Multimed Tools Appl (2016) 75:10153–10174
DOI 10.1007/s11042-015-2966-x

Recovering solid geometric object from single line drawing image

Jinxin Zheng[1] · **Yongtao Wang**[1] · **Zhi Tang**[1]

Received: 19 May 2015 / Revised: 1 September 2015 / Accepted: 22 September 2015 /
Published online: 5 October 2015

Abstract Many educational materials contain a lot of solid geometric figures. The solid geometric objects in these figures are usually drawn as 2D line drawings thus have lost their 3D information. This paper presents a method to recover the 3D information of the solid geometric object from single line drawing image taken from the geometric books, which would be used to help the users better present and understand the solid geometric object on their mobile devices. The main advantage of our method is the ability to handle inaccurately processed sketches as opposed to the previous methods which require perfect line drawings as inputs. Our method consists of three main steps as follows. First, the sketch of the input line drawing image is automatically extracted and further represented as an undirected graph. Second, candidate 3D models from the pre-built 3D model database are found by graph similarity-based searching and sub-graph isomorphism matching. Third, for each candidate 3D model, the model parameters, the rotation and the translation aligning the model with the sketch are found by minimizing an objective function which is composed of the residuals between the vertices of the sketch and the 2D projections of the candidate model's vertices, and an optimal reconstruction solution is further selected as the final result. Extensive experimental results demonstrate the effectiveness and robustness of our method for recovering the solid geometric object from single line drawing image.

Keywords Line drawing · 3D reconstruction · Geometric object

✉ Yongtao Wang
wyt@pku.edu.cn

[1] Institute of Computer Science and Technology, Peking University, No.5 Yiheyuan Road, Haidian District, Beijing, 100871, China

1 Introduction

In the current geometry lectures, especially in primary and secondary schools, the educational materials are usually limited to traditional paper books or drawing on the black

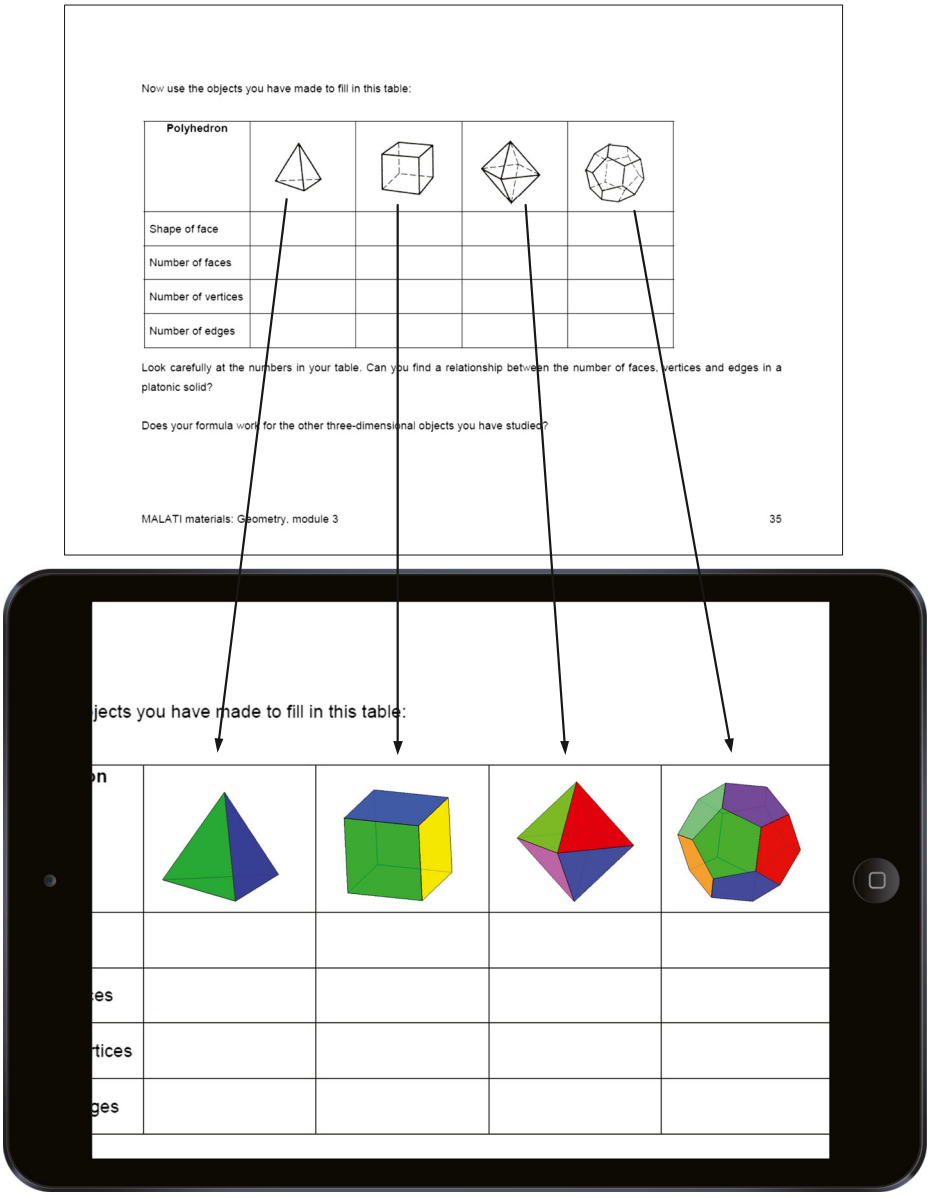


Fig. 1 Illustration of the application of solid geometric object reconstruction technology to mobile reading. In this paper, the handled solid geometric objects are the primitive objects, as listed in the figure from left to right, they are the tetrahedron, the cube, the octahedron and the dodecahedron respectively. The primitive solid geometric objects are represented by undirected graphs and pre-defined in a 3D model database

boards, which is far from adequate for the needs of the students and the teachers. With the help of the rapid progress of the information technology, it's now possible for the readers to learn the books on their mobile devices (e.g. smart phones and tablets) by the electronic documents (e.g. PDF documents). However, because of the loss of three-dimensional (3D) structures, sometimes it's too difficult to quickly understand the geometric objects in the two-dimensional (2D) paper books. Even the electronic documents do not solve the problem as they are still in 2D. The 2D geometric objects in the documents are usually drawn as 2D line drawings (i.e., line segments, elliptic arcs and their intersections). They are actually the parallel projections of the corresponding 3D geometric objects whose 3D structures are lost in the 2D projection. If we can recover the lost 3D information of the geometric objects in the electronic documents from their 2D information, we can present the illustrations of them in the 3D style on the mobile devices, and thereby can significantly improve the users' reading and learning experience, especially for the students and teachers.

In this paper, we propose an algorithm to recover the 3D information of the solid geometric objects from single line drawing images taken from the electronic documents. Over decades, a number of methods have been proposed to reconstruct the 3D geometric objects from single line drawings. The mentioned methods assume that the input is the perfect sketch of the line drawings, that is, all the line segments and their intersections are correctly obtained. These methods are not capable of correctly reconstruct the solid geometric objects from the inaccurate line drawings. The main advantage of our method over the existing methods is that it can reconstruct the inaccurately extracted sketches (i.e. over-complete or under-complete sketches) of the line drawings, while performing equally good over the accurate sketches compared to the existing methods. The proposed algorithm is an extended version of our conference article [24]. Compared to [24], this paper provides more details of the algorithm. Moreover, the graph similarity-based searching technique is employed to speed up the process of the candidate 3D models selection. And in addition, more extensive experiments are conducted to evaluate the performance of the refined algorithm.

Based on the proposed algorithm, we implement a mobile application as illustrated in Fig. 1 which enables the user to select the geometric object from the PDF document, then the application instantly reconstructs the geometric object in the selected image, and renders the reconstructed 3D geometric object onto the screen. The users can interact with the 3D object using their fingers including dragging and rotating, which is essential to improve the user experience in reading such educational materials.

The rest of this paper is organized as follows. The related work is briefly reviewed in Section 2. An overview and some assumptions of our method are provided in Section 3. Section 4 mainly discusses the sketch extraction process in our method. Section 5 describes the 3D model matching process. Section 6 presents the 3D reconstruction algorithm. Experimental results are provided in Section 7 and conclusions are drawn in Section 8.

