to the assumption that all the vertices in the desired object are trihedral and several theorems developed by this assumption. And then they generate a set of possible invisible structures by reducing the initial invisible structure with a strategy of cutting-and-merging of edges and vertices. Finally, they select the most plausible structure based on the visual psychological properties from Gestalt psychology. This algorithm can recover an invisible structure less than 1 s when dozens of invisible vertices are contained in a line drawing.

In our work, we utilize the algorithm in [3] to derive the invisible 2-D topological structure of M_{v3dw} because this algorithm is efficient and has relatively short pipeline. The steps for inferring the invisible structure of a natural line drawing can be directly used here since M_{v3dw} includes all the information of its corresponding natural line drawing. Note that all the invisible vertices in the new generated invisible structure have only initial 3-D locations (see Fig. 4(a2)). All the *x*-, *y*-, and *z*-coordinates of these invisible vertices need to be found.

4.2. Derivation of the 3-D locations of all the invisible vertices

We summarize the algorithm deriving the 3-D locations of all the invisible vertices (indexed by Algorithm 2) as the following three steps:

Step 1. Find the face topology of the complete object.
Step 2. Compute the 3-D coordinates of the invisible vertices which are passed through by three known non-coplanar planes, recursively.
Step 3. Optimize the 3-D positions of the remaining invisible vertices.

In Step 1, the algorithm in [14] is used to identity all the faces of the desired object O_t. The face topology is very important for the derivation of the 3-D coordinates of the invisible vertices. For the invisible vertices which are passed through by three known non-coplanar planes, their 3-D coordinates can be directly computed. As shown in Fig. 4(a1–a3), the 3-D coordinates of vertex v_5 can be computed by the three known planes passing through f_2, f_5, and f_9.

In some cases, there are some other invisible vertices which are not passed through by three known non-coplanar planes, however their 3-D coordinates may be obtained by recursively performing Step 2. Our experiments find that the 3-D coordinates of all the invisible vertices can be recovered only by Steps 1 and 2 in most cases. For the invisible vertices whose 3-D coordinates cannot recover only by the property of coplanarity in some cases (e.g., the invisible vertices encircled by red circle in Fig. 4(b1)), we use an optimization-based approach to tackle the reconstruction problem. The objective function consists of two components. The first one is a symmetry measure, and the other one is about planarity. Both of these two components can only be used for the target structure composed of the faces that contain the remaining invisible vertices in Step 3.

We use the symmetry measure S defined as

$$S = \frac{A}{P^2}, \tag{7}$$

to measure the symmetry of an underlying face. In (7), A and P are the area and perimeter of the boundary cycle of the measured face respectively. It holds that $S \le 1/(4\pi)$ for any closed planar figure [2]. A cycle is the most symmetrical planar figure with $S = 1/(4\pi)$. The target 3-D structure consists of more than three faces, each being a polygon. We consider it as the integration of all the planar faces containing the invisible vertices described in Step 3. Thus, the whole symmetry measure of the target 3-D structure with n faces is defined as

$$Symmetry = \sum_{i=1}^{n} \frac{P_i^2}{A_i}, \tag{8}$$

where A_i and P_i, $1 \le i \le n$, are the area and perimeter of face i in the target structure respectively.

In the second component, we evaluate the derivation from planarity for the vertices on each face in the target 3-D structure. Given a face f_i represented by a vector (a_i, b_i, c_i), assume that it contains m vertices, then the derivation from planarity for all these m vertices on f_i can be computed by

$$DP_i = \frac{1}{a_i^2 + b_i^2 + c_i^2} \sum_{i=1}^{m} (a_i x_{ij} + b_i y_{ij} + c_i z_{ij} - 1)^2, \tag{9}$$

where x_{ij}, y_{ij}, and z_{ij} denote the 3-D coordinates of the i th vertex in the i th face. For the target 3-D structure with n faces, the total deviation from planarity is defined as

