This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

Progressive 3D Reconstruction of Planar-Faced Manifold Objects with DRF-Based Line Drawing Decomposition

Changqing Zou, *Student Member, IEEE,* Shifeng Chen, *Member, IEEE,* Hongbo Fu, and Jianzhuang Liu, *Senior Member, IEEE*

Abstract—This paper presents an approach for reconstructing polyhedral objects from single-view line drawings. Our approach separates a complex line drawing representing a manifold object into a series of simpler line drawings, based on the degree of reconstruction freedom (DRF). We then progressively reconstruct a complete 3D model from these simpler line drawings. Our experiments show that our decomposition algorithm is able to handle complex drawings which are challenging for the state of the art. The advantages of the presented progressive 3D reconstruction method over the existing reconstruction methods in terms of both robustness and efficiency are also demonstrated.

Index Terms—3D reconstruction, degree of reconstruction freedom, decomposition, line drawing, optimization-based

1 Introduction

A line drawing is the simplest and most direct way of illustrating a 3D object. The human vision system is able to interpret 2D line drawings as 3D objects with no difficulty. Emulating this ability is an important but challenging research topic for machine vision. 3D reconstruction from line drawings benefits various applications such as the design of flexible sketching interfaces for conceptual designers [1]–[6], and sketch-based 3D object retrieval [7]–[9]

Similar to previous 3D reconstruction methods in [3], [10]–[14], we tackle the problem of 3D reconstruction from single line drawings with an optimization-based approach. Optimization-based 3D reconstruction methods determine the depth values (z-coordinates) of the vertices of a line drawing from the solution that optimizes a certain objective function. The main problem with the existing methods is that they become less robust and less efficient when a line drawing is complex.

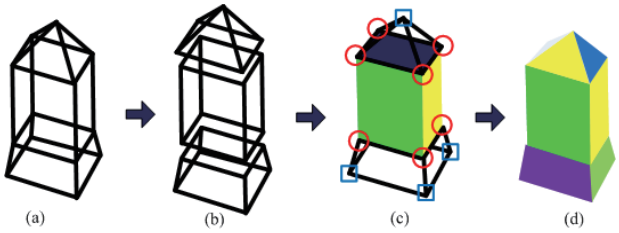


Fig. 1: Our 3D reconstruction pipeline. (a) and (b) show that a line drawing is decomposed into three parts, and (c) and (d) illustrate the progressive reconstruction which sequentially and dependently reconstructs the 3D shapes from the three parts. Note that the good initial depths of the vertices circled by □ in (c) are estimated from the depths of the vertices marked by ◯ of the already reconstructed middle part.

This work focuses on improving the performance of optimization-based 3D reconstruction from complex line drawings. We intend to avoid the search in a high-dimensional space in the optimization process. We present an efficient degree-of-reconstruction-freedom (DRF) based algorithm (Section 4), which decomposes a complex line drawing into a series of simpler parts with small DRFs (Figures 1(a) and (b)). As we will show, compared to the existing line drawing decomposition technique [13], our algorithm is much more efficient and is able to successfully handle more line drawings with high complexity.

- *Changqing Zou is with the Department of Physics and Electronic Information Science, Hengyang Normal University, Hengyang, China, Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China, and also with Shenzhen College of Advanced Technology, University of Chinese Academy of Sciences, China. E-mail: aaronzou1125@gmail.com*
- *Shifeng Chen is with Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Shenzhen, China. E-mail: shifeng.chen@siat.ac.cn*
- *Hongbo Fu is with the City University of Hong Kong. E-mail: hongbofu@cityu.edu.hk*
- *Jianzhuang Liu is with the Media Laboratory, Huawei Technologies Co., Ltd., Shenzhen, China, and also with the Department of Information Engineering, The Chinese University of Hong Kong, Hong Kong. E-mail: liu.jianzhuang@huawei.com*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

In addition, we propose an estimate-and-optimize strategy for progressive reconstruction of the separated simpler parts (Section 5). Our method starts with 3D reconstruction of an initial part (Fig. 1(c)), and sequentially reconstructs other parts (Fig. 1(d)) based on the already reconstructed parts. The latter provides a good initial setting for the reconstruction of subsequent ones, thus greatly improving the robustness of optimization-based 3D reconstruction from line drawings.

2 Related Work

Computational interpretation of line drawings has spanned more than four decades. A full review of this topic is beyond the scope of this paper. See [15]–[17] for insightful surveys. The earliest work towards 3D reconstruction from single line drawings is line labeling, which focuses on finding a set of consistent labels from a line drawing and testing its correctness and realizability [18]–[20]. Line labeling is useful to detect some perceptual relations in a line drawing, but it does not explicitly give the 3D shape represented by the line drawing. The methods based on linear programming [21]–[23] reconstruct a 3D model by solving a linear system which is built from a set of geometrical conditions that the model must fit. In general, linear programming has difficulty tolerating sketching errors that often exist in a line drawing.

Modern methods of 3D reconstruction from line drawings are often optimization based. The seminal work by Marill [14] presented a very simple regularity for 3D reconstruction, i.e., minimizing the standard deviation of the angles (MSDA) in the reconstructed objects so that a 2D line drawing can be inflated into a 3D shape. Motivated by MSDA, Brown and Wang [24] proposed to minimize the standard deviation of the segment magnitudes (MSDSM), and Shoji *et al*. [25] presented the criterion of minimizing the entropy of angle distribution (MEAD). MSDA, MSDSM, and MEAD can only recover simple objects from line drawings. Leclerc and Fischler's method [11] considers not only MSDA but also the planarity constraint on the faces of the object (planarity constraint is a powerful property and many methods have been proposed to find faces from a line drawing [26]–[31]). This method performs better than MSDA, MSDSM, and MEAD. Later, Lipson and Shpitalni [3] extended Leclerc and Fischler's method by using more constraints like line parallelism, line verticality, isometry, etc., enabling the reconstruction of more complex objects than Leclerc and Fischler's. Based on the works [3], [11], [14], Turner *et al*. recovered simple planar 3D objects from scenes [32]. Shesh and Chen applied Lipson and Shpitalni's algorithm to their sketching system in [33]. Liu *et al*. proposed a plane-based optimization method

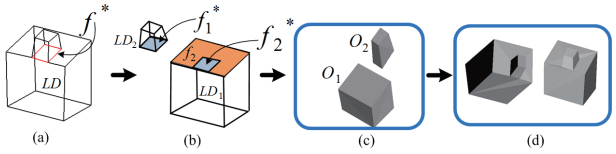


Fig. 2: Illustration of the reconstruction method in [13]
.

in [12] which finds desired objects in a much lower dimensional search space by enforcing a set of linear constraints on the unconstrained optimization problem. Their method can tackle 3D reconstruction of more complex objects, but fail to obtain an expected 3D object when the degree of reconstruction freedom of an object is large [13].

Our work might be the most relevant to that by Liu *et al.* [13], which also presented a line drawing decomposition algorithm. As illustrated in Fig. 2, their algorithm firstly decomposes a line drawing into a set of simpler line drawings (LD_1 and LD_2 in Fig. 2(b)) along internal faces (f^*, marked in red in Fig. 2(a)), then reconstructs 3D shapes (O_1 and O_2 in Fig. 2(c)) from the resulting simpler line drawings independently, and finally merges the 3D shapes together and fine-tunes the merged 3D model. By decomposing a complex line drawing into simpler line drawings, their method avoids the search in a high dimension space.

One of the limitations of the method in [13] is that the decomposition algorithm based on finding internal faces is NP-complete. To make this algorithm run in tolerable time, a predefined maximum search depth has to be set. Therefore, it cannot find an internal face when the number of its edges is larger than the threshold. Another deficiency of [13] is that the 3D shapes are reconstructed independently from the decomposed line drawings, without using the geometrical relationships between the decomposed line drawings. This might cause artifacts in the subsequent merging step, e.g., non-planar faces in the complete model in Fig. 2(d) after merging O_1 and O_2 via f_1^* and f_2^*. This is mainly because while f_1^* and f_2^* are consistent in the decomposed line drawings, the corresponding faces in O_1 and O_2 might become inconsistent due to independent reconstruction. This justified their extra fine-tuning step, which, however, does not always work well and is time consuming.

Our progressive reconstruction method based on an estimate-and-optimize strategy shares some resemblances to those proposed in [3], [10], and [34], which employ certain geometrical properties to provide an initial guess for the subsequent optimization-based 3D reconstruction. For example, to obtain a preliminary approximation of the object, Lipson and Shpitalni [3] assume that the edges in the line drawing have three main axis directions. The approach of Company et al. [10] is

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

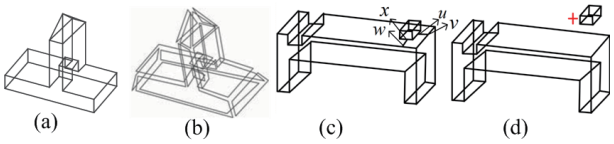


Fig. 3: (a) A line drawing. (b) Face circuits of the line drawing in (a). (c) Two artificial lines $\{x, w\}$ and $\{u, v\}$. (d) Two separated line drawings after preprocessing the line drawing in (c).

applicable only to the input line drawing representing a (quasi-)normalon, while the method proposed by Lee and Fang [34] requires the availability of at least one cubic corner in a desired object. Our approach does not require such special geometrical properties of an underlying object in the line drawing. This is achieved by decomposing the line drawing into multiple parts and progressively optimizing each part, instead of solving the optimization problem on the entire line drawing. Our approach is thus more general and is applicable to a wider class of objects.

3 Assumptions, Preprocessing, and Terminology

Similar to [13], our paper focuses on 3D reconstruction of a large class of common planar-faced solids, called manifolds. On the surface of a manifold, every point has a neighborhood topologically equivalent to an open disk in the 2D Euclidean space [35]. A line drawing in this paper is assumed to be an orthographic projection of the edges of a 3D planar-faced manifold in a generic view, with hidden lines and vertices visible.