2 Related work

In the past two decades, a lot of researchers made efforts to resolve the single line drawing-based 3D reconstruction problem. These methods can be roughly categorized into 3 types: the regularity-based methods, the deduction-based methods and the divide-and-conquer-based methods.

Regularity-based methods use some geometric rules as constraints to construct a cost function, and then minimize this function to obtain the 3D object. Conventional rules

include: (1) the face planarity rule: the coplanar vertices of the line drawing should also be coplanar 3D points after reconstruction [6, 13–15, 19]; (2) angularity rule: all the angles at the vertices of a line drawing should be the same [1, 16, 18]. Besides the preceding two rules, Lipson and Shpitalni [9] propose another 10 rules, such as line parallelism, line verticality, isometry and corner orthogonality, et al. Since the dimension of the search space according to this type of methods is very high, some works [11, 20] try to reduce the dimension of the search space to improve the computational efficiency of these methods.

Deduction-based methods usually make stronger assumptions over the 3D objects corresponding to the input line drawings, e.g., the 3D object has cubic corners [7, 8], or a symmetric plane exists in the 3D object [4], and so on. Based on these assumptions, the reconstruction result is obtained by a deduction process.

The third type of methods adopt divide-and-conquer strategy to reconstruct the complex line drawings [2, 12, 21–23, 26, 27]. These methods split the line drawing to a set of simpler parts. In particular, the traditional regularity-based methods are often used to reconstruct each part. Among these methods, Xue et al. [22] propose a refined divide-and-conquer-based method, in which an example-based approach is used to reconstruct each part of the complex line drawing.

Given the perfect sketch of the line drawings as the input, the above methods achieve good reconstruction results. However, none of them demonstrate their abilities to handle inaccurate sketches. In this work, we present a more robust method to solve the single line drawing-based 3D reconstruction problem, which can handle inaccurate sketches. It worths noting that our method is example-based, the same as the Xue et al. [22]'s method (E3D). The main differences between our method and E3D are twofold: 1. The E3D method can only take the sketch of a line drawing as input, while our method directly takes line drawing image as input; 2. The E3D method needs complete input sketches without missing or erroneous edges, while our method does not need that. Although the E3D method can handle more complex objects based on a divide-and-conquer approach, it requires the input to be a complete sketch of the line drawing. Actually, all the experimental data shown in E3D is man-made sketches created from a CAD software, while our input data is the geometric images from the PDF documents. Experimental results demonstrate that our method is able to produce good results for inaccurate sketches automatically extracted from the input line drawing images.

3 Overview

In this paper, the input image is assumed to be the synthetic geometric line drawing image. Parallel projection is used for the geometric object in the image. Visible lines of the object are drawn as solid lines in the image, and hidden lines of the object are drawn as dashed lines. The object may be coupled with some descriptive letters or short texts around it. Some extra lines that are not part of the geometric object itself may be drawn on the surface of it or inside it. It is worth noting that, in this paper, we only consider the line drawings solely consisting of straight line segments, since most of the illustrations fall into this category. The line drawings containing arcs will be considered in our future work.

As illustrated in Fig. 2, our method consists of three main steps. First, we extract the lines from the image and convert them to an undirected connected graph, which is called the sketch. Note that due to the limitation of the sketch extraction algorithm, the extracted sketch is highly possible to be inaccurate or even incomplete. Second, the extracted sketch

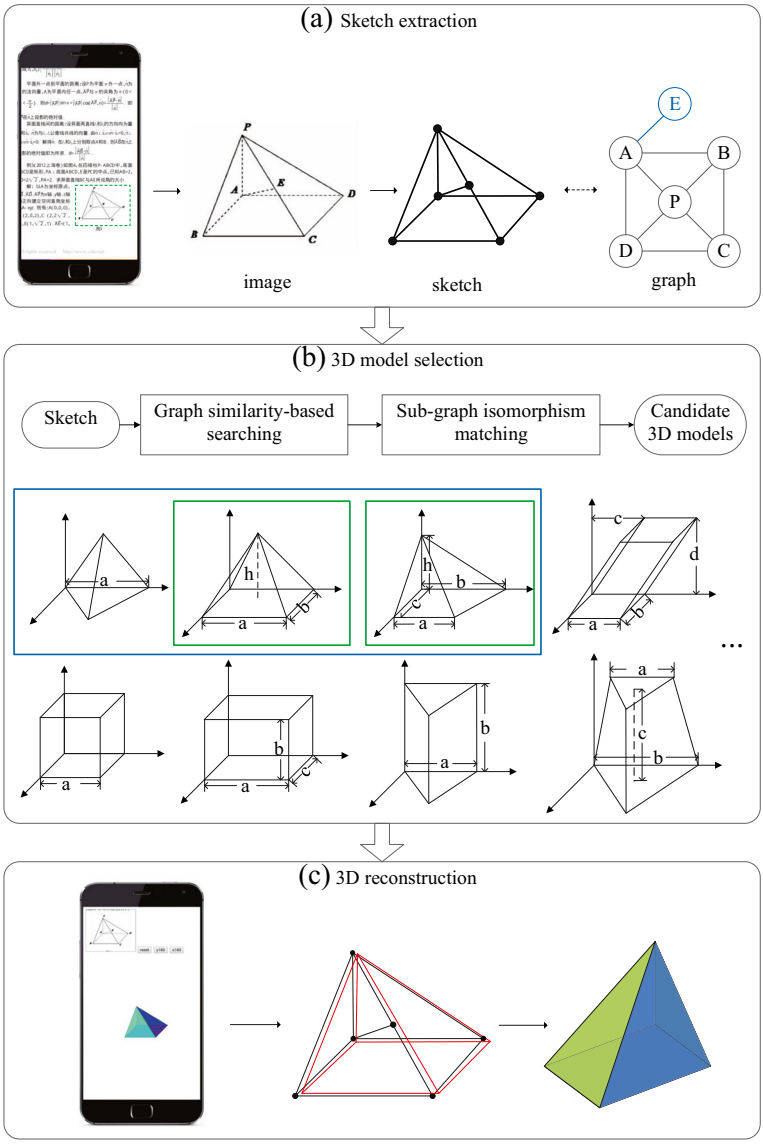


Fig. 2 A brief illustration of our method. The method consists of three main steps: **a** Sketch extraction: the image is cropped from a PDF document using the phone app. The sketch is extracted by detecting the line segments and intersections from the input line drawing image, and further represented by an undirected connected graph. **b** 3D model selection: A 3D model database is pre-defined to describe the primitive 3D models. Some candidate 3D models are first filtered by graph similarity searching (marked by *blue rectangle*) and then selected by sub-graph isomorphism (marked by *green rectangles*). **c** 3D reconstruction: the reconstruction result is obtained by fitting the candidate models (*red lines*) to the sketch (*black lines*). The reconstructed solid geometric object (*a pyramid*) is shown on the phone (*on the left*), and from another point of view (*on the right*)

is matched within a pre-built 3D model database, producing some candidate models that are analog to the sketch. Finally, an objective function is constructed based on the coordinate residuals between the graph and the candidate models, and then an optimal reconstruction solution is found based on an optimization selection process.

In order to clearly present our solution to the single line drawing-base 3D reconstruction problem, we first fix the related notations and definitions.

Definition 1 A **2D sketch** is the graph representation of the line drawing image, denoted by $S = (\mathbf{x}, G_s)$, where $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$ are the 2D coordinates of the vertices, and G_s is the undirected connected graph indicating which two vertices are connected.

Definition 2 A **3D object** is represented as an undirected connected graph in 3D space, that is, $O = (\mathbf{X}, G_o)$, where $\mathbf{X} = \{X_1, X_2, \ldots, X_m\}$ are the 3D coordinates of the vertices of the object, and G_o is the undirected connected graph indicating which two vertices are connected.

Definition 3 A **3D model** represents a kind of 3D object controlled by a set of model parameters, which is denoted by $M = (\mathbf{A}, \mathbf{X}, G_m)$, where $\mathbf{A}$ is the set of parameters, $\mathbf{X}$ is the set of vertices, and G_m is the undirected connected graph indicating which two vertices are connected.