$$Planarity = \sum_{i=1}^{n} DP_i. \tag{10}$$

In the target 3-D structure, the faces can be divided into two categories. The first one consists of the faces containing more than three vertices in the visible 3-D structure M_{v3dw}, except for several vertices whose 3-D coordinates are unknown. For such face types, we directly compute the plane vector (a_i, b_i, c_i) using the 3-D positions of all the visible vertices. This plane vector can be obtained by the least square fitting technique [11]. The second category includes the faces containing less than three vertices in the visible 3-D structure M_{v3dw}. For each face f_i in this category, we obtain the best fit plane vector (a_i, b_i, c_i) using the 3-D positions of all the vertices in f_i.

Finally, the objective function to be minimized is defined as

$$f(x_1, y_1, z_1, \ldots, x_u, y_u, z_u) = Planarity + w \cdot Symmetry, \tag{11}$$

where $x_1, y_1, z_1, \ldots, x_u, y_u, z_u$ are the x-, y-, and z-coordinates of the u vertices in the target structure, w is a parameter used to balance the contribution of the *Planarity* and *Symmetry* values to the objective function. In our implementation w is chosen to be 100 because *Planarity* is usually much larger than *Symmetry* if $w = 1$. Minimizing f makes the target structure as symmetrical as possible with the constraint of planarity.

5. Experiments and discussion

In our experiments, we compared our system with the one in [13]. We implemented the interface of our system and that in the system proposed by Li et al. [13] using C++, and the reconstruction algorithms in these two methods using MATLAB. Our evaluation focused on three parts: (1) user interface, (2) reconstruction accuracy, and (3) invisible part reconstruction. We first set up six test scenes (shown in Fig. 1 and Fig. 6(1)–(5)), for which the ground truth data for evaluations could be easily obtained, and then took one test photo for each scene in a generic view. We regarded connected planar-faced objects in a scene as a single complete desired object. These scenes were mainly used for the evaluation of user interface and reconstruction accuracy.

Study interface: Similar to some previous experimental designs such as [16], we conducted a user study to evaluate the performance of the user interfaces of the two compared systems. We recruited 12 novice participants, all of them with little graphics background and aged from 22 to 32. Before using a new system each participant was given a short tutorial session (10–15 min) and practiced until they felt comfortable to model one simple object (the one in Fig. 6(6)) with the associated user interface. Each participant was then required to model one object for each of the six test images (the one in Fig. 1 and the five images in Fig. 6 (1)–(5)), using each of the two systems, whose order for each test image was counterbalanced. In sum, there were in total 12 (participants) $\times$ 6 (images) $\times$ 2 (systems) = 144 trials. The participants were allowed to take breaks between two modeling tasks but no breaks during the process of modeling an object.

All 12 participants found that each sketch vertex or sketch line in the user input with our system was identified instantly (less than 1 s). It was the same case for the vertex or edge identification in [13]. On average each participant took 106, 118, 124, 101, 71, and 39 s to generate the initial 2-D wireframes for the six test images with our system. Correspondingly, the average times to draw 12 complete 2-D wireframes for the method in [13], were 142, 156, 163, 137, 128, and 57 s, respectively. Such statistics indicate that our interface was more efficient for the tested examples. The times for the reconstruction algorithms of our system varied from 3 to 17 s, while the system [13] took from 2 to 4 s. Note that the reconstruction processes in both systems took much less times than the user drawing processes. Therefore, our system was overall more efficient. As shown next, although our reconstruction algorithm was slightly slower, it was more accurate.