Preprocessing. The face topology is very crucial for the line drawing decomposition and reconstruction. Given a line drawing, we use the algorithm in [28] to find its face topology. Here, the face topology denotes the set of circuits that represent all the faces of the 3D object. For example, the line drawing in Fig. 3(a) has 15 faces, as shown in Fig. 3(b). See another example in Figures 4(c) and (d). Artificial lines (see examples in Fig. 3(c)), added by the designer, are used to indicate the coplanarity of two circuits in solid modeling [36], [28]. Detecting artificial lines is an easy task according to the connection between an artificial line and the edges it connects to [13]. After removing the artificial lines, a line drawing becomes two or more separated line drawings (Fig. 3(d)).

For better understanding of the contents in the following sections, we here give the definitions of the terms that appear in the rest of the paper.

Definition 1. *Let a line drawing be the projection of a 3D object. The minimum number of depths (z-coordinates) that can uniquely define this 3D object is called the degree of reconstruction freedom (DRF) for the line drawing.*

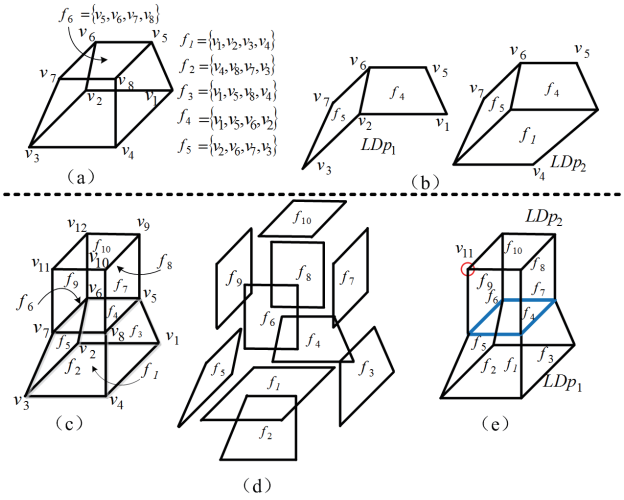


Fig. 4: Illustration of some terms. (a) A line drawing representing a hexahedron. (b) Two 4DRF parts of the line drawing in (a), where the part LD_{p_2} is a 4DRF-extended part of LD_{p_1}. (c) Another line drawing which has ten faces. (d) Ten identified faces of the line drawing in (c). (e) Two neighboring parts $LD_{p_1} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$ and $LD_{p_2} = \{f_6 \cup f_7 \cup f_8 \cup f_9 \cup f_{10}\}$. Once the 3D vertices of LD_{p_1} are recovered, the part LD_{p_2} is the largest-conditional-1DRF neighboring part of LD_{p_1}.

This definition comes from [12], where DRF is used to find a search space for 3D reconstruction. Instead, in our paper, it is used to decompose complex line drawings. Now, let us analyze the DRF for a simple line drawing shown in Fig. 4(a). Assume that the line drawing is the precise projection of a 3D planar-faced object. Thus, all 3D vertices on the same face are coplanar. For example, all the vertices $\mathbf{v}_{1-4}$ are on the plane defined by $a_1x + b_1y + c_1 - z = 0$, which passes through the face $f_1 = (\mathbf{v}_1, \mathbf{v}_2, \mathbf{v}_3, \mathbf{v}_4)$. Next, we can show that the 3D object is defined if z_1, z_2, z_3, and z_5 are given, where z_i is the depth value of vertex $\mathbf{v}_i$. When z_1, z_2 and z_3 are known, the 3D plane $a_1x+b_1y+c_1-z = 0$ is defined. Then, z_4 can be calculated by $z_4 = a_1x_4 + b_1y_4 + c_1$. Since z_4 is known and z_1 and z_5 are given, $\mathbf{v}_8$ is determined because it lies on the plane defined by $\mathbf{v}_1, \mathbf{v}_5$ and $\mathbf{v}_4$. Analogously, the depths of all other vertices can be determined. On the other hand, it is obvious that the given depth values of only three vertices are not sufficient to determine a unique 3D object for this line drawing. Therefore the DRF for this drawing is 4.

Definition 2. *Let a line drawing be $\mathcal{LD} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$ where $\mathcal{V}$, $\mathcal{E}$, and $\mathcal{F}$ are the sets of vertices, edges, and faces of $\mathcal{LD}$, respectively. A partial line drawing, or a part of $\mathcal{LD}$ is formed by one or more connected faces*

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

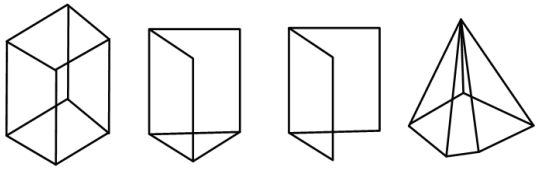


Fig. 5: Examples of 4DRF line drawings.

denoted by $\mathcal{LD}_p = (\mathcal{V}_p, \mathcal{E}_p)$ with $\mathcal{V}_p \subseteq \mathcal{V}$ and $\mathcal{E}_p \subseteq \mathcal{E}$. A neighboring face of LD_p is a face that has at least one edge (but not all) in $\mathcal{LD}_p$. Given two parts $\mathcal{LD}_{p_1}$ and $\mathcal{LD}_{p_2}$, let $F(\mathcal{LD}_{p_1})$ and $F(\mathcal{LD}_{p_2})$ be the sets of the faces in $\mathcal{LD}_{p_1}$ and $\mathcal{LD}_{p_2}$, respectively; let $E(\mathcal{LD}_{p_1})$ and $E(\mathcal{LD}_{p_2})$ be the sets of the edges in $\mathcal{LD}_{p_1}$ and $\mathcal{LD}_{p_2}$, respectively. $\mathcal{LD}_{p_1}$ and $\mathcal{LD}_{p_2}$ are called two neighboring parts if $F(\mathcal{LD}_{p_1}) \cap F(\mathcal{LD}_{p_2}) = \varnothing$ and $E(\mathcal{LD}_{p_1}) \cap E(\mathcal{LD}_{p_2}) \neq \varnothing$.

Fig. 4(b) shows two parts of the line drawing in Fig. 4(a). The face f_1 in $\mathcal{LD}_{p_2}$ is a neighboring face of the part $\mathcal{LD}_{p_1} = \{f_4 \cup f_5\}$. In Fig. 4(e), $\mathcal{LD}_{p_1}$ and $\mathcal{LD}_{p_2}$ are two neighboring parts, which share the same circuit (in blue) but share no face.

Definition 3. *A 4DRF part is a part whose DRF is four. A manifold is called a 4DRF manifold if the DRF for its line drawing is four.*

Fig. 5 shows four examples of 4DRF line drawings. Their complexities vary, though they have the same DRF. The line drawing in Fig. 5(c) has only two faces. It is the simplest 4DRF line drawing. The line drawing shown in Fig. 5(a) has 6 faces and is more complex. It is easy to show that a 4DRF line drawing or 4DRF partial line drawing has at least two faces. The cuboid represented by the line drawing in Fig. 5(a) is a 4DRF manifold.

Definition 4. *A 4DRF-extended part of a 4DRF part $\mathcal{LD}_p$, denoted as $4DRFExt(\mathcal{LD}_p)$, is the part which contains $\mathcal{LD}_p$ and one neighboring face of $\mathcal{LD}_p$, which has two non-collinear edges in $\mathcal{LD}_p$. The largest-4DRF-extended part of a 4DRF part $\mathcal{LD}_p$, denoted as $Largest4DRFExt(\mathcal{LD}_p)$, is the 4DRF part which has the largest number of faces among the parts, each of which contains $\mathcal{LD}_p$.*

Note that the DRFs of $\mathcal{LD}_p$, $4DRFExt(\mathcal{LD}_p)$, and $Largest4DRFExt(\mathcal{LD}_p)$ are all 4. As shown in Fig. 4(b), given a part $\mathcal{LD}_{p_1}$ ($\mathcal{LD}_{p_1} = \{f_4 \cup f_5\}$), a 4DRF-extended part of $\mathcal{LD}_{p_1}$ is the partial line drawing which contains all the faces in $\mathcal{LD}_{p_1}$ and the face f_1, since f_1 passes through two non-collinear edges in $\mathcal{LD}_{p_1}$. For a 4DRF part of a line drawing, it may have no 4DRF-extended part. As shown in Fig. 4(e), each neighboring face of $\{f_3 \cup f_7\}$ shares less than two non-collinear edges with $\{f_3 \cup f_7\}$ and hence the part $\{f_3 \cup f_7\}$ has neither 4DRF-extended part nor largest-4DRF-extended part. It is worth noting that a largest-4DRF-extended part does not necessarily represent a manifold, though this work focuses the decomposition of line drawings representing manifolds only.

Given a 4DRF part $\mathcal{LD}_p$, the following procedure $4DRFExt\_F(\mathcal{LD}_p)$ (Procedure 1) obtains one possible 4DRF-extended part:

Procedure 1 $4DRFExt\_F(\mathcal{LD}_p)$
1. $\mathcal{LD}_{tmp} \leftarrow \mathcal{LD}_p$;
2. add a neighboring face of $\mathcal{LD}_p$ to $\mathcal{LD}_{tmp}$ if this face has two non-collinear edges in $\mathcal{LD}_{tmp}$;
3. $4DRFExt(\mathcal{LD}_p) \leftarrow \mathcal{LD}_{tmp}$;

Procedure 2 $Largest4DRFExt\_F(\mathcal{LD}_p)$
1. $\mathcal{LD}_{tmp} \leftarrow \mathcal{LD}_p$;
2. $LD'_{tmp} \leftarrow 4DRFExt\_F(LD_{tmp})$;
3. **if** $LD'_{tmp} \supset LD_{tmp}$
4. $\quad \mathcal{LD}_{tmp} \leftarrow LD'_{tmp}$; **goto** Step 2;
5. $Largest4DRFExt(\mathcal{LD}_p) \leftarrow LD'_{tmp}$.

