Computers & Graphics 37 (2013) 574–581

Contents lists available at SciVerse ScienceDirect

Computers & Graphics

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

SMI 2013

Illustrating the disassembly of 3D models

Jianwei Guo [a], Dong-Ming Yan [a,b,*], Er Li [d], Weiming Dong [a], Peter Wonka [b,c], Xiaopeng Zhang [a]

[a] NLPR-LIAMA, Institute of Automation, Chinese Academy of Sciences, China
[b] GMSV Center, King Abdullah University of Science and Technology, Saudi Arabia
[c] Department of Computer Science and Engineering, Arizona State University, USA
[d] State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences, China

ARTICLE INFO

Article history:
Received 12 March 2013
Received in revised form
28 May 2013
Accepted 31 May 2013
Available online 11 June 2013

Keywords:
Disassembly illustration
Shape analysis
Visualization

ABSTRACT

We present a framework for the automatic disassembly of 3D man-made models and the illustration of the disassembly process. Given an assembled 3D model, we first analyze the individual parts using sharp edge loops and extract the contact faces between each pair of neighboring parts. The contact faces are then used to compute the possible moving directions of each part. We then present a simple algorithm for clustering the sets of the individual parts into meaningful sub-assemblies, which can be used for a hierarchical decomposition. We take the stability of sub-assemblies into account during the decomposition process by considering the upright orientation of the input models. Our framework also provides a user-friendly interface to enable the superimposition of the constraints for the decomposition. Finally, we visualize the disassembly process by generating an animated sequence. The experiments demonstrate that our framework works well for a variety of complex models.

1. Introduction

The disassembly of 3D models is a process that involves the decomposition of a given assembled model into individual parts, without breaking any part. Designing a disassembly is also an important concept of Green Design. As discussed by Knoth et al. [1], for a product, "almost all end-of-life possibilities – upgrade, reuse, recycling of materials – require some form of disassembly." Disassembly can provide an optimal solution for repair and maintenance, such as for the replacement of a model's damaged parts or performing equipment maintenance. Furthermore, product design for assembly can be facilitated by performing disassembly analysis. Disassembly analysis either leads to a feasible assembly sequence (by reversing the disassembly steps) or alerts the designers of an assembly problem in the product design that has to be fixed [2].

As mentioned above, disassembly has numerous applications in the domain of industrial engineering. However, currently disassembly remains mainly a manual process [3]. Complex 3D objects are typically composed of numerous components, and understanding the spatial relationships of parts within the assembly is difficult. Thus, skilled humans are needed to generate disassembly sequences with specific tools.

To the best of our knowledge, research on disassembly in computer graphics is just at the beginning. Existing works focus on designing assembly sequences [4] and generating exploded view diagrams [5,6]. In these papers, the authors use blocking constraints to derive moving directions and the ordering of individual parts. We build on these initial works and improve them for our purposes. First, we noticed that solving the disassembly problem based only on blocking constraints is insufficient. Stability and the needs of illustration must also be considered. Second, previous works also exhibit a number of limitations, for example, the moving directions of one part contain only six directions, and the creation of part hierarchies for an assembly is performed manually.

In this paper, we present a framework for the automatic disassembly of 3D models and provide several visualization tools to illustrate the process. In our approach, we use the geometric properties of individual parts to retrieve their motion parameters (e.g., translational axis, separation direction and distance) and compute the hierarchical structure of the model. We observed that using the blocking constraint to determine the disassembly order for sub-assemblies might be unstable. To overcome this issue, we provide an interface that enables users to specify some fixed parts (e.g., bottom grey parts in Fig. 1). We then use this information to perform a top-down and constrained disassembly. Finally, we develop visualization methods to illustrate the disassembly process automatically.