Definition 4 An **instance of the 3D model** is an 3D object that is generated by the rotation and translation of the 3D model in 3D space.

4 Sketch extraction

The sketch extraction algorithm we use is listed in Fig. 3. The input gray-scale image (the color image is first converted to the gray-scale one) is scaled to a normal size(the longer edge is sized to 400 pixels and the shorter edge is sized into corresponding proportion), and then binarized with the OTSU method [17], and then connected components of the foreground pixels (the black pixels of the original input image) are obtained.

4.1 Connected component clustering

As shown in Fig. 4a, there are usually 3 types of the connected components according to their sizes: the main body (the one bounded in blue rectangle), the dashed line dots (the ones in red rectangles), and the characters (the ones in green rectangles). Based on these observations, We adopt the k-means [5] clustering (we set k = 3 straightforwardly) algorithm to classify the types of the connected components. The count of the foreground pixels, the height and width of the bounding box are taken as the feature for clustering. After clustering, we divide the connected components into 3 types: the one whose bounding box is significantly larger than the others is selected as the main body; the ones with smallest bounding boxes and similar foreground pixel counts are classified to the dashed line dots; the remaining ones are then classified to the characters, which are not involved in our further processing. This strategy is simple yet effective to provide an estimation of the component types.

4.2 Line extraction

We use the edge segment-based [10] to extract the solid lines (Fig. 4b), and propose a sample-consensus-based algorithm (summarized in Algorithm 1) for dashed line extraction. The dashed dots are first shrink to their center points as the input of the algorithm. The dashed line extraction process is illustrated in Fig. 5. Figure 5a shows all the input center points: the inliers are marked in green color and the outliers are in red. Then two lines are sequentially extracted by the sample-consensus process in Fig. 5b and c, after which only outlier points are left in Fig. 5d. This algorithm is efficient and robust in presence of a few outliers. Figure 4c shows an example of the extracted dashed lines from the line drawing image.

Algorithm 1 Dashed line extraction

Input:
A set of central points $\mathbf{V} = \{v_1, v_2, \ldots, v_n\}$;
Output:
A set of lines $\mathbf{L}$;
1: $\mathbf{L} = \phi, \mathbf{G} = \phi$;
2: For each pair of points $v_i, v_j \in \mathbf{V}$, calculate the line l_{ij} that goes through them, and find the inlier points set $\mathbf{P}_{ij}$. Add $\mathbf{P}_{ij}$ to $\mathbf{G}$;
3: **while** $\mathbf{G}$ is not empty **do**
4: $\mathbf{P} := \arg\max |\mathbf{P}_{ij}|$, $i, j \in [1, n]$. If $|\mathbf{P}| < 3$ output $\mathbf{L}$ and exit.
5: Calculate the line l by least-squares fitting of the points in $\mathbf{P}$.
6: Add l to set $\mathbf{L}$. Remove $\mathbf{P}$ from $\mathbf{G}$. Remove all the points in $\mathbf{P}$ from the set $\mathbf{V}$, and from all existing $\mathbf{P}_{ij}$ sets in $\mathbf{G}$.
7: **end while**
8: Output $\mathbf{L}$ and stop.

4.3 Generating the sketch

After the solid and dashed lines are extracted from the input image, we further convert them into an undirected connected graph. First, the intersection points of the lines are determined, and the lines are cut into line segments according to these intersection points. Second, an image-based validation process is performed to remove the false segments that only cover very few foreground pixels. Third, the line segments that are adjacent and collinear are merged to one segment. Finally, the vertices of the sketch graph are obtained by merging the end points that are very close to each other, and the edges of the sketch graph are obtained according to the line segments. As shown in Fig. 4d, the graph vertices are circled and labeled by Arabic numbers, and the graph edges are drawn in different colors.

4.4 Unnecessary edges removal

Some of the extracted line segments are correct ones, but not useful or even harmful to the following reconstruction process. In order to successfully reconstruct the geometric object

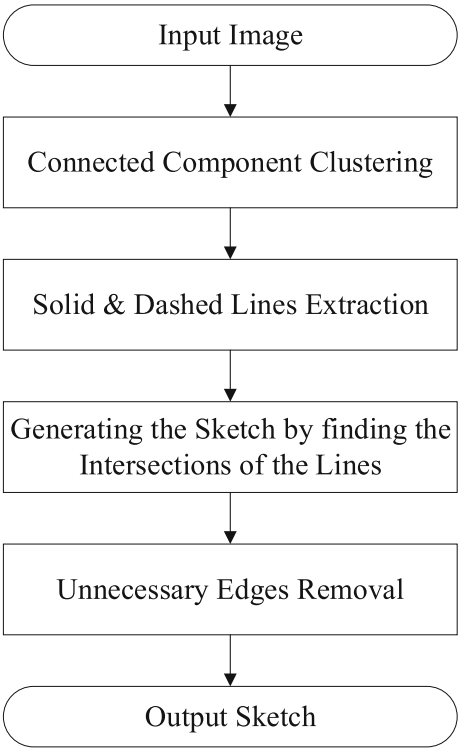


Fig. 3 Flowchart of the sketch extraction algorithm

in the line drawing image, we use some heuristics to remove some of the vertices and edges from the obtained sketch graph as follows.

Type 1: Dangling edges. In the extracted sketch graph, a vertex point whose degree equals to 1 is called a dangling point, and the corresponding edge is called the dangling edge. For example, the edge $2-7$ in Fig. 6a is a dangling one.

Type 2: Docking edges. We call a line segment as "docking line segment" if one of its end points is at the middle of another line segment. And the corresponding edge is a docking

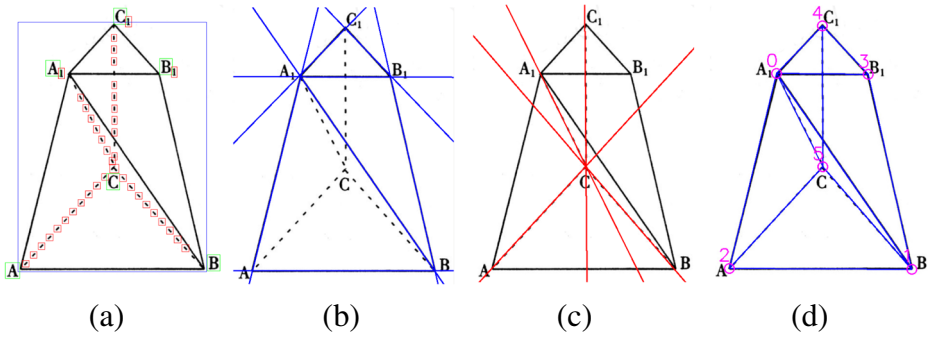


Fig. 4 Illustration of sketch extraction. **a** connected component clustering. **b** solid line extraction. **c** dashed line extraction. **d** generating the sketch (undirected connected graph)

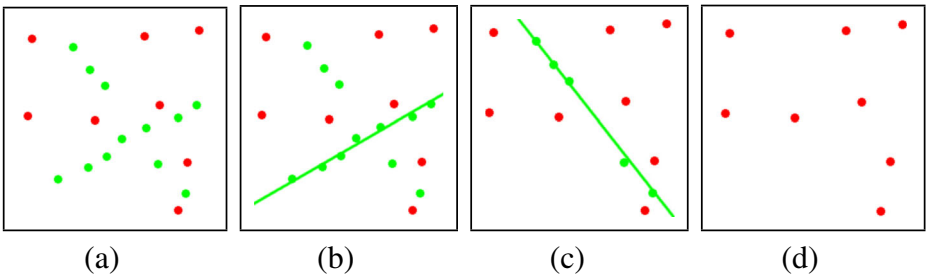


Fig. 5 The consensus-sampling extraction process (**a**) the inliers and the outliers. **b**, **c** lines extracted by consensus-sampling of the inliers. **d** left outliers

edge. For example, in Fig. 6b, the line segments CE, CF, C_1E, C_1F and EF are docking ones.

Type 3: Diagonal edges. The "diagonal line segment" is the one that connects the diagonal points of a parallelogram in the sketch. And the corresponding edge is a diagonal edge. As shown in Fig. 6c, the line segments A_1B, BC_1, and A_1C_1 are diagonal ones.