Obviously, the output of $4DRFExt\_F(\mathcal{LD}_p)$ is a valid 4DRF-extended part of $\mathcal{LD}_p$ only if $4DRFExt\_F(\mathcal{LD}_p) \supset \mathcal{LD}_p$ ($\neq \mathcal{LD}_p$) is satisfied. Based on $4DRFExt\_F(\mathcal{LD}_p)$, starting from a 4DRF part, called a seed, $Largest4DRFExt(\mathcal{LD}_p)$ can be obtained by the procedure $Largest4DRFExt\_F(\mathcal{LD}_p)$ (Procedure 2).

Definition 5. *A neighboring part of a part $\mathcal{LD}_p$ is called a conditional-1DRF neighboring part of $\mathcal{LD}_p$ if its DRF is 1 when the 3D coordinates of all the vertices of $\mathcal{LD}_p$ are known. A conditional-1DRF neighboring part of $\mathcal{LD}_p$ is called the largest-conditional-1DRF neighboring part of $\mathcal{LD}_p$ if it has the maximum number of faces among all the conditional-1DRF neighboring parts of $\mathcal{LD}_p$. Given a neighboring face f of $\mathcal{LD}_p$, the largest-conditional-1DRF neighbor part of $\mathcal{LD}_p$ containing f is denoted as $LargestCon1DRF(LD_p, f)$.*

We take the line drawing shown in Fig. 4(e) to illustrate the two terms in Definition 5. Assume that a 3D shape has been reconstructed from the part $\mathcal{LD}_{p_1} = \{f_1 \cup f_2 \cup f_3 \cup f_4 \cup f_5\}$. From Definition 2, there exist multiple neighboring parts of $\mathcal{LD}_{p_1}$, such as $\{f_6\}$, $\{f_6 \cup f_7\}$, and $\mathcal{LD}_{p_2} = \{f_6 \cup f_7 \cup f_8 \cup f_9 \cup f_{10}\}$. All of them are also conditional-1DRF neighboring parts of $\mathcal{LD}_{p_1}$, since when the 3D shape of $\mathcal{LD}_{p_1}$ is known, the depth of one vertex (not in $\mathcal{LD}_{p_1}$) of such a part can uniquely define the 3D position of this part.

Given a part $\mathcal{LD}_p$ and one of its neighbor faces f, $LargestCon1DRF(LD_p, f)$ can be obtained by the following procedure $LargestCon1DRF\_F(LD_p, f)$:

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

Procedure 3 $LargestCon1DRF\_F(\mathcal{LD}_p, f)$

1. $\mathcal{LD}_{tmp} \leftarrow f$;
2. $\mathcal{LD}'_{tmp} \leftarrow \mathcal{LD}_{tmp}$;
3. **for** each neighboring face f_j of $\{\mathcal{LD}_p \cup \mathcal{LD}_{tmp}\}$ **do**
4. **if** f_j has two non-collinear edges in $\{\mathcal{LD}_p \cup \mathcal{LD}_{tmp}\}$
5. $\mathcal{LD}'_{tmp} \leftarrow \{\mathcal{LD}'_{tmp} \cup f_j\}$;
6. **end for**
7. **if** $\mathcal{LD}'_{tmp} \supset \mathcal{LD}_{tmp}$
   $\mathcal{LD}_{tmp} \leftarrow \mathcal{LD}'_{tmp}$; **goto** Step 2;
8. $LargestCon1DRF(LD_p) \leftarrow LD'_{tmp}$.

In Fig 4(e), arbitrarily selecting a neighboring face of $\mathcal{LD}_{p_1}$, say f_7, we have $LargestCon1DRF(\mathcal{LD}_{p_1}, f_7) = LD_{p_2}$ by running the procedure $LargestCon1DRF\_F(LD_{p_1}, f_7)$ (Procedure 3) .

Definition 6. *A dual graph G of a line drawing $\mathcal{LD}$ is a graph whose vertices denote the faces of $\mathcal{LD}$, and each of whose edges connects two vertices that are neighboring faces of $\mathcal{LD}$.*

An example of a line drawing and its corresponding dual graph can be found in Fig. 7(a) and (b), respectively.

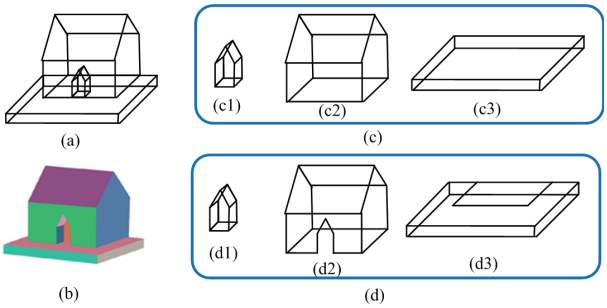


Fig. 6: Illustration of the proposed decomposition algorithm. (a) A 2D line drawing representing a manifold object. (b) 3D object corresponding to (a). (c) Parts obtained by the algorithm in [13], which decomposes a line drawing along the internal faces of the object. (d) Parts obtained by our DRF-based algorithm.

4 DRF-BASED DECOMPOSITION

We first introduce the details of our DRF-based line drawing decomposition algorithm and then discuss its computational complexity.

4.1 Algorithm

We observed that many man-made objects, like the house model shown in Fig. 6, are usually formed by simpler manifolds whose DRFs are four. This motivated us to separate a complex line drawing into a set of 4DRF parts. The resulting 4DRF parts are much less complex and usually have regular geometry, making their 3D reconstruction much easier.

However, for such complex line drawings, finding parts with low DRFs corresponding to manifolds is not trivial. Liu et al. [13] proposed to decompose a complex line drawing from its internal faces where simpler objects are glued. Unfortunately, their algorithm to find the internal faces from a line drawing is NP-complete. To make it run in a reasonable time, a predefined maximum search depth has to be set. As a result, the internal faces cannot be found when the numbers of their edges are larger than the threshold.

Different from [13], our work utilizes a DRF-based algorithm to decompose a line drawing into a set of parts which are approximate to the 4DRF parts representing manifolds. More specifically, our algorithm sequentially decomposes a line drawing into a set of *largest-4DRF-extended* parts and *largest-conditional-1DRF neighboring* parts. Fig. 6 shows a decomposition result by our algorithm, compared to that by the algorithm in [13]. For this example, both the algorithms lead to successful

Algorithm 1 DRF-based line drawing decomposition.

Input: A line drawing $G_0 = (\mathcal{V}_0, \mathcal{E}_0, \mathcal{F}_0)$ where $\mathcal{V}_0$, $\mathcal{E}_0$, and $\mathcal{F}_0$ are the sets of vertices, edges, and faces, respectively.

1. build the dual graph $G = (\mathcal{V}, \mathcal{E})$ of G_0;
2. $Label(v) \leftarrow 0$ for every vertex $v \in \mathcal{V}$; $i \leftarrow 0$; $\mathcal{V}_u \leftarrow \emptyset$; $\mathcal{E}_u \leftarrow \emptyset$; let $G_u = (\mathcal{V}_u, \mathcal{E}_u)$;
3. $flag_{l4} \leftarrow 0$; find a set of seeds G_{sp} from $G - G_u$;
4. **if** $G_{sp} = \emptyset$ **goto** Step 9;
5. **for** each seed G_{sp_j} in G_{sp} **do**
6. (a) $G_{tmp} \leftarrow 4DRFExt\_F(\widehat{G_{sp_j}})$ (Procedure 1);
7. (b) **if** $G_{tmp} \supset G_{sp_j}$ **then**
   $flag_{l4} \leftarrow 1$ and **goto** Step 10;
   end if
8. **end for**
9. **if** $flag_{l4} = 0$ **then**
   $i \leftarrow i + 1$, $G'_{p_i} \leftarrow G - G_u$, and **goto** Step 22;
   end if
10. $G'_{p_i} \leftarrow Largest4DRFExt\_F(\widehat{G_{tmp}})$ (Procedure 2);
11. $i \leftarrow i + 1$; $Label(v) \leftarrow 1$ for every vertex $v \in G'_{p_i}$;
12. add v to $\mathcal{V}_u$ for every vertex $v \in G'_{p_i}$; add e to $\mathcal{E}_u$ for every edge $e \in G'_{p_i}$;
13. **if** $G_u = G$ **goto** Step 22;
14. $AdjVer(G_u) \leftarrow$ all the neighboring vertices of G_u; $flag_{lc1} \leftarrow 0$;
15. **for** each vertex $v_j \in AdjVer(G_u)$ **do**
16. check if there is another neighboring face of $\widehat{G_u}$ in G_0 containing two non-collinear edges in $\{\widehat{G_u} \cup \widehat{v_j}\}$. If yes, $flag_{lc1} \leftarrow 1$ and **goto** Step 19;
17. **end for**
18. **if** $flag_{lc1} = 0$, **goto** Step 3;
19. $G'_{p_i} \leftarrow LargestCon1DRF\_F(\widehat{G_u}, \widehat{v_j})$ (Procedure 3);
20. $i \leftarrow i + 1$; $Label(v) \leftarrow 1$ for every vertex $v \in G'_{p_i}$;
21. add v to $\mathcal{V}_u$ for every vertex $v \in G'_{p_i}$; add e to $\mathcal{E}_u$ for every edge $e \in G'_{p_i}$; **goto** Step 13;
22. **if** $i > 1$ **Output:** $(\widehat{G'_{p_1}}, ..., \widehat{G'_{p_i}})$ **else Output:** $\widehat{G'_{p_1}}$.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

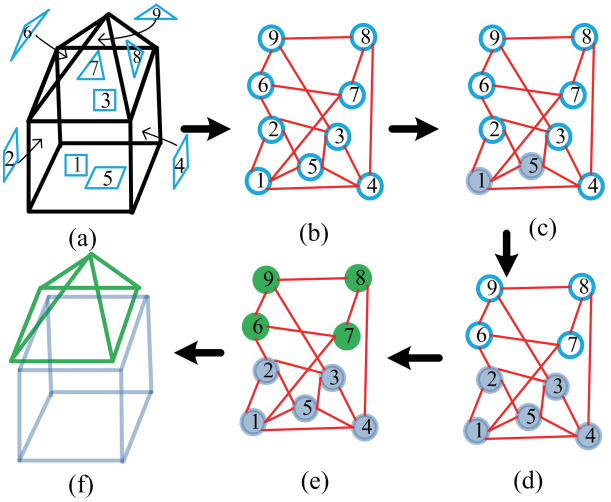


Fig. 7: Illustration of Algorithm 1. (a) A line drawing with 9 faces which are denoted by the 9 numbers each enclosed by a blue pattern whose shape is similar to its corresponding face in the line drawing. (b) The dual graph of (a). (c) A pair of adjacent vertices (1 and 5) are selected to find the largest-4DRF-extended part. (d) The largest-4DRF-extended part $\{1, 2, 3, 4, 5\}$ obtained by Step 8 in Algorithm 1. (e) The largest-conditional-1DRF neighboring part $\{6, 7, 8, 9\}$ of the part $\{1, 2, 3, 4, 5\}$. In (e), all the 9 vertices are labeled by 1. (f) Two decomposed parts.

but slightly different decompositions. We will give a thorough evaluation of these two algorithms in Section 6.