The contributions of our work include: (1) an improved method for the calculation of the blocking relationships between parts based on their geometric properties; (2) an automatic

* Corresponding author at: NLPR-LIAMA, Institute of Automation, Chinese Academy of Sciences, China.
E-mail addresses: jianwei.guo@nlpr.ia.ac.cn (J. Guo), yandongming@gmail.com (D.-M. Yan), lier@ios.ac.cn (E. Li), weiming.dong@ia.ac.cn (W. Dong), pwonka@gmail.com (P. Wonka), xiaopeng.zhang@ia.ac.cn (X. Zhang).

0097-8493/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cag.2013.05.020

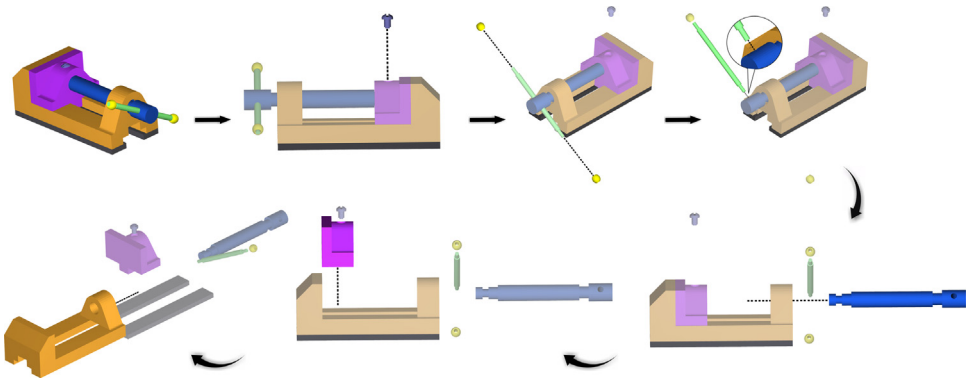

Fig. 1. Disassembly sequence of the bench vice model generated by our framework.

technique that clusters individual parts into groups; and (3) new constraints for stable disassembly.

1.1. Related work

Although assembly planning has been a classic problem for a long time, studies on disassembly only became popular in recent decades and in the area of robotics [1–3]. The input of their system includes the geometrical models of products and the different features of parts stored in a database (e.g., the type of each part and the style of contacts between parts). Based on such information, a disassembly theory is applied to compute the optimal disassembly sequence from all possible candidate solutions. However, these plans are only used by robotic machines and do not present any ways to depict disassembly operations for humans.

Assembly/disassembly planning. In the field of computer graphics, many works have focused on assembly planning and visualization for 3D models [7,4,8,9]. For example, Agrawala et al. [4] combine the planning with presentation techniques to design effective step-by-step assembly instructions by automatically determining the separation order and directions. The limitation of their work is that the direction to which one part can move only contains six principal directions (consider the green oblique cylinder in Fig. 1). The work of Xin et al. [9] illustrates the puzzle assembly or disassembly process, in which they aim to design and create burr puzzles from 3D models. Song et al. [8] develop a constructive approach for generating recursive interlocking puzzles. However, only one specific sequence of assembly and disassembly exists for such an interlocking puzzle, making it unsuitable for general mechanical assemblies.

Exploded views. The exploded view diagram is a powerful tool used to convey the spatial relationships between components within complex objects. To generate such diagrams, illustrators design many methods to explode one part from others with the proper direction and distance. Agrawala et al. [4] and Li et al. [6] use blocking information to infer the exploded directions. The latter also includes an interface that enables users to view the parts of interest. However, the parts of their input models are organized into a hierarchy manually beforehand. Li et al. [5] present a system to allow interactive cutaway illustrations for complex 3D models. Tatzgern et al. [10] generate compact explosion diagrams by finding similar sub-assemblies and only rendering the representatives. Most of these works generate traditional static diagrams or provide an interactive way that requires user's participation.