Given a sketch graph G_s, we detect the unnecessary edges from it and remove them from G_s, obtaining a refined version of G_s, which is denoted as G_r. The full sketch G_s and the refined sketch G_r are both used in the 3D model selection process which is described in the next section.

5 3D model selection

After obtaining the sketch graph of the line drawing, we select some candidate 3D models with the similar graph structure from the pre-built 3D model database. The flow of the

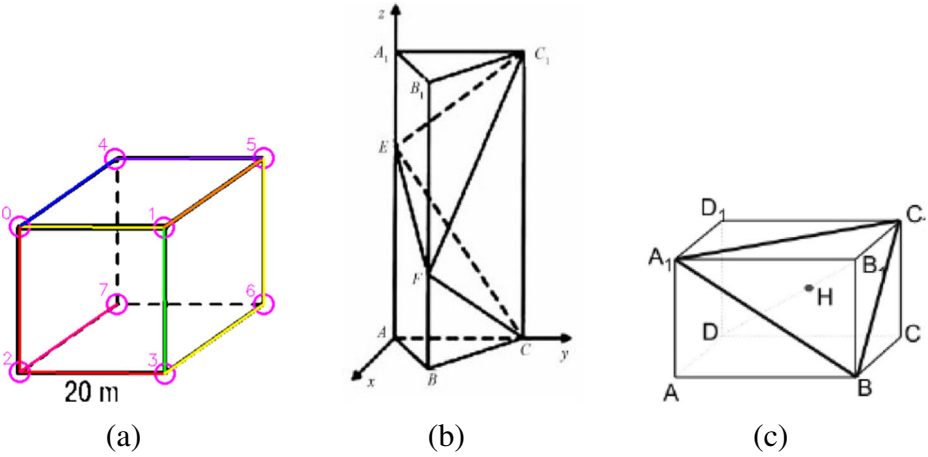


Fig. 6 3 types of unnecessary edges. **a** type 1. **b** type 2. **c** type 3

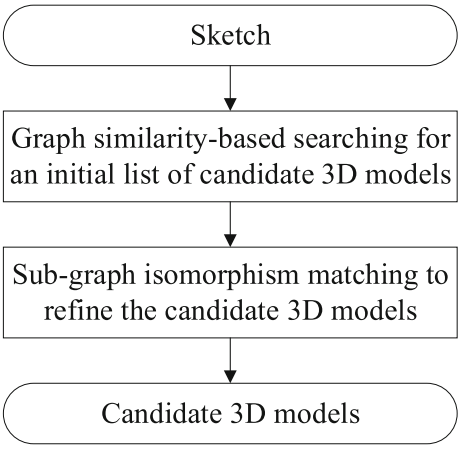


Fig. 7 Flowchart of the candidate 3D models selection process

selection process is listed in Fig. 7. The details will be introduced in the following subsections.

5.1 3D model database

In this work, a 3D model is represented as an undirected connected graph in the 3D space. Each vertex of the model has a 3D coordinate X_i, and the graph is represented by G_m.

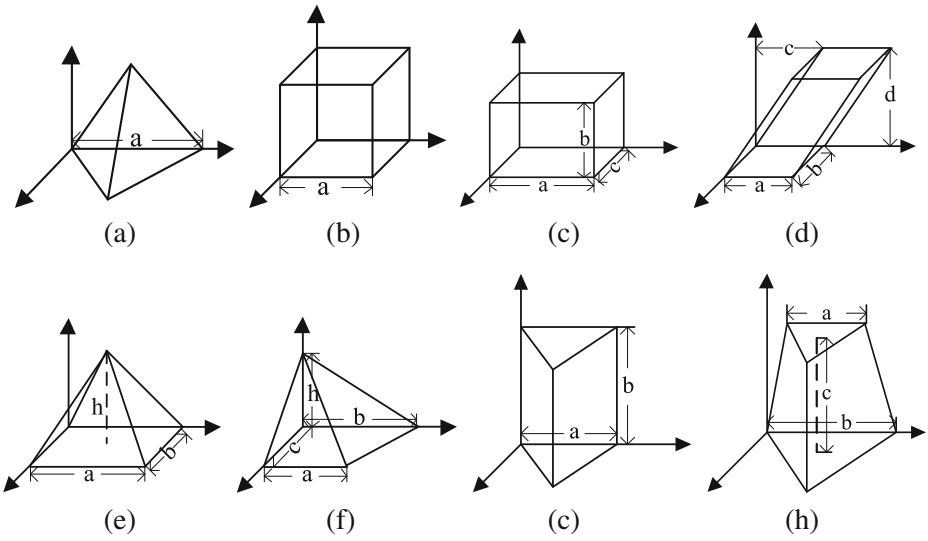


Fig. 8 Examples of the 3D models. **a** tetrahedron. **b** cube. **c** cuboid. **d** oblique cuboid. **e** pyramid. **f** irregular pyramid. **g** triangle-prism. **h** tetrahedron-frustum

Moreover, a 3D model is controlled by a number of model parameters (denoted as set **A**), which represent the geometric attributes of the 3D model, such as width, height, depth. Some of the examples are shown in Fig. 8. Figure 8c is a cuboid model. It has 3 parameters: $\mathbf{A}_{cuboid} = \{a, b, c\}$, where a, b, c are the width, height and length of the cuboid model respectively. For a 3D model M, we use a matrix which is composed by the elements of **A** to represent the 3D coordinates of the vertices, which is called the parametric matrix of the model and is denoted by V_M. For example, the parametric matrix of the cuboid is

$$V_{cuboid} = \begin{pmatrix} 0 & a & a & 0 & 0 & a & a & 0 \\ 0 & 0 & 0 & 0 & b & b & b & b \\ 0 & 0 & c & c & 0 & 0 & c & c \end{pmatrix}, \tag{1}$$

in which each column V_i is the 3D Euclidean coordinates of a vertex. Obviously, the parametric matrix always has 3 rows and the number of columns equals to the number of vertices.

We have totally defined 19 models to build the 3D model database, which can cover almost all the cases that appear in our experimental data set.

5.2 Candidate models searching

We select the candidate models from the 3D model database based on two techniques: the graph similarity searching combined with the additional sub-graph isomorphism step. The graph similarity searching is used to filter the models that are unlikely to be the candidate model. And The sub-graph isomorphism is used to find each candidate model that matches the sketch we extracted from the image.

5.2.1 Graph similarity-based searching

The first step is the graph similarity searching. The idea is to find some models that are similar to the sketch before we do the sub-graph isomorphism matching over the models in the database. The minimum graph edit distance (MGED) is used to estimate the graph similarity in our approach. An efficient graph similarity searching algorithm [25] is used to filter the models similar to the sketch from the model database. Specifically, we input the sketch as the query graph q to the algorithm in [25], and the model database as the graph database D, and a threshold τ where the MGED is allowed maximumly (we set τ to 10 in this paper).

After the graph similarity searching, the algorithm returns a list of graphs whose MGED is not greater than τ. We sort the list of graphs according to the MGEDs of them. From the list, we select the top T (we study this parameter in the experiments section) graphs with the smallest MGEDs, and input the selected graphs *SIM* to Algorithm 2, which searches the candidate models and the corresponding sub-graph configurations (i.e., the correspondences of the vertices and the edges of the isomorphism projection) of G_s. We first find the possible matching sub-graphs in *SIM* (lines 9–11 in Algorithm 2). However, if unfortunately we didn't find anything good in *SIM*, we fall back to a full scan over the whole model database (lines 12–16 in Algorithm 2).