The detailed decomposition steps are summarized in Algorithm 1 and illustrated in Figure 7. In Step 1, the dual graph G (Fig. 7(b)) of the input line drawing G_0 is built to facilitate the design of the decomposition algorithm. G_u in Step 2 is a subgraph of G, and is used to keep the currently decomposed parts. We use a binary label $Label(v)$ for each vertex v in G to denote whether v is in the obtained parts in G_u.

In Steps 2–22, to have a concise description, a sign and its over-hatted sign are used to denote the two corresponding entities in G and G_0, respectively. For example, v_j is a vertex in G while $\widehat{v_j}$ is the corresponding face in G_0; G_{sp_j} denotes two adjacent vertices in G while $\widehat{G_{sp_j}}$ denotes the corresponding two adjacent faces in G_0. $G-G_u$ denotes the remaining subgraph by removing $\mathcal{V}_u$ and the edges connecting the vertices in $\mathcal{V}_u$ from G.

Through Steps 3–12 our algorithm finds the parts belonging to largest-4DRF-extended parts. A binary label $flag_{l4}$ is used to check if a largest-4DRF-extended part can be found. More specifically, in Step 3 a seed set G_{sp} is constructed for the generation of a largest-4DRF-extended part. In this step, all the pairs of adjacent faces, each pair sharing a trihedral vertex (i.e., a vertex passed through by three faces) in $\widehat{G-G_u}$, are selected as the seeds and added into G_{sp}. See an example of such a seed (adjacent vertices 1 and 5) in Fig. 7(c). We use such pairs as the seeds since a trihedral vertex usually exists in simple manifolds that are the targeted decomposition results, while a vertex passed through by more than three faces (say, v_5 in Fig. 4(c)) often indicates a place where two simple manifolds are merged into a complex manifold. The part growing step ($Largest4DRFExt\_F$) starting from a seed usually gives a desired largest-4DRF-extended part, e.g., $\{1, 2, 3, 4, 5\}$ in Fig. 7(d). Note that if the seed set constructed in Step 3 is empty, Algorithm 1 will output $\widehat{G-G_u}$ as the last part and stop.

Steps 14–17 check whether G_u has a largest-conditional-1DRF neighboring part using a binary label $flag_{lc1}$. According to $flag_{lc1}$ in Steps 16 and 18, the algorithm goes on to find a largest-conditional-1DRF neighboring part of G_u (Steps 19–21) or another largest-4DRF-extended part (Steps 3–12). Fig. 7(e) shows a largest-conditional-1DRF neighboring part $\{6, 7, 8, 9\}$ of the part $\{1, 2, 3, 4, 5\}$. Fig. 7(f) gives two decomposed parts resulting from Algorithm 1.

4.2 Complexity

We now analyze the complexity of the decomposition algorithm. Suppose a line drawing G_0 has N_v vertices, N_e edges, and N_f faces. Assume that every face in G_0 has less than K_e edges or adjacent faces, and G_0 is decomposed into K_p parts. Then the average number of faces in a part is N_f/K_p.

The main computation is carried out by Steps 3–21. In Step 3, the algorithm tests each pair of two adjacent faces in a subgraph of G_0 (i.e., $G-G_u$ in G) and adds those sharing a trihedral vertex into the set of seeds. This step takes less than $O(K_eN_f)$ time and obtains less than K_eN_f seeds. The number of seeds determines the number of loops in Steps 6 and 7. Step 6 calls the procedure $4DRFExt\_F()$ to test whether the current seed can grow or not. In procedure $4DRFExt\_F()$, to test if a given neighboring face f of $\mathcal{LD}_p$ has two non-collinear edges in $\mathcal{LD}_p$, we first find all the edges in f that are also in $\mathcal{LD}_p$ and then check if there are two non-collinear edges among them. The former and the latter are bounded by $O(K_eN_e)$ and $O({K_e}^2)$, respectively. Therefore the time of one execution of $4DRFExt\_F()$ is bounded by $O(N_f(K_eN_e+{K_e}^2)) = O(K_eN_eN_f)$. Considering Step 7 takes much less time than Step 6, the computation of Steps 5–8 is bounded by $O({K_e}^2N_e{N_f}^2)$. The main computation of Steps 4–9 is determined by Steps 5–8, and thus the computation of Steps 4–9 is also bounded by $O({K_e}^2N_e{N_f}^2)$. The main computation of Steps 10–13 is determined

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

by the procedure $Largest4DRFExt\_F()$ in Step 10, which is approximately equal to repeated execution of $4DRFExt\_F()$ for N_f/K_p times. Thus, the time of one execution $Largest4DRFExt\_F()$ is bounded by $O(K_e N_e {N_f}^2/K_p)$. Then, the total time of obtaining a largest-4DRF-extended part (Steps 3–13) is bounded by $O(K_e N_f) + O({K_e}^2 N_e {N_f}^2) + O(K_e N_e {N_f}^2/K_p) = O({K_e}^2 N_e {N_f}^2)$.

Next, we discuss the complexity of Steps 14–21. In these steps, the algorithm first checks if G_u has a conditional-1DRF neighboring part or not (Steps 15–17), and then finds a conditional-1DRF neighboring part (Steps 19–21) or another largest-4DRF-extended part (Steps 3–13). The complexity of Steps 14–21 is thus approximate to the time taken by Steps 3–13. Finally, the algorithm conducts Steps 3-13 or Steps 14–21 for K_p times. Considering K_e is constant, we have the complexity of Algorithm 1 bounded by $O(K_e^2 N_e {N_f}^2 K_p) = O(K_p N_e {N_f}^2)$, which is polynomial.

In fact, from our experiments, we find that the time spent by Algorithm 1 and by the line drawing preprocessing step (typically less than 1 second), can be almost neglected compared to the time taken by the 3D reconstruction algorithm (see Fig. 12(a); typically from 10 to 30 seconds).

5 Progressive 3D Reconstruction

DRF-based decomposition, introduced in the previous section, results in a set of simpler line drawings. A straightforward solution to recover a 3D model from such line drawing parts is to first reconstruct 3D shapes from individual parts of the line drawing independently and then integrate the reconstructed 3D shapes into a complete 3D model. However, as shown in Fig. 2(d), such an approach might easily cause reconstruction artifacts. To address this problem we introduce a novel progressive 3D reconstruction algorithm. Below we first give the algorithm details in Section 5.1 and then the implementation details in Section 5.2 and 5.3.

5.1 Algorithm

Our algorithm takes an estimate-and-optimize strategy and sequentially reconstructs 3D shapes from line drawing parts one by one based on already reconstructed parts. The algorithm is summarized in Algorithm 2.

Due to the progressive nature our algorithm would accumulate reconstruction errors. To reduce such errors we use the following scheme to derive a reconstruction sequence (Step 2), i.e., an ordered list of the decomposed parts, denoted as $(G_{p_1}, \ldots, G_{p_N})$, where N is the number of the decomposed parts of the input line drawing. As illustrated in Fig. 8 we first build a graph G' with each vertex representing a line drawing part and each edge connecting two neighboring parts. To determine the initial part G_{p_1}, we estimate the reconstruction error starting from each vertex $v_{G'_{p_i}}$ (i.e., $\widehat{G'_{p_i}}$ of Algorithm 1) and take the one with the minimum error as G_{p_1}. Mathematically it is formulated as follows:

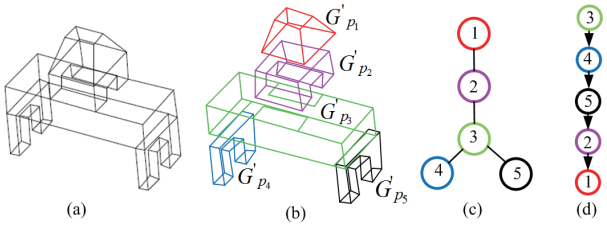


Fig. 8: Illustration of determining the reconstruction sequence. (a) and (b) An input line drawing and its decomposed parts. (c) A graph G' built where each vertex represents a part and each edge represents the connection between two neighboring parts. (d) The determined reconstruction sequence $(G_{p_1}, ...G_{p_5}) = (G'_{p_3}, G'_{p_4}, G'_{p_5}, G'_{p_2}, G'_{p_1})$.

$$G_{p_1} = \underset{i}{\operatorname{argmin}}\, Err(v_{G'_{p_i}}) = \sum_{j=1, j\neq i}^{N} dis(v_{G'_{p_i}}, v_{G'_{p_j}}), \tag{1}$$

where $dis(v_{G'_{p_i}}, v_{G'_{p_j}})$ denotes the length of the shortest path between two vertices in the graph G', which makes G_{p_1} roughly correspond to the center of the graph G'. In case there are two or more vertices with the same minimum Err, we pick G_{p_1} as the one corresponding to the part with the maximum N_{ppe}/N_e, where N_{ppe} denotes the number of pairs of parallel edges in a part, and N_e denotes the number of the edges in the part, since a part with more parallel lines usually corresponds to a more regular 3D shape and its reconstruction is thus more robust. We determine the order of the rest of the parts using classic breadth-first search in G', starting from the vertex corresponding to G_{p_1}.