Shape analysis. Shape analysis has been demonstrated to help deduce the characteristic properties of man-made objects. Recently, understanding the higher-level representations of 3D models has received considerable interest (see survey [11]), including symmetry detection [12–14], upright orientation inference [15] and shape abstractions [16]. Benefitting from these techniques, a wide spectrum of applications are spawned, such as shape manipulation [17–19], shape synthesis [20], motion analysis [21–24] and computational furniture design [25]. In addition, many works focus on the analysis of individual parts within an input model, which are similar to our framework. Mitra et al. [22] analyze the interactions and motions of mechanical parts based on recent advances in geometric shape analysis. Jain et al. [20] present a system to interpolate new shapes from two database shapes by decomposing input shapes and recombining individual parts according to constraints deduced through shape analysis.

2. Principles for disassembly

We extend the conventions from traditional illustrations [26,27,6] for mechanical assemblies to principles that are suitable for the disassembly process. Although disassembly is distinct from assembly, we can still obtain some heuristics from the assembly process [4], because the sequence of assembly is usually the reverse of that of disassembly.

Hierarchy of parts. Generally, a mechanical assembly can be divided into several sub-assemblies. During the disassembly process, if a sub-assembly can be removed entirely, people prefer that the parts within this sub-assembly are disassembled later after the sub-assembly is separated.

Stability. In actual disassembly studies, stability analysis is used to investigate whether sub-assemblies can be collapsed by gravity. Some parts usually perform a supporting function in an assembly. Such holding parts should be fixed, and unstable sub-assemblies should be pruned out early in the disassembly process.

Step-by-step operations. Disassembly sequences are listings of subsequent disassembly operations. Showing all these disassembly operations in a single static diagram hinders users from identifying the order in which parts are removed. On the other hand, depicting the disassembly process step-by-step using animation is easy to understand and follow.

Visibility. Visibility serves an important function in visualization. The sub-assembly being removed must be visible to users. A notable exception is that not all the parts in a symmetry group can be seen simultaneously.

Non-destructive disassembly. Depending on the goals, disassembly has two main types [28]. One is “destructive disassembly”, in which a part is removed from a previously assembled product by destroying or damaging some other parts of the product. The other is “non-destructive disassembly”, in which each of the parts can be removed without affecting any of the others. Our work focuses on non-destructive disassembly for the purpose of reuse, such as equipment maintenance and recycling of materials.

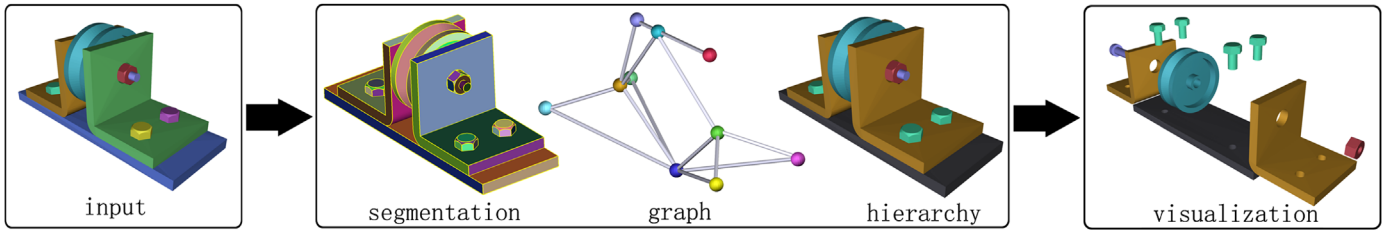

Fig. 2. Flowchart of our framework. Given an assembled 3D model, our framework first analyzes the geometry of each part and retrieves the separated direction and distance information. The parts are then automatically grouped into sub-components. Finally, we provide various visualization tools to illustrate the disassembly process.