Reconstruction evaluation: The second set of experiments was set to evaluate the accuracy of the proposed reconstruction approach. We first scaled up all the reconstructed objects to the same sizes as the ground truth. The reconstruction error for a reconstructed object was then defined as follows:

$$RErr(O_{3d}) = \frac{1}{N_{box}} \sum_{j}^{N_{box}} \frac{\sum_i^8 dis(v_i^{(j)}, \overline{v}_i^{(j)})}{8 \times \sqrt[3]{L_j \times W_j \times H_j}}, \tag{12}$$

where $dis(v_i^{(j)}, \overline{v}_i^{(j)})$ denotes the 3-D distance between a pair of corresponding vertices, which belong to the ground-truth and the reconstructed object, respectively. L_j, W_j and H_j are the respective length, width and height of the j-th cuboid in the ground-truth. N_{box} is the total number of cuboids in the reconstructed object O_{3d}. For a common test image, the mean and variance of the 12 reconstruction errors from the 12 users for each system were used for reconstruction quality evaluation.

It can be easily seen from Fig. 5 that our system led to more precise objects reconstructed from single images. This is because it

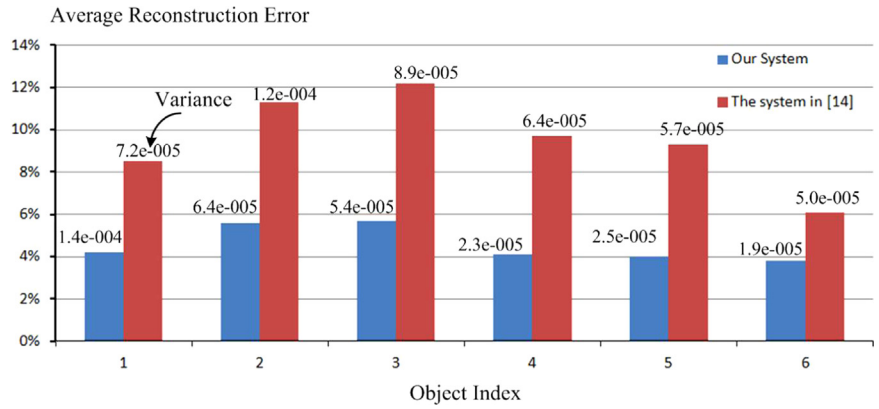

Fig. 5. Mean and variance of reconstruction errors measured using Eq. (12) for each object in six test scenes. Note that scene index 1 corresponds to the scene in Fig. 1 and scene indexes 2–6 correspond to the scenes in Fig. 6(1)–(5).