Algorithm 2 3D reconstruction from a complex line drawing.

1. decompose an input line drawing with Algorithm 1;
2. determine the reconstruction sequence $(G_{p_1}, \ldots, G_{p_N})$;
3. reconstruct a 3D O_{p_1} shape from G_{p_1};
4. **for** i = 2 **to** N **do**
5. a) estimate a rough 3D shape $\widetilde{O_{p_i}}$ with the information of $\{G_{p_1}, \ldots, G_{p_i}\} \cup \{O_{p_1}, \ldots, O_{p_{i-1}}\}$;
6. b) reconstruct a 3D shape O_{p_i} for G_{p_i} based on $\widetilde{O_{p_i}} \cup \{O_{p_1}, \ldots, O_{p_{i-1}}\}$ and $\{G_{p_1}, \ldots, G_{p_i}\}$;
7. **end for**

In Step 3, a 3D shape is reconstructed from the initial part G_{p_1}. Steps 4–7 sequentially reconstruct 3D shapes from the subsequent parts, where $\widetilde{O_{p_i}}$ is the rough 3D shape estimated from the 3D information

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

of the reconstructed 3D shapes $\{O_{p_1}, O_{p_2}, \ldots, O_{p_{i-1}}\}$ and $\{G_{p_1}, \ldots, G_{p_i}\}$. With $\widetilde{O_{p_i}}$, $\{O_{p_1}, \ldots, O_{p_{i-1}}\}$, and $\{G_{p_1}, \ldots, G_{p_i}\}$, a 3D shape O_{p_i} is recovered by an optimization-based algorithm. As we will show in Section 6 that, by estimating rough 3D shapes first, the reconstruction algorithm usually converges much faster.

Step 6 can be achieved by adapting existing optimization-based 3D reconstruction algorithms like those proposed in [3], [10], and [12]. Specifically, O_{p_i} can be reconstructed by minimizing the following objective function:

$$\Phi(\mathbf{Z}_{p_i}) = \sum_{j=1}^{N_c} w_j \phi_j(\mathbf{Z}_{p_1}, \ldots, \mathbf{Z}_{p_{i-1}}, \mathbf{Z}_{p_i}), \qquad (2)$$

where $\mathbf{Z}_{p_i}$ is the set of the z-coordinates of the vertices in the ith part which are unknown and their initial values are obtained from $\widetilde{O_{p_i}}$, $\mathbf{Z}_{p_1}, \ldots, \mathbf{Z}_{p_{i-1}}$ denote the known z-coordinates of the vertices in the $i-1$ already reconstructed parts, ϕ_j, $1 \leq j \leq N_c$, are the N_c constraints, i.e., image regularities, derived from all the i parts, and w_j is a weighting factor. We adopt the most commonly used image regularities [3], [10]–[12], including face planarity, line parallelism, line collinearity, skewed facial symmetry, isometry, corner orthogonality, and minimizing the standard deviation of angles in the reconstructed objects.

Below we give the implementation details of how to reconstruct an initial 3D shape (Step 3) in Section 5.2, and how to estimate an initial 3D shape for a subsequent part (Step 5) in Section 5.3

5.2 3D Reconstruction of Initial Part

It is well known that the robustness of optimization-based methods for 3D reconstruction from line drawings is often sensitive to the initial settings of the z-coordinates of the vertices [12]. There is no available 3D shape to guide 3D reconstruction from the initial part G_{p_1}. For robustness we independently reconstruct a 3D shape from G_{p_1} with different initial settings: randomized initialization of z-coordinates for multiple times ([11], [12], [13]) and initial zero setting of z-coordinates ([14]). The reconstruction result corresponding to the minimum value of the objective function $\Phi(\mathbf{Z}_{p_1}) = \sum_{j=1}^{N_c} w_j \phi_j(\mathbf{Z}_{p_1})$ (part of Equation (2)) is selected as the final 3D shape of the initial part. In our experiments, we find that ten randomizations plus one zero setting are sufficient to obtain a good 3D shape. It is worth mentioning that if the initial part is a (quasi-)normalon [10] or has a cubic corner [10], [34], [37], there exist more advanced algorithms to derive a more effective initial approximation for the optimization process.

5.3 3D Rough Shape Estimation of Subsequent Parts

When the z-coordinates of the vertices of a line drawing have been partially reconstructed, various image regularities which capture 3D geometrical relationships between parts can be used to estimate a rough shape close to the optimal one for a subsequent part, as illustrated in Fig. 9. For example, if two edges in two line drawing parts under orthographic projection are parallel, they should be parallel in 3D space too.

From Section 3, we know that when the 3D shape of a part has been reconstructed, the DRF of its neighboring part reduces to one. In our derived reconstruction sequence, it is easy to see that G_{p_i} is a neighboring part of $\{G_{p_1}, \ldots, G_{p_{i-1}}\}$. Thus, the 3D shape of G_{p_i} can be derived given a known depth (z-coordinate) of one of its *free vertices* which is in G_{p_i} but not in the already reconstructed parts.

As illustrated in Figs. 9(b)-(e), we use image regularities [3], [10] including line collinearity (IR_{lc}), line parallelism (IR_{lp}), line verticality (IR_{lv}), and SDA (standard deviation of the angles formed at a vertex equal to zero, denoted as IR_{sda}) to estimate the depth of one free vertex of G_{p_i}. After the depth of one free vertex is reconstructed, face planarity (Fig. 9(a)) can then be used to derive the depths of the other vertices in this part. It is possible that multiple image regularities are in conflict with each other. To solve this problem, we specify the priority of the four image regularities as $IR_{lc} > IR_{lp} > IR_{lv} > IR_{sda}$. For instance, for a free vertex whose depth can be derived by both IR_{lp} and IR_{lv}, we use line parallelism to estimate its depth.

In the implementation, we firstly select all the vertices with unknown depths as free vertex candidates, which connect to a part whose 3D shape has been reconstructed, and then we check if the condition of any image regularity is satisfied for one of the free vertex candidates in the order of the priority. If one image regularity is satisfied, we select the corresponding vertex as the free vertex and compute its depth with this image regularity. Finally, face planarity is exploited to derive the rough depths of the other vertices in this part. Note that when IR_{lp} is used to derive the depth of a free vertex, there usually exist multiple reconstructed lines parallel to the line containing the free vertex. In this case, the z-coordinate of the free vertex is set to the average of the values derived by these reconstructed lines.

6 Experiments

We conducted several experiments to evaluate the efficiency and effectiveness of the proposed decomposition and reconstruction algorithms. We focused on comparisons with the Divide-and-Conquer (DaC) method

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

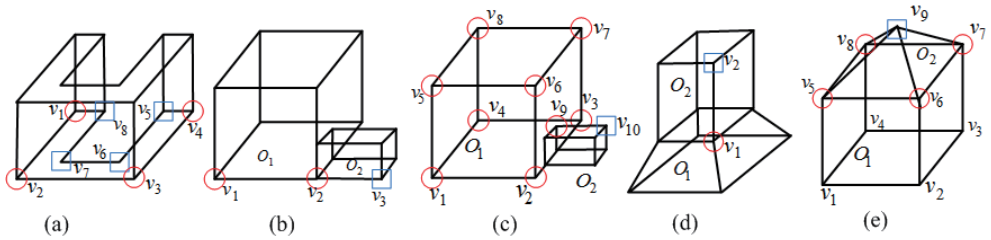


Fig. 9: Examples of estimating the unknown depths of vertices with image regularities. The vertices whose depths have been recovered are marked by ◯ in red and their 3D positions are used to estimate the depths of the vertices marked by □ in blue. (a) shows that the depths of the vertices v_5, v_6, v_7, v_8 can be computed with the 3D positions of v_1, v_2, v_3, v_4 using face planarity. In (b), the depth of the vertex v_3 can be obtained with v_1 and v_2 using line collinearity. (c) shows how to estimate the depth of the vertex v_{10} using line parallelism, since edges $\{v_1, v_2\}$, $\{v_3, v_4\}$, $\{v_5, v_6\}$, and $\{v_7, v_8\}$ are parallel with the edge $\{v_9, v_{10}\}$ and the 3D positions of the vertices v_{1-9} have been recovered. (d) shows that the depth of the vertex v_2 is estimated by line verticality, since the 3D shape O_1 of the bottom part has been reconstructed. (e) shows a case in which the depth of the vertex v_9 is estimated by SDA.

proposed by Liu et al. [13] since their method can handle more complex manifold objects than other previous methods. We implemented the proposed algorithms in C++, and ran them on a PC with an Intel(R) Dual Core(TM) i5 CPU M540@2.53GHz (only a single thread used for simplicity).

In the first experiment we tested our decomposition algorithm (Algorithm 1) and the decomposition algorithm in DaC (Steps 1 and 2 of Algorithm 3 in [13]) on 30 line drawings shown in the first and fourth columns of Fig. 10. Most of these line drawings were collected from previous papers such as [12], [13], and [38]. The DRFs of these line drawings vary from 4 to 17.

For the line drawings with indices (1)–(24), our algorithm obtained the decomposition results similar to those by DaC. Neither our algorithm (because their DRFs are already 4) nor DaC (because they have no internal faces) succeeded to decompose the two 4DRF line drawings (25) and (26). However, we will show in Section 7 that our algorithm can be extended to handle these two drawings by generating some new vertices and edges.