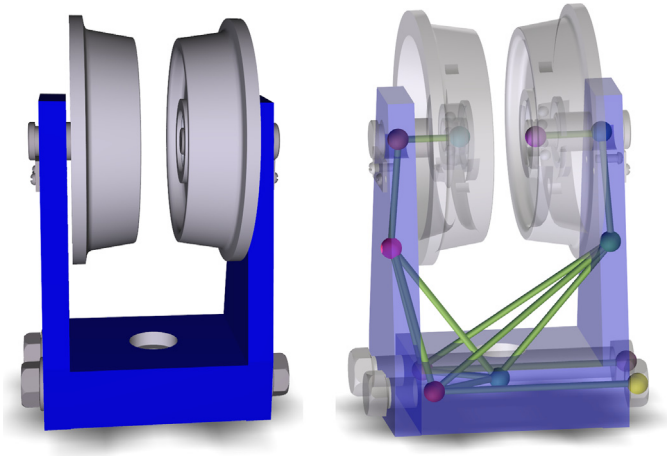

Fig. 3. Left: Trolley assembly. Right: Contact graph. Nodes represent the parts, and edges store the contacting and blocking information. Each edge is composed of two opposite directed edges.

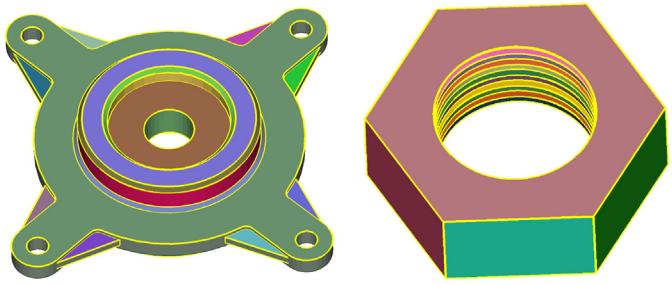

Fig. 4. Segmentation results. Sharp edge loops are shown in yellow. Segmented patches are displayed in different colors. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

3. Overview

Our framework has two main components: shape analysis and disassembly illustration. Given an input assembled model, our system first performs shape analysis. In this step, each part of the input model is segmented and a contact graph is built for the parts. The constrained disassembly sequences are computed based on the contact graph. Our system then uses various visualization techniques (e.g., animation and viewpoint selection, etc.) to illustrate the disassembly process. Fig. 2 shows the flowchart of our system.

4. Disassembly computation

In this section, we present the framework for analyzing and generating the disassembly sequences of input models. The input of our framework is a 3D geometric model M, which consists a set of interlocked individual parts $\{p_i\}_{i=1}^{n}$. Each part p_i is assumed to be clean so that it is 2-manifold. In the remainder of this section, we first introduce an improved version of the contact graph, which is used for determining the disassembly sequence (Section 4.1). Then, we present a simple algorithm to group the individual parts into sub-assemblies, which can be used for hierarchical decomposition (Section 4.2).

4.1. Contact graph construction

To represent the blocking relationships, Agrawala et al. [4] define the *directional blocking graph* which stores the separating directions of parts. Li et al. [6] extend the directional blocking graph to the *explosion graph*. However, auxiliary data structures are needed in their systems to encode the low-level computations, including whether parts touch or block any others. Furthermore, they use a local translational blocking (LTB) cone [29] to calculate the blocking directions. It is complicated to compute the LTB cone and partition the translational motion sphere.

Based on the observation that the contact faces strongly restrict the directions along which one part is to be removed, we generalize the *contact graph* of [22] by encoding the contact faces into the graph. A contact graph is denoted by $G=\{N,E\}$, where N is the set of nodes $\{n_i\}$ and E is the set of the directed edges $\{e_{ij}\}$ connecting the nodes in contact. Fig. 3 illustrates the contact graph used in our framework, which is analogous to the half-edge data structure. Each node n_i corresponds to a part p_i, whereas each directed edge e_{ij} stores the contact and blocking information of p_i with respect to p_j. To construct the contact graph, we first analyze the geometrical properties of individual parts and detect the contact relationship between parts. We then use the contact faces to infer the separation parameters of the parts.

Part segmentation. We follow the approach of Mitra et al. [22] to segment an individual part into patches by extracting sharp edge loops (see Fig. 4). For each patch, we fit a simple low degree algebraic surface (including planes, spheres, cones and cylinders) to detect the type of the patch. We then detect planar circular loops, which are always incident to separating axes. Furthermore, we cluster the circular loops into different groups according to their axes and consider these clustered axes as the potential set D of the separating directions.