Algorithm 2 SearchCandidateModels(G_s, *SIM*, **M**)

Input:
G_s, the extracted sketch graph;
SIM, the list of model graphs after similarity searching;
M, the full model database;
Output:
CAND, the list of the candidate models;
SUB, the sub-graph configurations of G_s corresponding to the candidate models;

```
1: procedure SearchSubIso(G_s, G_m)
2: SUB_s ← SubIsoMatch(G_s, G_m)
3: if |SUB_s| > 0 then
4:    CAND ← CAND + G_m
5:    SUB ← SUB ∪ SUB_s
6: end if
7: end procedure
8:
9: for each graph G_m ∈ SIM do
10:   SearchSubIso(G_s, G_m)
11: end for
12: if |CAND| = 0 then
13:   for each graph G_m ∈ M do
14:      SearchSubIso(G_s, G_m)
15:   end for
16: end if
17: return CAND, SUB
```

5.2.2 *Sub-graph isomorphism matching*

The sub-graph isomorphism is a graph matching technique to find a sub-graph in a given bigger graph G isomorphic to a given smaller graph H, (if there exists a graph G' that is a sub-graph of G, and H is isomorphic to G', we say that H is sub-graph isomorphic to G). We adopt the VF-2 [3] algorithm to accomplish our sub-graph isomorphism task. The reason why we choose this method is that it's one of the most efficient sub-graph isomorphism matching algorithms so far. For more details about sub-graph isomorphism, please refer to [3].

As listed in Algorithm 3, we perform the sub-graph isomorphism twice. In the first time, we take the sketch graph G_s as the bigger graph and the model graph G_m as the smaller graph; and in the second time, we conversely take G_m, G_r (the refined version of G_s) as the bigger graph and the smaller graph respectively.

The first time of sub-graph isomorphism is performed to handle the completely or over-completely extracted sketches. For example, Fig. 4d illustrates the extracted sketch from a line drawing image of tetrahedron-frustum, in which the lines $\mathtt{A_1B}$ and $\mathtt{A_1C}$ are not contained in the model of the tetrahedron-frustum. We find the models whose number of vertices are equal to or smaller than that of the sketch, and are isomorphic to a sub-graph of the sketch. And among these models, only the ones with the largest number of vertices are selected as the candidate models. For the example shown in Fig. 4d, we first find 4 models – pyramid, tetrahedron, triangle-prism and tetrahedron-frustum, and finally select two models

– the triangle-prism and the tetrahedron-frustum as the candidate models, which are shown in Fig. 8g and h.

During the process of the sketch extraction, some line segments may be not extracted or partially extracted, and thus the obtained sketch graph is usually incomplete. In order to deal with these cases, we perform the second time of sub-graph isomorphism—fix the 3D model G_m as the bigger graph, and find sub-graphs that are isomorphic to the refined sketch graph G_r in it. For example, Fig. 6a shows an incomplete sketch extracted from a line drawing image of the cuboid. To be more specific, the two dashed lines ($4-7$ and $6-7$) are completely missing. By applying the second time of sub-graph isomorphism, the correct candidate models (the cube and the cuboid et al.) can still be selected.

Algorithm 3 SubIsoMatch(G_s, G_m)

Input:
The extracted sketch graph G_s;
the 3D model G_m to be matched;
Output:
$SUB_s = \{G_{sub_i}\}$, where G_{sub_i} is a graph that is sub-graph isomorphic to G_s;
1: fix G_s as the bigger graph and G_m as the smaller graph, then invoke VF-2[3] to find the sub-graphs $\{G_{sub}\}$ of G_s
2: $SUB_s \leftarrow SUB_s \cup \{G_{sub}\}$
3: fix G_m as the bigger graph and the refined sketch G_r as the smaller graph, then invoke VF-2 to test whether G_r is sub-graph isomorphic to G_m
4: **if** the isomorphism test is successful **then**
5: $SUB_s \leftarrow SUB_s + G_r$
6: **end if**
7: **return** SUB_s

The sub-graph isomorphism matching process is rather expensive in terms of efficiency due to its high complexity [3], especially when we want to do a full scan over all the models in the database. In order to speed up the candidate models searching process, we use the graph similarity searching prior to the sub-graph isomorphism to prune some models that are unlikely to be the candidate model. The models selected by the two times of sub-graph isomorphism are all added to the list of candidate models, which will be used in the 3D reconstruction process.

6 3D reconstruction

For each candidate 3D model, the reconstruction result is obtained by minimizing an objective function of the residuals between the vertices of the sketch and the 2D projections of the candidate model's vertices. And then, some bad reconstruction results are rejected. Finally, the optimal result that best fits the sketch is selected. The details of our reconstruction algorithm are introduced in the following sub-sections.

6.1 Recovery of each candidate model

After a sub-graph isomorphism, we obtain some correspondence $\mathcal{C} = (\mathbf{X}^{\mathcal{C}}, \mathbf{x}^{\mathcal{C}})$ between the subsets of the vertices of the candidate model and the sketch graph, where $\mathbf{X}^{\mathcal{C}}$ and $\mathbf{x}^{\mathcal{C}}$ are subsets of $\mathbf{X}$ and $\mathbf{x}$ respectively, and $\mathcal{C}$ is a one-to-one correspondence relationship between

$\mathbf{X}^{\mathcal{C}}$ and $\mathbf{x}^{\mathcal{C}}$. The coordinates of the subset $\mathbf{X}^{\mathcal{C}}$ are $\mathbf{X}^{\mathcal{C}} = \{X_{i_1}, X_{i_2}, \ldots, X_{i_{\mathcal{C}}}\}$, where i_k is the sub-indices of $\mathbf{X}$, and the coordinates of the subset $\mathbf{x}^{\mathcal{C}}$ are $\mathbf{x}^{\mathcal{C}} = \{x_{j_1}, x_{j_2}, \ldots, x_{j_{\mathcal{C}}}\}$, where j_k is the sub-indices of $\mathbf{x}$.

For each candidate model, the 3D reconstruction process of our method aims to find an instance of it whose 2D projection can best fit the sketch. To be more specific, our target is to optimize an objective function that minimizes the coordinate residuals between the matched vertices of the sketch and the candidate model. The objective function is the *projection error*, which is given by

$$f = \sum_{k=1}^{n_{\mathcal{C}}} \left\| K(RV_{i_k} + \mathbf{t}) - x_{j_k} \right\|^2, \tag{2}$$

where $n_{\mathcal{C}}$ is the number of corresponding pairs of the vertices; $K = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \end{pmatrix}$ is the parallel projection matrix, R is the rotation matrix, $\mathbf{t}$ is the translation vector, i_k and j_k are the indices of the corresponding vertices of the sketch and the model, V_{i_k} is the i_k-th column of the parametric matrix V, and x_{j_k} is the 2D coordinate of the j_k-th vertex of the sketch. The optimal solution is found by solving the following problem

$$\begin{gathered} \tilde{\mathbf{A}}, \tilde{R}, \tilde{\mathbf{t}} = \arg\min \sum_{k=1}^{n_{\mathcal{C}}} \left\| K(RV_{i_k} + \mathbf{t}) - x_{j_k} \right\|^2 \\ \text{subject to}: R^T R = I, \end{gathered} \tag{3}$$

where $\tilde{\mathbf{A}}$ is the optimal geometric parameters of the model, $\tilde{R}$ is the optimal rotation matrix, and $\tilde{\mathbf{t}}$ is the optimal translation vector.

The objective function is a quadratic function with an orthogonal constraint. We use the algorithm in [22] to solve the problem in (3).

6.2 Optimal reconstruction result selection

For each candidate model, we obtain an reconstruction result by solving the objective function in (3). We further select the best result among these reconstruction results. The selection process is introduced as follows.

First, the result with large projection error should be rejected, that is, if the projection error $f > \delta$, the result should be directly rejected, where δ is an empirically set threshold (in this work, we set δ to 90.0).

Second, for the candidate models selected by the second time of sub-graph isomorphism, there exists some missing vertices and missing lines. For example, in Fig. 9a (the vertical line inside the tetrahedron has been removed as it's a dangling line), the dashed line of the tetrahedron is missing due to failure of the sketch extraction step. Two candidate models are selected by the second time of sub-graph isomorphism: the tetrahedron and the pyramid, as shown in Fig. 9b and c respectively. The tetrahedron model's missing line accurately overlaps with the dashed line in the image, while the pyramid model's missing lines do not hit any line in the image. Although the projection errors of the two models are exactly the same, we can select a better model by the "pixel validation": for each candidate model's reconstruction result, the "pixel vote"—the count of the foreground pixels that the missing lines of the reconstructed instance of the model pass through is collected. We look for the pixels on both sides of the missing lines at a small range, for tolerance of some coordinate inaccuracies. If a candidate model's pixel vote is significantly smaller than the others, it should be rejected.