The line drawings (27–30) show four manifolds which could not be decomposed by DaC (with the maximum search depth D_{max} set to 10), but were successfully decomposed by our algorithm. In these line drawings, the internal faces have more than 10 vertices, which is beyond the predefined maximum search depth for DaC. The decomposition results of the line drawings (27) and (28) indicate that our algorithm can be extended to handling curved-face manifolds if a curved surface is approximated by multiple planar faces. The time for decomposing each of these 30 line drawings was within 0.01 second with our algorithm, and was much less than that needed by DaC (about 5–14 seconds for each successfully decomposed drawing).

We further evaluated computational performance on line drawings of increasing complexity, as shown

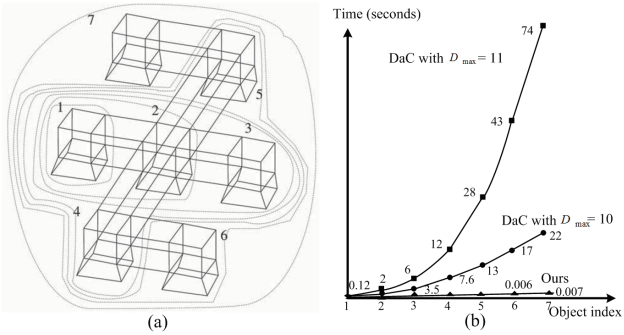


Fig. 11: (a) Seven line drawings of increasing complexity indexed by seven loops. (b) The times used to decompose the line drawings in (a) by our algorithm and DaC.

in Fig. 11. There are in total seven line drawings (Fig. 11(a)), indexed by seven loops O_{1-7} with $O_i \subset O_{i+1}$, $1 \leq i \leq 6$. The times for decomposing O_{1-7} by our algorithm, DaC with $D_{max} = 10$ and DaC with $D_{max} = 11$, are illustrated by the three respective curves in Fig. 11(b). The results indicate that our algorithm is approximately linear in the number of line drawing parts, while DaC is largely exponential. In addition, on O_7, our algorithm was about four orders of magnitude faster than DaC with $D_{max} = 11$.

We conducted another experiment to evaluate the 3D reconstruction performance of the proposed method, including the decomposition step. Our method, PR for short, was compared to DaC and LS, the method by Lipson and Shpitalni [3]. LS is a typical 3D reconstruction method without line drawing decomposition. For fair comparison, the same reconstruction algorithm [3] was used for PR, DaC and LS.

The line drawings (1)–(24) in Fig. 10 were used in this experiment since the decomposition results by DaC and our algorithm were similar. The convergence criteria

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

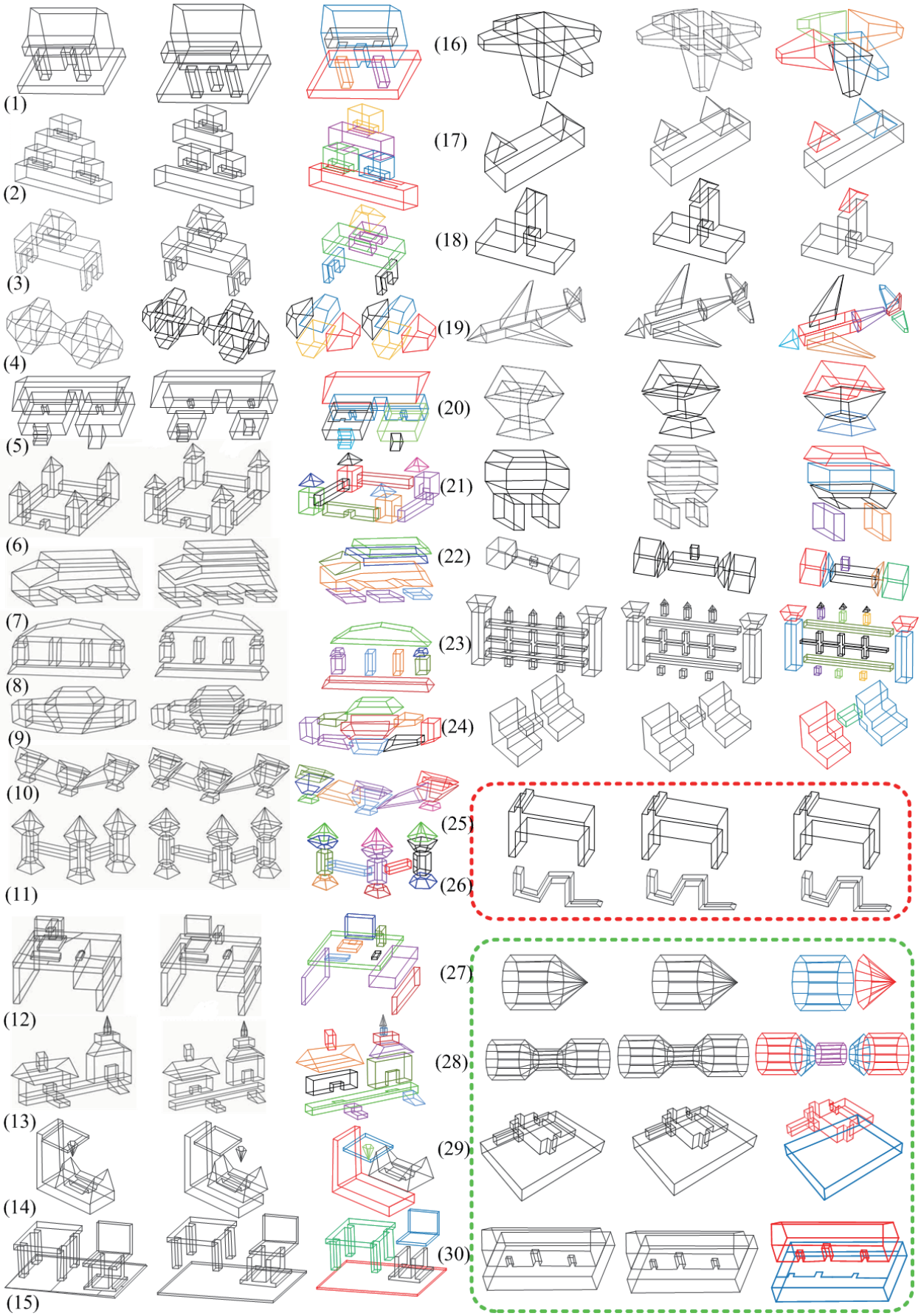


Fig. 10: Thirty tested line drawings (1st and 4th columns), and their decomposition results by DaC (2nd and 5th columns) and our algorithm (3rd and 6th columns). Better viewed on the screen by enlarging the figure.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

for all the algorithms were the same: stopped when the difference between the values of the objective function in two consecutive iteratons was below a given threshold (0.001 in our experiment). For each line drawing we ran each algorithm ten times. Two measurements were adopted to evaluate the robustness of the reconstruction process using these algorithms.

One is the average planarity error (APE) of all the faces of the ten reconstructed 3D object parts O_{1-10} from a line drawing $\mathcal{L}$, defined as

$$APE(O_{1-10}|\mathcal{L}) = \frac{1}{10\Delta L}\sum_{k=1}^{10}\frac{1}{N_f}(\sum_{i=1}^{N_f}\sum_{j=1}^{M_{f_i}} D_{ij}^k), \quad (3)$$

where $D_{ij}^k = \frac{|a_{f_i}^k x_j^k + b_{f_i}^k y_j^k + c_{f_i}^k z_j^k + 1|}{\sqrt{(a_{f_i}^k)^2+(b_{f_i}^k)^2+(c_{f_i}^k)^2}}$, $a_{f_i}^k, b_{f_i}^k$ and $c_{f_i}^k$ are the parameters of the best-fit plane for face f_i obtained from the kth running (how to obtain the best-fit plane from a set of vertices can be found in [3]), M_{f_i} is the number of vertices in face f_i, $\Delta L = \max\{\Delta_x, \Delta_y\}$ denotes the size of the original line drawing $\mathcal{L}$ (Δ_x is the width and Δ_y is the height of $\mathcal{L}$), N_f is the number of the faces in $\mathcal{L}$, and (x_j^k, y_j^k, z_j^k) is the 3D coordinate of the jth vertex of face f_i obtained from the kth running. The term D_{ij}^k denotes the distance between vertex (x_j^k, y_j^k, z_j^k) and the best-fit plane.

Another measurement is the average line-parallelism error (ALE) of all the parallel lines of the ten reconstructed 3D objects O_{1-10}, defined as

$$ALE(O_{1-10}|\mathcal{L}) = \frac{1}{10N_p}\sum_{k=1}^{10}\sum_{(l_i,l_j)\in S}\theta(((l_i^k)^{3D}, (l_j^k)^{3D})), \quad (4)$$

where S is the set of pairs of parallel lines (l_i, l_j) in the line drawing which are recognized as parallel in 3D space by users, N_p is the number of pairs in S, and $\theta(((l_i^k)^{3D}, (l_j^k)^{3D})) = cos^{-1}((\overline{l_i^k})^{3D} \cdot (\overline{l_j^k})^{3D})$ is the angle between two unit vectors $(\overline{l_i^k})^{3D}$ and $(\overline{l_j^k})^{3D}$ of two 3D parallel lines corresponding to l_i and l_j, respectively, from the k running. In this experiment, S was obtained by these two steps: 1) find the set of all the pairs of parallel lines in the 2D line drawing (two lines were regarded as parallel if the angle between them was less than 7°); 2) then remove the pairs that were not interpreted as parallel by the users.