Contact detection. Given the input model M, we first compute the shortest distance between each pair of parts. If the distance is smaller than a threshold ϵ (we use $\epsilon = l_{diag} \times 10^{-4}$, where l_{diag} is the diagonal length of the bounding box of the input model), we mark the pair of parts as being in contact. We then add two edges in opposite directions between the corresponding nodes and store the contact faces in the half-edges. For example, if parts p_i and p_j are in contact, the edge e_{ij} stores the contact faces that belong to part p_i, and its opposite edge e_{ji} stores the contact faces of p_j.

Surface fitting. For each part, the contact faces between this part and others have been stored in the corresponding edges. Notably,

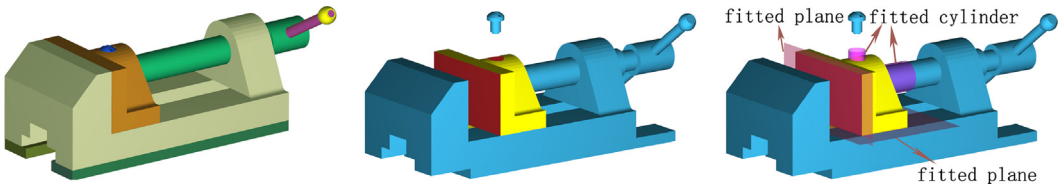

Fig. 5. Fitted surfaces for a part. Left: Input geometry model of a bench vice assembly. Middle: For the yellow part, the red regions denote the contact faces with respect to other parts. We translated one of the contact parts slightly for a better view. Right: Four fitted planes (two of them are in the unseen slot position) and two cylinders to the contact faces. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

these contact faces can also be segmented into small regions by using sharp edge features. We call such a region as *contact region*. However, the faces in a contact region may have different normals. Determining which normal direction can be used to compute the separation direction is difficult. To mitigate this problem, we fit the contact regions with planes or cylinders (see Fig. 5). The fitted surfaces and their normals are also stored in the corresponding half-edges. Moreover, we identify a part as a cylindrical part if all of its fitted surfaces are cylinders.

Determine separation direction. Recall that in the segmentation step, we have computed the potential set D of removal directions and then fitted contact surfaces with their normals for each part p_i. We cut the direction $\mathbf{d}_i \in D$ from the set D if a fitted plane with normal direction $\mathbf{n}_i$ exists and if $\mathbf{d}_i \cdot \mathbf{n}_i \approx 1$. Then the new set D' becomes the set of separating directions for part p_i. In Fig. 5, the yellow part p has only two potential separation directions. The method described here is suitable for non-cylindrical parts. For a cylindrical part, it usually moves only along the two directions of the axis of this cylinder.

Having obtained the set D' of unblocked directions along which a part p_i can be removed, we follow the iterative approach of Li et al. [6] to choose the separation direction from D' and to determine the disassembly order and separation distance by computing the minimum distance required to escape the bounding box of the unremoved parts in contact with p_i.

4.2. Part hierarchy

Most complex 3D assemblies exhibit some hierarchical structures. Disassembling the sub-assemblies entirely before removing their individual parts is reasonable. However, to the best of our knowledge, no automatic algorithm can partition an assembly into groups of sub-assemblies. Given this interesting problem, we conduct a user study that asks 43 participants to partition the assemblies used in this paper so that we may understand how people cluster/segment the assemblies. One-third of the participants (14) have a mechanical engineering background, and the others (29) are people without such a background. Given the photograph of a model in which each part has been labeled with a number, the participants are asked to partition the model and record the reasons for doing so. Based on this user study, we summarized three principles that can be used to guide the partition.

Symmetry. The symmetric parts are always disassembled in the same manner. To simplify the disassembly sequence, we collect the symmetric parts into one group and disassemble them at the same time. Fig. 6 (left) shows a result of symmetry detection.