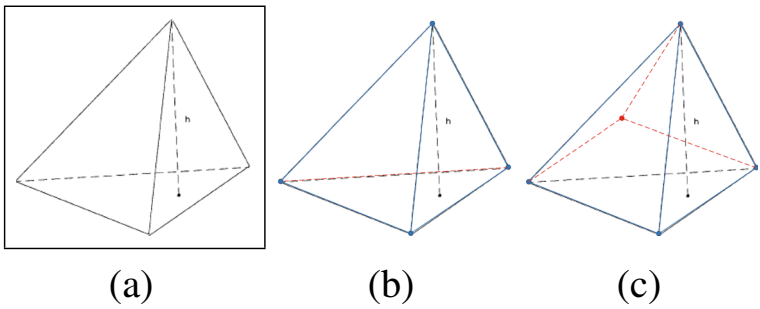


Fig. 9 Example of pixel level validation. **a** a line drawing image contains a tetrahedron. **b** tetrahedron model with a missing line (red dashed line). **c** pyramid model with missing lines

Third, if two candidate models' projection error are nearly the same, we reject the one with a larger parameter set **A**, since it's a more complex model than the other one. We want to select the model as simple as possible, in order to solve the over-fit issue.

Finally, if there are still multiple candidate models, we choose the one with the smallest projection error as the final selected model.

The optimal reconstruction algorithm is shown in Algorithm 4.

Algorithm 4 3D reconstruction from extracted sketch

Input:

extracted sketch graph G_s, candidate models $\{M_k\}$;

Output:

the reconstructed 3D object;

1: For each candidate model M_k, solve the minimization problem in (3) by using algorithm in [22]. Let f_k denote the projection error, p_k denotes the pixel vote, and $\mathbf{A}_k$ denotes the parameters set.
2: Find the candidate model with the highest pixel vote. Let $\hat{p}$ denote the highest pixel vote.
3: From the candidate models, remove the ones whose $f_k > \delta$.
4: If one candidate model is generated by the second time of sub-graph isomorphism, and its pixel vote $p_k < 0.1 * \hat{p}$, remove it.
5: Keep the candidate models whose parameters set $\mathbf{A}_k$ has the smallest number of parameters, remove all the others.
6: If there are still more than one candidate models, remove all except the one with the smallest projection error f_k.
7: With the only one left model and it's parameters, Output $RV + \mathbf{t}$ as the final reconstructed 3D object.

7 Experiment

We implement our algorithm in C++, and also develop a graphical application both on PC and mobile phone to demonstrate our method. As illustrated in our demo, with this graphical application, user first loads a PDF file and selects an rectangular area of 3D geometry illustration from some pages; and after the system processing, the reconstruction result is

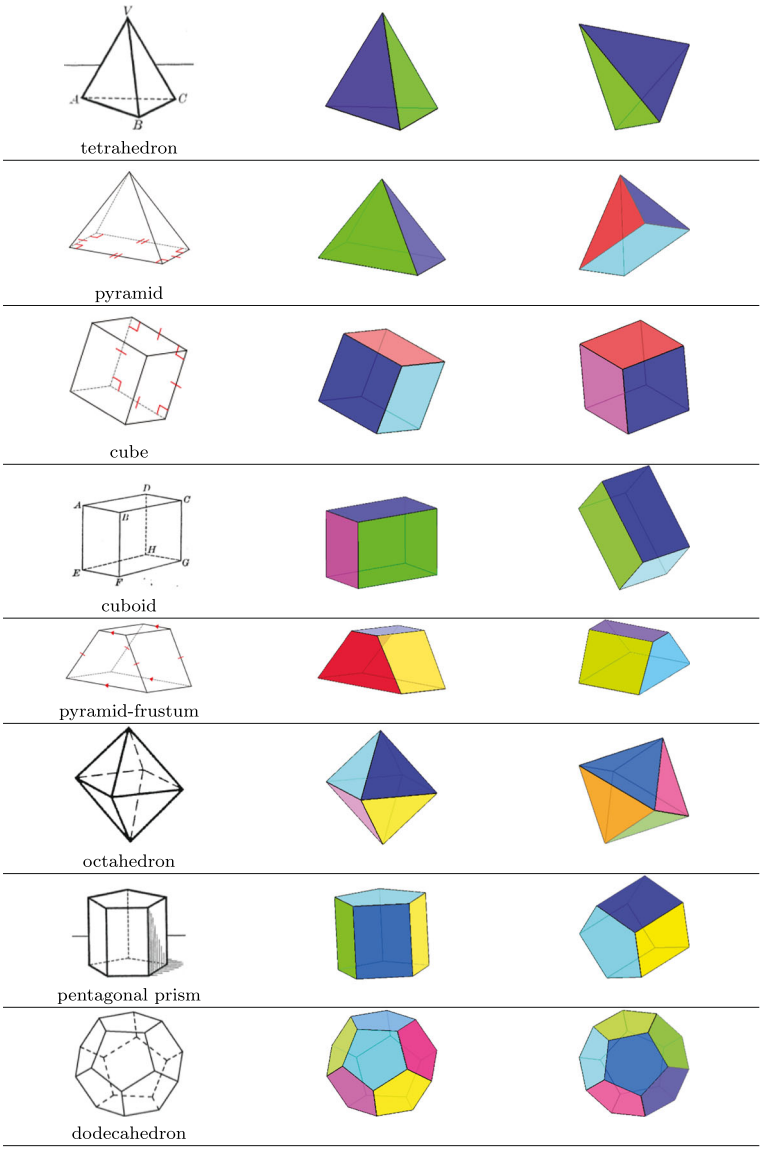


Fig. 10 Examples of reconstruction results. The first column is the input line drawing image and the name of the 3D model. The second column is the reconstructed object from original angle of view. The third column is the object viewed from different angles

shown in a pop up window. User can zoom and drag to rotate the reconstructed object, as if it's immersed in a 3D space. No parameter setting is required for input; and the whole process usually only takes 1–10 s.

We collected over 40 PDF documents from the internet (including books, papers, teaching materials, slide shows and other types of documents), and captured 303 line drawing images from them. Our algorithm is tested over these line drawing images. Some quick

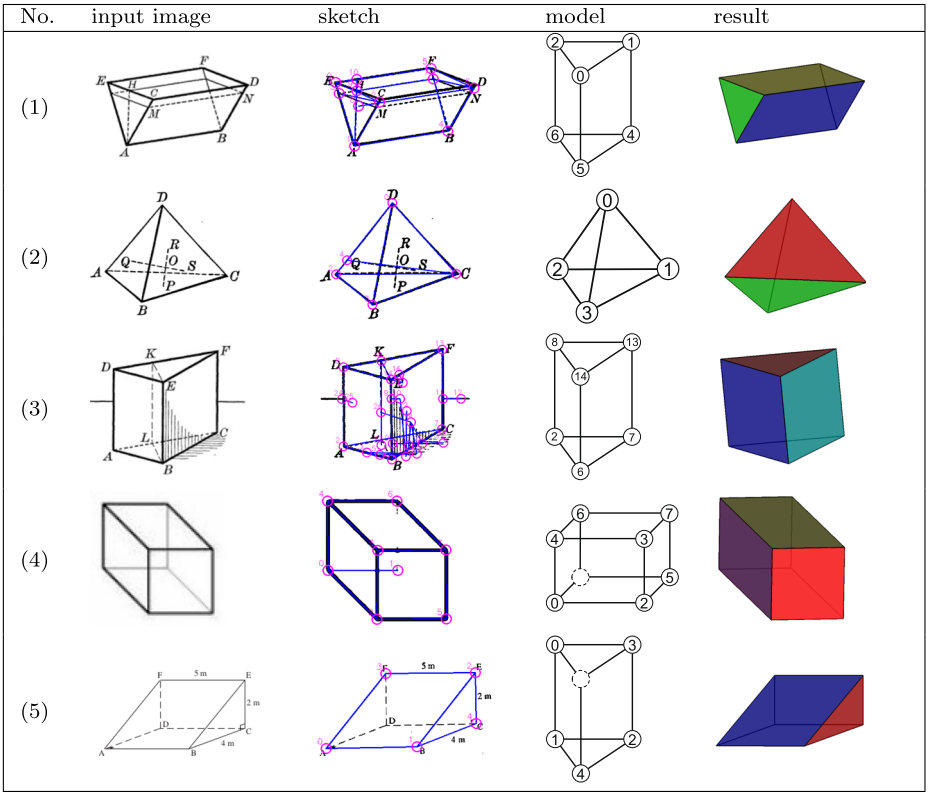
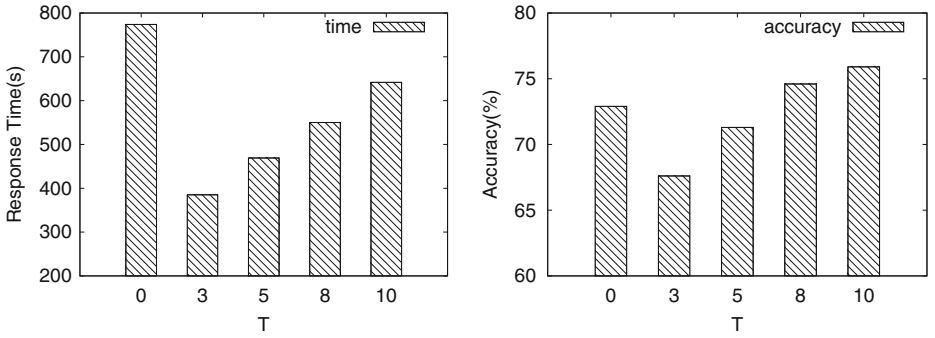