The quantitative evaluation results for these algorithms on the test examples are shown in Fig. 12. Fig. 12(a) indicates that PR is more efficient than the other two algorithms for all the examples. It is because PR computes initial 3D shapes which are already close to the optimal ones, thus greatly reducing the iterations of the optimization algorithm in reconstruction. It is shown in Figs. 12(b) and (c) that in most cases our method led to more robust reconstruction results than both DaC and LS. In fact large values of APE and ALE often signal distorted or failed reconstructions. Fig. 13 shows the

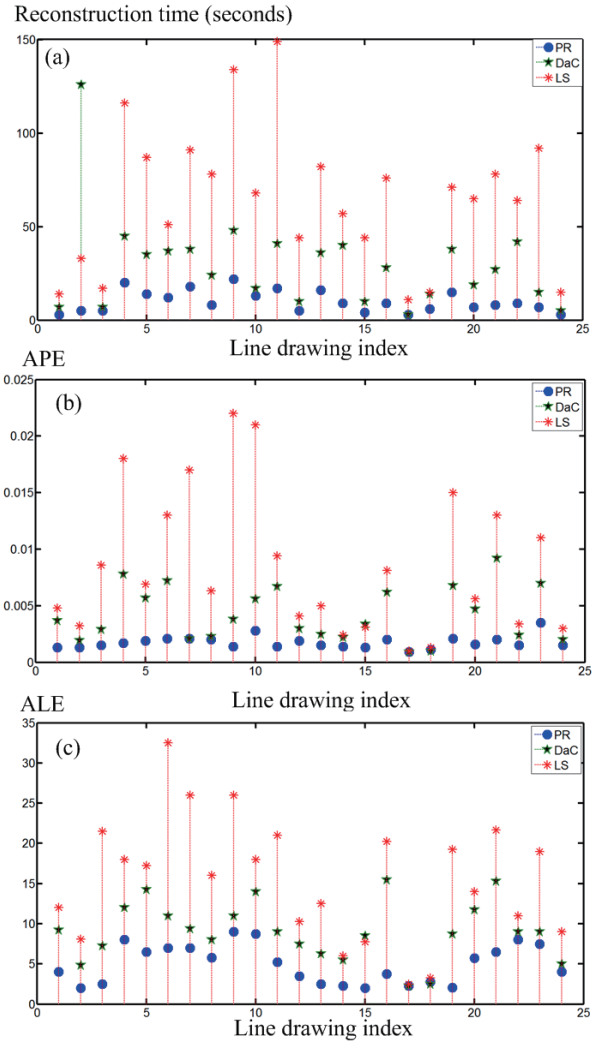


Fig. 12: (a) Reconstruction times of PR, DaC, and LS, including the line drawing decomposition step for PR and DaC, but excluding the face identification step for all the three methods). (b) Reconstruction robustness comparison evaluated by APE. (c) Reconstruction robustness comparison evaluated by ALE.

reconstruction results from the first six line drawings of the test examples. It can be seen that for some examples, such as the one in the third column, although DaC could give roughly correct shapes from the decomposed parts, the integrated results might suffer from artifacts. This is mainly because DaC fails to faithfully maintain the globally geometrical relationships between parts during reconstruction.

Our algorithm is largely robust to sketching errors. Take the line drawing in Fig. 10(6) as an example. To analyze the sensitivity of our algorithm, we generated its random variations by randomly disturbing the 2D x- and y-coordinates of the vertices according to a Gaussian distribution $N(0, \sigma)$. Figs. 14(b–d) show three such noisy

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

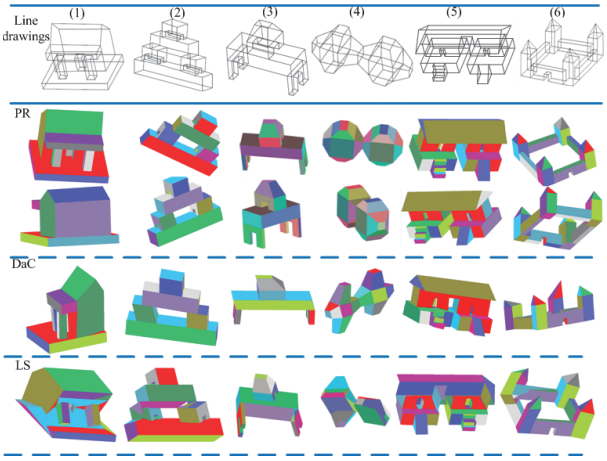


Fig. 13: Experimental results on six test examples. The first row shows the line drawings. The second and third rows show two views of the reconstruction results by PR. The fourth and fifth rows show the results by DaC and LS, respectively. Different colors are used to denote the recovered faces of the reconstructed 3D objects.

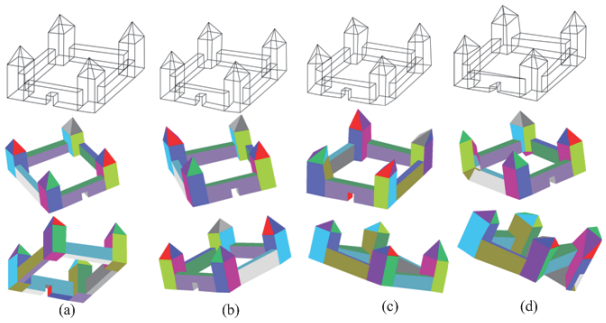


Fig. 14: (a) The line drawing in Fig. 10(6). (b) to (d): line drawings with increasing sketching noise (with $\sigma = W/400$, $W/200$, and $W/100$, respectively). The middle and bottom rows show the results obtained by our algorithm, under two different views.

line drawings, with increasing sketching noise (σ being $W/400$, $W/200$, and $W/100$, respectively, where W is the width of the line drawing, shown in Fig. 14(a) too). The middle and bottom rows show the corresponding reconstruction results by our algorithm. It can be seen from (b) that our algorithm almost perfectly handles a slightly noisy line drawing. For the line drawing with stronger sketching errors our algorithm is still able to produced reasonably good results. However, when the input line drawing is highly distorted, the result is not satisfactory any more.

7 Discussions

DRF-based Decomposition. Our decomposition algorithm can deal with a wide range of objects formed by 4DRF parts. From our observations, we found that a line drawing that can be well decomposed is required to satisfy two conditions. First, there are no missing edges and vertices in the line drawing. That is, the input line drawing must contain all the vertices and edges of the 4DRF manifolds forming the desired 3D object. Second, there must exist at least one trihedral vertex in the line drawing.

The second condition comes from the fact that in Step 3 of Algorithm 1, each selected pair (seed) of faces share a trihedral vertex. The line drawing shown in Fig. 15(a) is one without any trihedral vertex and thus cannot be decomposed by Algorithm 1. However, if a seed is not required to share a trihedral vertex (i.e., any two neighboring faces can be a seed), then Algorithm 1 can still decompose such line drawings. Fig. 14(b) gives such a result, which contains four decomposed parts p_{1-4}. Obviously, a more desired decomposition is to have parts p_{1-3} combined into a single part. One possible solution to merge these non-manifold small parts into a 4DRF manifold is to use the information provided by p_4. From p_4, we can run the face identification algorithm in [28] on p_4 to obtain the new face $(1, 2, 3, 4, 5)$. Such face information makes the combination of p_{1-3} still a 4DRF manifold. In fact, we can always merge decomposed parts each with fewer faces into a larger line drawing, which may or may not represent a 4DRF manifold.

Therefore, to remove the second condition, the following four steps can be carried out: 1) run Algorithm 1 first; 2) check whether there is a part without any trihedral vertex; 3) if there is such a part, then on this part, run Algorithm 1 again but with a modified seed selection criterion, allowing any pair of neighboring faces as a seed; and 4) finally merge the parts whose numbers of faces are smaller than a threshold. Developing a complete merging method is our future work.

As shown in Figs. 10(25) and (26), a 4DRF object with trihedral vertices and many faces cannot be decomposed by Algorithm 1 or DaC. However, if suitable new edges and vertices are generated on the original line drawing, then the line drawing can still be decomposed into 4DRF parts. For example, in Fig. 15(c), if the dotted edges are added to the original line drawing represented by the solid lines, our algorithm could separate it into four cuboids (Fig. 15(d)). How to design a method to automatically add such edges and vertices is an interesting topic to explore further in the future.

Progressive 3D Reconstruction. The better performance of our approach comes from the estimate-and-optimize strategy for each subsequent part. There is still room for improvement in this strategy. The estimation of the initial 3D shape for one part may be improved by selecting the most suitable regularity depending on the nature of each part (such as using the axonometric inflation scheme in [10] for a quasi-normalon part). In

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

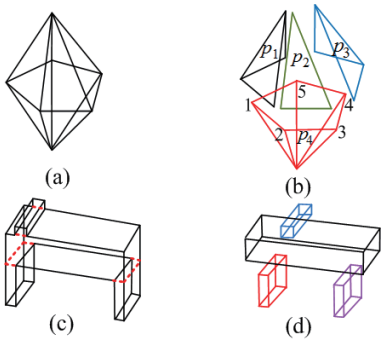


Fig. 15: (a) An object without a trihedral vertex. (b) One possible decomposition result by Algorithm 1 with the modified seed selection criterion. (c) A line drawing (solid lines) representing a 4DRF object that cannot be decomposed. (d) Decomposition result by Algorithm 1 if the new edges (dotted lines in (c)) are added to the original line drawing.

addition, a better strategy, which considers both the topological relationship between parts and the reconstruction quality of each part, can be utilized to determine a better reconstruction sequence of all the decomposed parts.

3D Beautification. After a 3D object is reconstructed by our algorithm (or other algorithms), it is possible to use a beautification algorithm to improve the result. There exist a few methods [39], [40], [41] for this task, in which the goal is to refine the x-, y-, and z-coordinates of all the 3D vertices for example through constrained optimization. It should be mentioned that if a reconstructed object is too distorted, a 3D beautification algorithm cannot help much usually; developing a good reconstruction algorithm is more crucial.

ACKNOWLEDGMENTS

This work was partially supported by grants from the Research Grants Council of HKSAR, China (Project No. 113513 and 11204014), Science, Industry, Trade, and Information Technology Commission of Shenzhen Municipality (No. JC201005270378A), and the Construct Program of the Key Discipline in Hunan Province.

REFERENCES

[1] X. Chen, S. Kang, Y. Xu, J. Dorsey, and H. Shum, "Sketching reality: Realistic interpretation of architectural designs," *ACM Trans. Graph.*, vol. 27, no. 2, 2008.