Coaxial. Coaxial parts usually have the same separated direction. Given a pair of parts, if their axes are contained by a single line, we classify the two parts as coaxial (see Fig. 6 (middle)). The coaxial parts are clustered into the same groups.

Contact relationship. It is obvious that the disjoint parts can be easily separated. Rather than using this principle to partition the assemblies, we apply it to verify whether a part is in contact with any others in the same group. If a part is not in contact with others, we remove it from its group.

In this paper, we partition the assemblies in two steps. We first detect the symmetric and coaxial parts to establish an initial partition. Symmetric parts are detected by computing the distance between two vectors, the three elements of which are the eigenvalues of principal component analysis (PCA), which are computed using all the vertices (including internal vertices) of parts. On the other hand, we use the definition of coaxial relationship to detect the coaxial parts. Next, for each group C, we eliminate the part p from C if we cannot find at least one part $q \in C$ that is in contact with p or is symmetric to p. We traverse all groups to determine the final partition. However, deriving a completely correct partition is difficult, because of lacking the functional and semantic properties of parts. We also provide a user interface that enables users to correct the misclassifications.

To determine the directions along which one group C can be removed, we compute the intersection between the directions of every part p in this group:

$$Dirs(C) = \bigcap_{\forall p \in C} Dirs(p) \tag{1}$$

Here, the function *Dirs* represents the set of possible removal directions for a part or a group.

5. Disassembly with constraints

We have demonstrated that the disassembly sequences can be efficiently generated by blocking constraint, such that no parts will

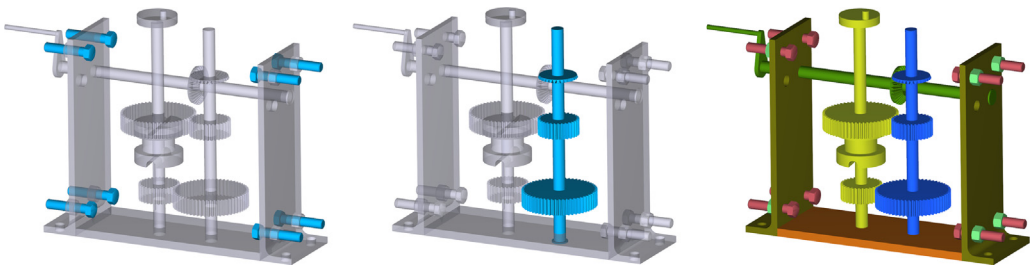

Fig. 6. Symmetric (left) and coaxial (middle) parts are highlighted by a cyan color. We render other parts in a semi-transparent mode. (Right) Grouping result. Each group is drawn using the same color. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

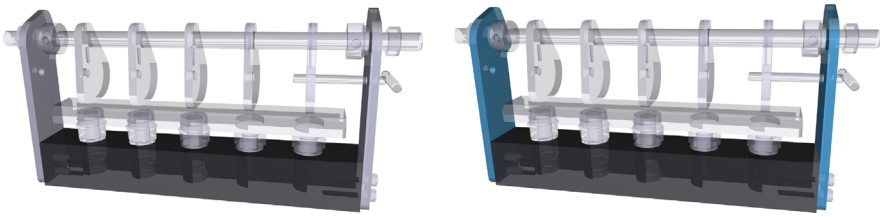

Fig. 7. Examples of base part (left) and fixed parts (right).

collide with others. However, in many cases, the generated disassembly operations are geometrically feasible but are not in line with the actual situation. For the F15 model (see Fig. 8), the left and right frames are first to be disassembled before the gears and axles in the middle are removed. The middle parts will then be collapsed by gravity because of the lack of holding devices. Thus, additional constraints are needed to implement more realistic disassembly. In this section, we describe two new constraints which take the stability principle into account.