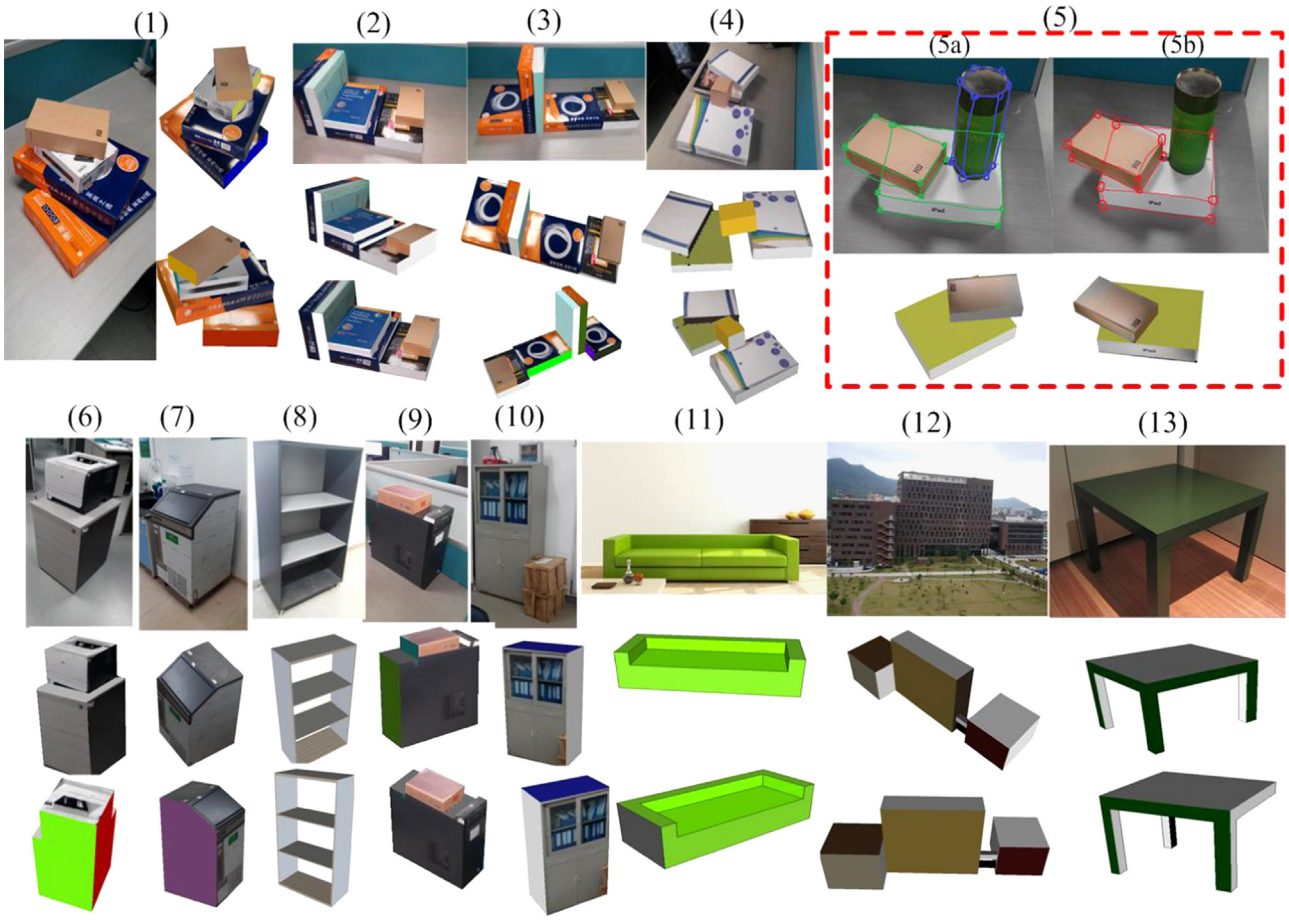

Fig. 6. Part of test images and their reconstructed results. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

is difficult for a user to precisely draw 2-D positions of invisible vertices, causing significantly higher reconstruction errors. In contrast, our system did not suffer from this problem in most cases, since the precise 3-D coordinates of the invisible vertices were directly computed based on the 3-D structure of the visible part in one reconstruction, as discussed in Section 4.

We also found that the qualities of the user-drawn sketches (sketch vertex quality, especially) have affected both the reconstruction accuracies and efficiencies. For the objects in Fig. 6(5), the times of the reconstruction from the sketched 2-D wireframe in green where vertices in objects were tightly encircled by sketch vertices, and that in red where some vertices in the object were not encircled by sketch vertices, were 3 s and 5 s, respectively. The reconstruction accuracies from the blue wireframe and the red one were 6% and 10%, respectively. It indicates that a high-quality sketched 2-D wireframe, with object vertices tightly encircled by sketch vertices, can result in more efficient and more accurate reconstruction.

Evaluation of invisible part reconstruction: Lastly we tested the capability of our system on reconstructing the invisible parts of