[2] P. Company, A. Piquer, M. Contero, and F. Naya, "A survey on geometrical reconstruction as a core technology to sketch-based modeling," *Computers & Graphics*, vol. 29, no. 6, pp. 892–904, 2005.

[3] H. Lipson and M. Shpitalni, "Optimization-based reconstruction of a 3d object from a single freehand line drawing," *Computer-Aided Design*, vol. 28, no. 8, pp. 651–663, 1996.

[4] L. Olsen, F. Samavati, M. Sousa, and J. Jorge, "Sketch-based modeling: A survey," *Computers & Graphics*, vol. 33, no. 1, pp. 85–103, 2009.

[5] A. Rivers, F. Durand, and T. Igarashi, "3d modeling with silhouettes," *ACM Trans. Graph.*, vol. 29, no. 4, pp. 109:1–10, 2010.

[6] R. Schmidt, A. Khan, K. Singh, and G. Kurtenbach, "Analytic drawing of 3d scaffolds," *ACM Trans. Graph.*, vol. 28, no. 5, 2009.

[7] L. Cao, J. Liu, and X. Tang, "3d object retrieval using 2d line drawing and graph based relevance reedback," in *ACM Multimedia*, 2006, pp. 105–108.

[8] P. Min, J. Chen, and T. Funkhouser, "A 2d sketch interface for a 3d model search engine," in *ACM SIGGRAPH 2002 conference abstracts and applications*, 2002, pp. 138–138.

[9] S. Ortiz, "3d searching starts to take shape search engine," *Computer*, vol. 37, no. 8, pp. 24–26, 2004.

[10] P. Company, M. Contero, J. Conesa, and A. Piquer, "An optimisation-based reconstruction engine for 3d modelling by sketching," *Computers & Graphics*, vol. 28, no. 6, pp. 955–979, 2004.

[11] Y. Leclerc and M. Fischler, "An optimization-based approach to the interpretation of single line drawings as 3d wire frames," *Int'l Journal of Computer Vision*, vol. 9, no. 2, pp. 113–136, 1992.

[12] J. Liu, L. Cao, Z. Li, and X. Tang, "Plane-based optimization for 3d object reconstruction from single line drawings," *IEEE Trans. PAMI*, vol. 30, no. 2, pp. 315–327, 2008.

[13] J. Liu, Y. Chen, and X. Tang, "Decomposition of complex line drawings with hidden lines for 3d planar-faced manifold object reconstruction," *IEEE Trans. PAMI*, vol. 33, no. 1, pp. 3–15, 2011.

[14] T. Marill, "Emulating the human interpretation of line-drawings as three-dimensional objects," *Int't Journal of Computer Vision*, vol. 6, no. 2, pp. 147–161, 1991.

[15] M. Cooper, *Line Drawing Interpretation*. Springer, 2008.

[16] G. Johnson, M. D. Gross, J. Hong, and E. Y. Do, "Computational support for sketching in design: A review," *Foundations and Trends in Human-Computer Interaction*, vol. 2, no. 1, pp. 1–93, 2009.

[17] W. Wang and G. Grinstein, "A survey of 3d solid reconstruction from 2d projection line drawings," in *Computer Graphics Forum*, vol. 12, no. 2, 1993, pp. 137–158.

[18] M. Cooper, "The interpretation of line drawings with contrast failure and shadows," *Int'l Journal of Computer Vision*, vol. 43, no. 2, pp. 75–97, 2001.

[19] K. Sugihara, *Machine interpretation of line drawings*. MIT press Cambridge, 1986.

[20] W. Whiteley, "A matroid on hypergraphs, with applications in scene analysis and geometry," *Discrete & Computational Geometry*, vol. 4, no. 1, pp. 75–95, 1989.

[21] S. Kyratzi and N. Sapidis, "From sketch to solid: an algebraic cross-section criterion for the realizability of a wireframe sketch," *Computing*, vol. 86, no. 2-3, pp. 219–234, 2009.

[22] H. Li, "nd polyhedral scene reconstruction from single 2d line drawing by local propagation," *Automated Deduction in Geometry*, pp. 169–197, 2006.

[23] L. Ros and F. Thomas, "Overcoming superstrictness in line drawing interpretation," *IEEE Trans. PAMI*, vol. 24, no. 4, pp. 456–466, 2002.

[24] E. Brown and P. Wang, "3d object recovery from 2d images: A new approach," in *SPIE Proc. Robtics and Computer Vision*, vol. 2904, 1996, pp. 138–145.

[25] K. Shoji, K. Kato, and F. Toyama, "3-d interpretation of single line drawings based on entropy minimization principle," in *CVPR*, 2001.

[26] M. C. Leong, Y. T. Lee, and F. Fang, "A search-and-validate method for face identification from single line drawings," *IEEE Trans. PAMI*, vol. 35, no. 11, pp. 2576–2591, 2013.

[27] J. Liu and Y. T. Lee, "Graph-based method for face identification from a single 2d line drawing," *IEEE Trans. PAMI*, vol. 23, no. 10, pp. 1106–1119, 2001.

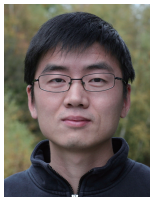
[28] J. Liu, Y. T. Lee, and W. K. Cham, "Identifying faces in a 2d line drawing representing a manifold object," *IEEE Trans. PAMI*, vol. 24, no. 12, pp. 1579–1593, 2002.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TVCG.2014.2354039, IEEE Transactions on Visualization and Computer Graphics

[29] J. Liu and X. Tang, "Evolutionary search for faces from line drawings," *IEEE Trans. PAMI*, vol. 27, no. 6, pp. 861–872, 2005.
[30] M. Shpitalni and H. Lipson, "Identification of faces in a 2d line drawing projection of a wireframe object," *IEEE Trans. PAMI*, vol. 18, no. 10, pp. 1000–1012, 1996.
[31] P. A. C. Varley and P. Company, "A new algorithm for finding faces in wireframes," *Computer-Aided Design*, vol. 42, no. 4, pp. 279–309, 2010.
[32] A. Turner, D. Chapman, and A. Penn, "Sketching space," *Computers & Graphics*, vol. 24, no. 6, pp. 869–879, 2000.
[33] A. Shesh and B. Chen, "Smartpaper: An interactive and user friendly sketching system," in *Computer Graphics Forum*, vol. 23, no. 3, 2004, pp. 301–310.
[34] Y. T. Lee and F. Fang, "A new hybrid method for 3d object recovery from 2d drawings and its validation against the cubic corner method and the optimisation-based method," *Computer-Aided Design*, vol. 44, no. 11, pp. 1090–1102, 2012.
[35] M. Armstrong, *Basic Topology*. Springer, 1983.
[36] S. Agarwal and J. Waggenspack, "Decomposition method for extracting face topologies from wireframe models," *Computer-Aided Design*, vol. 24, no. 3, pp. 123–140, 1992.
[37] Y. T. Lee and F. Fang, "3d reconstruction of polyhedral objects from single parallel projections using cubic corner," *Computer-Aided Design*, vol. 43, no. 8, pp. 1025–1034, 2011.
[38] T. Xue, J. Liu, and X. Tang, "Example-based 3d object reconstruction from line drawings," in *CVPR*, 2012, pp. 302–309.
[39] F. C. Langbein, A. D. Marshall, and R. R. Martin, "Choosing consistent constraints for beautification of reverse engineered geometric models," *Computer-Aided Design*, vol. 36, no. 3, pp. 261–278, 2004.
[40] C. Zou and J. Liu, "Optimization-based precise reconstruction of 3d objects from line drawings," *Journal of Computer-Aided Design & Computer Graphics (in chinese)*, vol. 24, no. 12, pp. 1585–1591, 2012.
[41] H. L. Zou and Y. T. Lee, "Constraint-based beautification and dimensioning of 3d polyhedral models reconstructed from 2d sketches," *Computer-Aided Design*, vol. 39, no. 11, pp. 1025–1036, 2007.

Changqing Zou received the B.E. degree from Harbin Institute of Technology, China, in 2005, and the M.E. degree from Institute of Remote Sensing and Digital Earth, Chinese Academy of Sciences, China, in 2008. He has been working toward the PHD degree at the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China, from 2011. He is also currently an Assistant Professor in the Department of Physics and Electronic Information Science, Hengyang Normal University, China. His interests include computer vision and computer graphics.

Shifeng Chen received the B.E. degree from the University of Science and Technology of China, Hefei, in 2002, the M.E. degree from City University of Hong Kong, Hong Kong, in 2005, and the Ph.D. Degree from The Chinese University of Hong Kong, Hong Kong, in 2008. He is now an Associate Professor in the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, China. His research interests include multimedia, image processing and computer vision.



Hongbo Fu is currently an Associate Professor in the School of Creative Media, City University of Hong Kong. Before joining CityU, he had postdoctoral research trainings at the Imager Lab, University of British Columbia, Canada and the Department of Computer Graphics, Max-Planck-Institut Informatik, Germany. He received the PhD degree in computer science from the Hong Kong University of Science and Technology in 2007 and the BS degree in information sciences from Peking University, China, in 2002. His primary research interests fall in the field of computer graphics with an emphasis on digital geometry processing. He was the Organization Chair for Pacific Graphics 2012, Program Co-chair for CAD/Graphics 2013, and the Emerging Technologies Program Chair for SIGGRAPH Asia 2013. Currently he is the Workshop & Co-located Events Chair for SIGGRAPH Asia 2014.

Jianzhuang Liu received the Ph.D. degree in computer vision from The Chinese University of Hong Kong, Hong Kong, in 1997. From 1998 to 2000, he was a research fellow with Nanyang Technological University, Singapore. From 2000 to 2012, he was a postdoctoral fellow, then an assistant professor, and then an adjunct associate professor with The Chinese University of Hong Kong. He joined Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, as a professor, in 2011. He is currently a chief scientist with Huawei Technologies Co. Ltd., Shenzhen, China. He has published more than 100 papers, most of which are in prestigious journals and conferences in computer science. His research interests include computer vision, image processing, machine learning, multimedia, and graphics.