Fig. 11 (*1*)–(*3*): Examples of the over-complete sketch using the first time of sub-graph isomorphism. Correspondences of the vertices are shown in the model with the numbered labels. (*4*)-(*5*): Examples of the under-complete sketch using the second time of sub-graph isomorphism. The resulting correspondence has missed a vertex and its sibling edges. The missing vertex is drawn with dashed circle in the model. In the 3D reconstruction step, the missing vertices are simply excluded from the pose computation of the model. The final result is generated from the pose (rotation and translation) of the model

examples of the reconstruction results are shown in Fig. 10, and some detailed examples of the reconstruction results are shown in Fig. 11.

Evaluation In the most related work of E3D [22], the authors use the RMSA (root mean squares of differences of angles) and the RMSE (root mean squares of differences of Euclidean distances) metrics to evaluate the reconstruction accuracy. However, we found these metrics not suitable for our case. The reasons can be described as below. First, the line drawings used in E3D are manually crafted in a CAD software, so the ground truth is precisely known; while our input data is just the line drawing image. Second, for the applications such as mobile reading and learning, users tend to care about whether the reconstruction result is the right 3D object illustrated in the original line drawing image, rather than caring about how close the reconstructed angles and points are to the original line drawing. For these reasons, we use a simple metric – the matching accuracy to evaluate the proposed method and E3D. Let $\mathbf{F}$ denote the test image set, and $\mathbf{F}_{correct}$ denotes the set of correctly matched images, then the matching accuracy is defined as $f_a = \frac{|\mathbf{F}_{correct}|}{|\mathbf{F}|}$.



(a) Response Time (b) Accuracy

Fig. 12 Performance vs. T

We vary the value of T (the number of the top selected graph similarity searching results) to evaluate the response time and the matching accuracy of our algorithm. When $T = 0$, the graph similarity searching is disabled, we directly do full scan over the whole model database. As shown in Fig. 12, the response time gets declined significantly if we enable the graph similarity searching, but we lose some accuracy if T is too small. However, when $T \geq 8$ the algorithm over performs the full scan regarding both the response time and the accuracy.

Table 1 compares the results of our method and E3D on our testing dataset. As we can see that the matching accuracy of our method ($T = 8$) is significantly higher than that of E3D.

The reason why E3D performs so poorly is that, it can only handle perfect and complete sketches. But in this paper, we need to handle the sketches extracted from the images, which mainly are incomplete or over-complete. To be more specific, only for 42 of the 303 experimental images we can get the perfect and complete sketches. Moreover, for all of the 42 images that we can get right sketches, E3D correctly reconstructs the corresponding solid geometric objects. And as stated in this paper, our method can also handle incomplete or over-complete sketches besides the perfect sketches. Therefore, for the dataset used in this paper, our method performs much better than E3D.

We can also extend our method to the natural images as shown in Fig. 13. Given an input image, we first detect the straight lines in it as in Fig. 13b, and then extract the sketch using the straight lines as in Fig. 13c. Note that in this example, the extracted sketch is incomplete, and the hidden side of the object is invisible. Then the 3D geometric object in the image is reconstructed. We paste the textures in the image onto the faces of the box, as shown in Fig. 13d and e. The whole process is done automatically, as opposed to the real image modeling example shown in E3D. They need the user to manually sketch along the edges of the object, and both the visible and invisible edges must be drawn in the sketch.

Table 1 Comparison between our method and E3D

Method	Correct	Incorrect	Accuracy
E3D	42	261	13.9 %
Ours	226	77	74.6 %

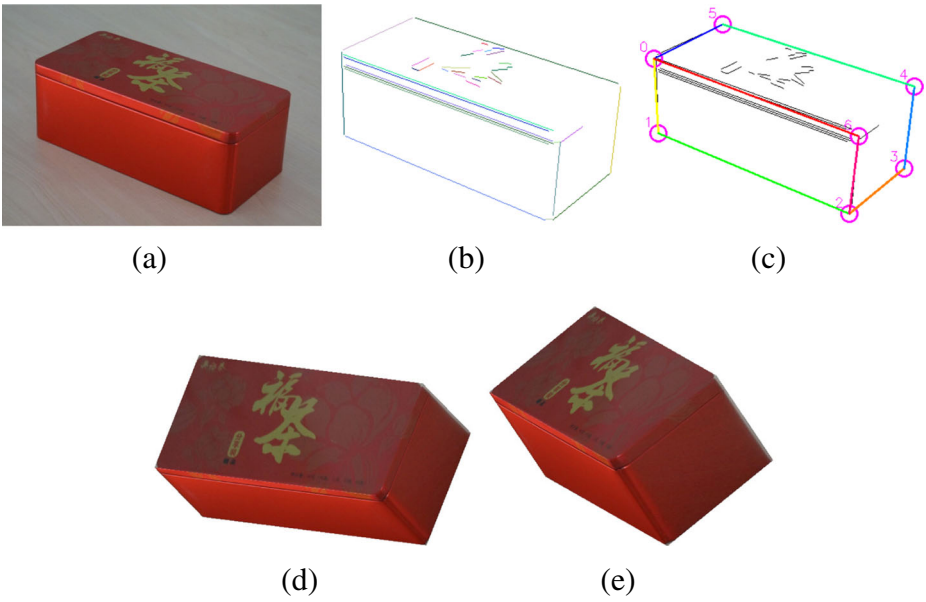


Fig. 13 Example of modeling from natural image. **a** an input image. **b** straight lines detection. **c** sketch extraction. **d**, **e** reconstructed 3D object in two different views

Failure analysis Most failure cases happen when there are both undetected and over-detected edges. For example, the oblique-pyramid in Fig. 14a is mistakenly reconstructed as a tetrahedron in Fig. 14c. From the sketch in Fig. 14b, we can see that the edge CD fails to be detected, and there are some extra edges. Most extra edges like the axis and the edge AF has been removed by the heuristics we use, but the edge AC is not removed since the edge CD is not detected (thus the "diagonal heuristic" does not apply). Finally, we end up getting only wrong candidate models from the first round of sub-graph isomorphism matching, and therefore it's impossible to get the correct reconstruction result.

Other failure cases include over-fitted model being taken, edges being too complicated to be reconstructed, et al. But these are only occasional cases. If we fine tune the algorithm to fit these occasional cases, the whole matching accuracy would decline.