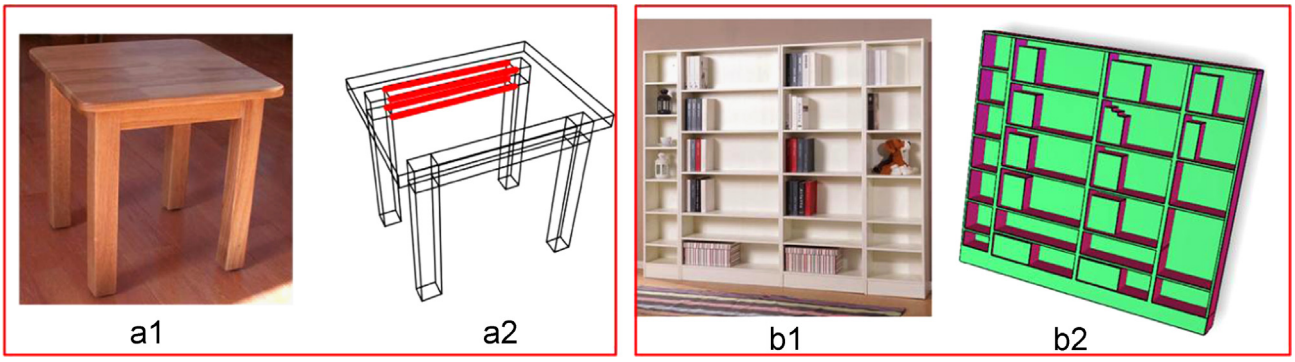

Fig. 7. (a1) and (a2) illustrate a limitation of our system. (b1) and (b2) show the capability of our system on modeling complex objects, including 43 connected cuboids in this example. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

various piecewise-planar objects in both indoor and outdoor environments. Several representative test images and their corresponding recovered objects are shown in the bottom row of Fig. 6. Among these test images, each of the objects in Fig. 6(6) and (7) has only one invisible vertex. For these two objects, Steps 1 and 2 in Algorithm 2 can be used to recover the 3-D locations of the invisible vertices. For the objects in Fig. 6(8)–(11), each one has more than two invisible vertices, the 3-D positions of their invisible vertices can also be completely reconstructed by Steps 1 and 2 in Algorithm 2, since each of these invisible vertices is passed through by three planes whose parameters can be determined by the 3-D visible structures. For the architecture shown in Fig. 6(12), Step 3 in Algorithm 2 has to be used for the reconstruction of the invisible vertices since the property of coplanarity only works for partial invisible vertices.

For the desk shown in Fig. 6(13), the reconstruction result missed the desk leg that is completely invisible in the image. A similar example is shown in Fig. 7(a), where our system failed to automatically reconstruct the 3-D structure marked in red. This is mainly caused by the invisible topological structure inference step (Section 4.1). These problems can be easily addressed by sketching out the 2-D structures of the completely invisible elements, and/or duplicating the visible part to the invisible part with a user-annotated global symmetry plane [21].

Discussions: We also evaluated the capability of our system on modeling more complex objects. For example, for the object in Fig. 7(b) involving 43 cuboids, it took 12 participants about 15 min (varying from 13 to 17 min) to draw sketches and generate initial 2-D wireframes. It took less than 2 min (varying from 91 to 126 s) to reconstruct complete objects from these initial 2-D wireframes. The reconstruction process might be significantly speeded up by first decomposing a complex object into several parts using the algorithm in [26] and then performing incremental reconstructions of individual parts.

It is possible to extend our method for the reconstruction of curve objects. For example, the cylindrical object in Fig. 6(5) can be reconstructed by first recovering a complete hexagonal prism from the user-drawn 2-D wireframe in blue and then generating a cylindrical box that fits perfectly inside the hexagonal prism.

Our current system requires that all the invisible vertices in desired objects must be trihedral and the invisible structures have no holes or pockets. It is limited by the assumptions used for the inference of the 2-D topological invisible structure in [3]. If the 2-D topological invisible structures can be provided by users' sketches, our system can successfully output precise complete objects.