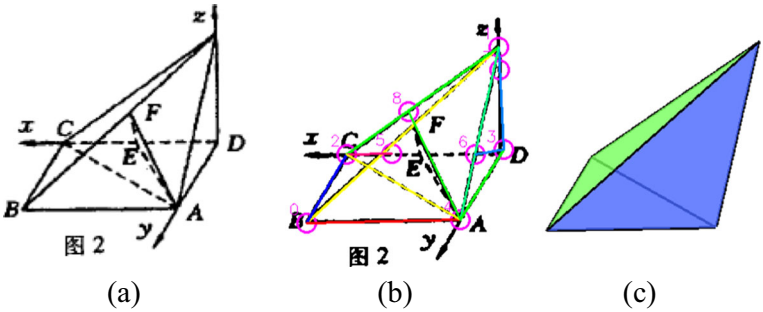


Fig. 14 An example of failure case. **a** input image. **b** extracted sketch. **c** reconstruction result

8 Conclusion and future works

We propose a robust method to recover the 3D information of the solid geometric object from single line drawing image. This is achieved by first extracting the solid and dashed straight line segments from the image and further representing them as an undirected graph, namely, the sketch. Then, candidate models from a pre-built 3D model database are selected by graph similarity-based searching and two times of sub-graph isomorphism matching. Furthermore, for each candidate model, a reconstruction result is obtained by minimizing a cost function of the residuals between the vertices of the sketch and the 2D projections of the corresponding vertices of the candidate model. Finally, some bad reconstruction results are pruned and the optimal result that best fits the sketch is outputted. We conduct the experiments on a set of 303 line drawing images collected from the internet. And the experimental results demonstrate that: (1) it can successfully recover the solid geometric object from single line drawing image; (2) it performs much better in terms of matching accuracy than the state-of-the-art method E3D. We also demonstrate the application of our method to natural image modeling and a failure analysis is provided.

In the future, we plan to do the following works: (1) extracting line styles and labels to preserve the style of the input; (2) recovering the curved objects, such as cylinders and spheres; (3) improving the reconstruction performance by taking context information into account.

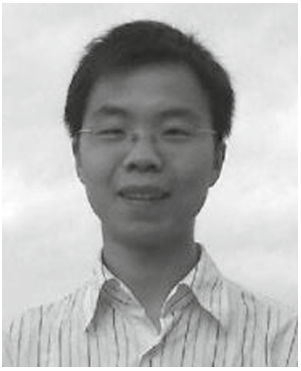
Acknowledgments We'd like to thank all the reviewers for providing encouraging and valuable comments to our work. This work is supported by National Natural Science Foundation of China under Grant 61300061 and Beijing Natural Science Foundation (4132033).

References

1. Brown E, Wang PS (1996) Three-dimensional object recovery from two-dimensional images: a new approach. In: Photonics East'96. International Society for Optics and Photonics, pp 138–147
2. Chen Y, Liu J, Tang X (2007) A divide-and-conquer approach to 3d object reconstruction from line drawings. In: IEEE 11th international conference on computer vision, 2007. ICCV 2007. IEEE, pp 1–8
3. Cordella LP, Foggia P, Sansone C, Vento M (2004) A (sub) graph isomorphism algorithm for matching large graphs. IEEE Trans Pattern Anal Mach Intell 26(10):1367–1372
4. Cordier F, Seo H, Melkemi M, Sapidis NS (2013) Inferring mirror symmetric 3d shapes from sketches. Comput Aided Des 45(2):301–311
5. Hartigan JA, Wong MA (1979) Algorithm as 136: a k-means clustering algorithm. Appl Stat 100–108
6. Leclerc YG, Fischler MA (1992) An optimization-based approach to the interpretation of single line drawings as 3d wire frames. Int J Comput Vis 9(2):113–136
7. Lee YT, Fang F (2011) 3d reconstruction of polyhedral objects from single parallel projections using cubic corner. Comput Aided Des 43(8):1025–1034
8. Lee YT, Fang F (2012) A new hybrid method for 3d object recovery from 2d drawings and its validation against the cubic corner method and the optimisation-based method. Comput Aided Des 44(11):1090–1102
9. Lipson H, Shpitalni M (1996) Optimization-based reconstruction of a 3d object from a single freehand line drawing. Comput Aided Des 28(8):651–663
10. Liu D, Wang Y, Tang Z, Li L, Gao L (2014) Automatic comic page image understanding based on edge segment analysis. In: Proc. SPIE 9021, document recognition and retrieval XXI, pp 90,210J–90,210J–12
11. Liu J, Cao L, Li Z, Tang X (2008) Plane-based optimization for 3d object reconstruction from single line drawings. IEEE Trans Pattern Anal Mach Intell 30(2):315–327

12. Liu J, Chen Y, Tang X (2011) Decomposition of complex line drawings with hidden lines for 3d planar-faced manifold object reconstruction. IEEE Trans Pattern Anal Mach Intell 33(1):3–15
13. Liu J, Lee YT (2001) Graph-based method for face identification from a single 2d line drawing. IEEE Trans Pattern Anal Mach Intell 23(10):1106–1119
14. Liu J, Lee YT, Cham WK (2002) Identifying faces in a 2d line drawing representing a manifold object. IEEE Trans Pattern Anal Mach Intell 24(12):1579–1593
15. Liu J, Tang X (2005) Evolutionary search for faces from line drawings. IEEE Trans Pattern Anal Mach Intell 27(6):861–872
16. Marill T (1991) Emulating the human interpretation of line-drawings as three-dimensional objects. Int J Comput Vis 6(2):147–161
17. Otsu N (1975) A threshold selection method from gray-level histograms. Automatica 11(285–296):23–27
18. Shoji K, Kato K (2001) Toyama, F: 3-d interpretation of single line drawings based on entropy minimization principle. In: Proceedings of the 2001 IEEE computer society conference on computer vision and pattern recognition, 2001. CVPR 2001, vol 2. IEEE, pp II–90
19. Shpitalni M, Lipson H (1996) Identification of faces in a 2d line drawing projection of a wireframe object. IEEE Trans Pattern Anal Mach Intell 18(10):1000–1012
20. Tian C, Masry M, Lipson H (2009) Physical sketching: reconstruction and analysis of 3d objects from freehand sketches. Comput Aided Des 41(3):147–158
21. Xue T, Liu J, Tang X (2010) Object cut: complex 3d object reconstruction through line drawing separation. In: 2010 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 1149–1156
22. Xue T, Liu J, Tang X (2012) Example-based 3d object reconstruction from line drawings. In: 2012 IEEE conference on computer vision and pattern recognition (CVPR). IEEE, pp 302–309
23. Yang L, Liu J, Tang X (2013) Complex 3d general object reconstruction from line drawings. In: 2013 IEEE international conference on computer vision (ICCV), pp 1433–1440. doi:10.1109/ICCV.2013.181
24. Zheng J, Wang Y, Tang Z (2015) Solid geometric object reconstruction from single line drawing image. In: International conference on computer graphics theory and applications (GRAPP 2015)
25. Zheng W, Zou L, Lian X, Wang D, Zhao D (2015) Efficient graph similarity search over large graph databases. IEEE Trans Knowl Data Eng 27(4):964–978. doi:10.1109/TKDE.2014.2349924
26. Zou C, Chen S, Fu H, Liu J (2014) Progressive 3d reconstruction of planar-faced manifold objects with drf-based line drawing decomposition. IEEE Trans Vis Comput Graph PP(99):1–1. doi:10.1109/TVCG.2014.2354039
27. Zou C, Yang H, Liu J (2014) Separation of line drawings based on split faces for 3d object reconstruction. In: 2014 IEEE conference on computer vision and pattern recognition (CVPR), pp 692–699. doi:10.1109/CVPR.2014.94

Jinxin Zheng received his M.S degree of applied computer technology in 2010 from Beijing University of Posts and Telecommunications, China. He is currently a Ph.D. student at the Institute of Computer Science & Technology, Peking University, China.



Yongtao Wang received his Ph.D. degree of pattern recognition and intelligent system in 2009 from Huazhong University of Science and Technology (HUST), China. During 2010, he was a research scientist of the Temasek Laboratories, Nanyang Technological University, Singapore. He is currently an assistant professor at the Institute of Computer Science & Technology, Peking University, China. His research interests involve document image analysis and understanding, wide baseline matching, and motion segmentation.

Zhi Tang received his Ph.D. degree of applied computer technology in 1995 from Peking University, China. He is now a professor at the Institute of Computer Science & Technology, Peking University. His research interests majorly include document analysis and understanding and digital rights management.