References

[1] Arbelaez P, Maire M, Fowlkes C, Malik J. Contour detection and hierarchical image segmentation. IEEE Trans. PAMI 2011;33(5):896–916.
[2] Berger M. Geometrie, CEDIC/Fernand Nathan, 1977.
[3] Cao L, Liu J, Tang X. What the back of the object looks like: 3D reconstruction from line drawings without hidden lines. IEEE Trans Pattern Anal Mach Intell 2008;30(3):507–17.
[4] Chen T, Zhu Z, Shamir A, Hu S, Cohen-Or D. 3-Sweep: extracting editable objects from a single photo. ACM Trans. Graph 2013;32(6).
[5] Grimstead I, Martin R. Creating solid models from single 2D sketches. In: Proceedings of the 3th ACM symposium on solid modeling and applications, 1995. pp. 323–37.
[6] Hartley R, Zisserman A. Multiple view geometry in computer vision. Cambridge, England: Cambridge University Press; 2004.
[7] Jelinek D, Taylor CJ. Reconstruction of linearly parameterized models from single images with a camera of unknown focal length. IEEE Trans. PAMI 2001;23(7):767–73.
[8] Jiang N, Tan P, Cheong L. Symmetric architecture modeling with a single image. ACM Trans. Graph 2009;28(3):113.
[9] Kalantar M, Jung F, Guedon J. Precise, automatic and fast method for vanishing point detection. Photogramm Record 2009;24(127):246–63.
[10] Lau M, Saul G, Mitanni J, Igarashi T. Modeling-in-context: user design of complementary objects with a single photo. In: Proceedings of the SBM, 2010. pp. 17–24.
[11] Lipson H, Shpitalni M. Optimization-based reconstruction of a 3D object from a single freehand line drawing. Comput Aided Des 1996;28(8):651–63.
[12] Liu J, Lee Y. Graph-based method for face identification from a single 2D line drawing. IEEE Trans. PAMI 2001;23(10):1106–19.
[13] Li Z, Liu J, Tang X. A closed-form solution to 3d reconstruction of piecewise planar objects from single images. In: Proceedings of the CVPR, 2007.
[14] Liu J, Lee YT, Cham WK. Identifying faces in a 2D line drawing representing a manifold object. IEEE Trans. PAMI 2002;24(12):1579–93.
[15] Liu J, Cao L, Tang X. Plane-based optimization for 3D object reconstruction from single line drawings. IEEE Trans. PAMI 2008;30(2):315–27.
[16] Ma C, Liu Y, Yang H, Teng D, Wang H, Dai G. KnitSketch: a sketch pad for conceptual design of 2D garment patterns. IEEE Trans. Autom Sci Eng 2011;8(2):431–7.
[17] Press W, Teukolsky S, Vetterling W, Flannery B. Numerical recipes in C++: the art of scientific computing. Cambridge, England: Cambridge University Press; 2002.
[18] Sturm P, Maybank S. A method for interactive 3d reconstruction of piecewise planar objects from single images. In: Proceedings of the BMVC, 1999.
[19] Varley PAC. Automatic creation of boundary-representation models from single line drawings [PhD thesis]. University of Wales; 2003.
[20] von Gioi RG, Jakubowicz J, Morel J, Randall G. LSD: a fast line segment detector with a false detection control. IEEE Trans. PAMI 2010;32(4):722–32.
[21] Xu B, Chang W, Sheffer A, Bousseau A, McCrae J, Singh K. True2Form: 3D curve networks from 2D sketches via selective regularization. ACM Trans Graph 2014;33(4).
[22] Xue T, Liu J, Tang X. 3-D modeling from a single view of a symmetric object. IEEE Trans. Image Processing 2012;21(9):4180–9.
[23] Yu C, Liu Y, Wu MT, Li K, Fu X. A global energy optimization framework for 2.1D sketch extraction from monocular images. Graph Models 2014;76(5):507–21.
[24] Zhang L, Dugas-Phocion G, Samson JS, Seitz SM. Single-view modelling of free-form scenes. J Vis Comput Animat 2002;13(4):225–35.
[25] Zou C, Liu JB, Liu J. Precise 3D reconstruction from a single image. In: ACCV, 2012.
[26] Zou C, Yang H, Liu J. Separation of Line Drawings Based on Split Faces for 3D Object Reconstruction. In: Proceedings of the CVPR, 2014.