5.1. Top-down disassembly

When we stack objects in our daily life, we naturally proceed from bottom to top under the influence of gravity. Similarly, the lower parts are usually assumed to support the upper parts in assemblies. On the other hand, a *base part* always lies on the ground plane (see Fig. 7) to support the whole object. The base part can be automatically detected by adapting the upright orientation approaches [15]. However, in our framework, we provide a simple user interface to enable a user to specify the base part and its upright orientation. Once the base part is specified, we perform a top-down approach to disassemble the input models.

In Section 4.1, we get a set of directions for the base part by clustering the axes of fitted circles. We rank these directions based on the number of elements in their corresponding clusters. The direction of the cluster with the most elements is selected as upright orientation of the model. We then find the global lowest vertex $\mathbf{v}$ of the input model along the upright direction. The center $\mathbf{c}$ of each group and the lowest vertex $\mathbf{v}$ form a vector $\overrightarrow{\mathbf{vc}}$. We compute the height of each group from the ground plane as the length of projection of $\overrightarrow{\mathbf{vc}}$ along the upright direction. Each time we disassemble the uppermost group if it can be removed.

If the model has no obvious base part, e.g., the stool model in Fig. 9, we provide a tool for the user to specify the upright direction.

5.2. Disassembly with fixtures

A fixed part is defined as the unchanged part through the whole disassembly sequence (see Fig. 7). In the real disassembly process, fixed parts are used to keep the intermediate sub-assemblies stable. In our framework, we provide a friendly interface for users to specify the fixed parts. Given that fixed parts cannot be removed, we first traverse each group to delete the fixed parts from this group and then place all fixed parts into one group. Next, we recompute the disassembly orders of parts using both the blocking and fixed constraints.

It should be noted that once we identify the fixed parts, it might conflict with the part hierarchies and sub-assemblies. Consider the three parts (the green axle and two yellow gears) in the lower position of the F15 model (see Fig. 8(a)), they are partitioned into one group before the user specifies the fixed parts. When we specify the left and right frames as fixed parts, the three parts cannot be removed together. In this case, we have to break the group into two groups, with only one group containing the axle while the other containing two gears.

6. Disassembly illustration

To allow the users to better understand the disassembly process, we develop several visualization tools to illustrate this process.

6.1. Viewpoint selection

In the disassembly process, the currently removed group must be visible. Meanwhile, the parts that are not yet removed should also be visible to provide context for the ongoing disassembly operation. To achieve these goals, we build a regular icosahedron at the center of bounding box of the model. Then twenty cameras with a perspective toward the center of the model are placed at the vertices of the icosahedron. The size of the icosahedron is large enough to ensure that the entire model can be seen from each viewpoint. To select the best viewpoint for the currently removed group, we adopt the viewpoint Kullback–Leibler distance (VKL) [30] in our framework. Given a viewpoint v, we define the VKL as follows:

$$KL_v = \sum_{i=1}^{N_c} \frac{a_i}{a_t} \log \frac{a_i/a_t}{A_i/A_t} \tag{2}$$

where N_c is the number of groups of parts, a_i is the projected area of group i, $a_t = \sum_{i=1}^{N_c} a_i$, A_i is the actual area of group i and $A_t = \sum_{i=1}^{N_c} A_i$ is the total area of the model. A_i can be computed directly from the geometry of the parts. To compute the projected area a_i, we render the group i with a single color and count the pixels of this color in the frame buffer as a_i.

6.2. Animation

Animation is the most natural approach to illustrate the disassembly process. In our framework, we generate a sequence of frames that illustrates every operation in the disassembly process. After obtaining the hierarchy of parts, we disassemble the groups iteratively. Starting from the first group, we disassemble this group from the rest of the assembly in a certain separated direction, a special camera position and a special color. The separation distance of this group is set as the minimum distance to escape the bounding box of the unremoved parts in contact with this group. Then, each part in this group is removed separately. We render and repeat this process until all the groups have been removed and placed in their final positions.

6.3. Rendering and highlighting

To distinguish the different groups better through the disassembly process, we employ a non-photorealistic rendering style where each group is rendered by a single color. Moreover, for each disassembly operation, we highlight the currently disassembled

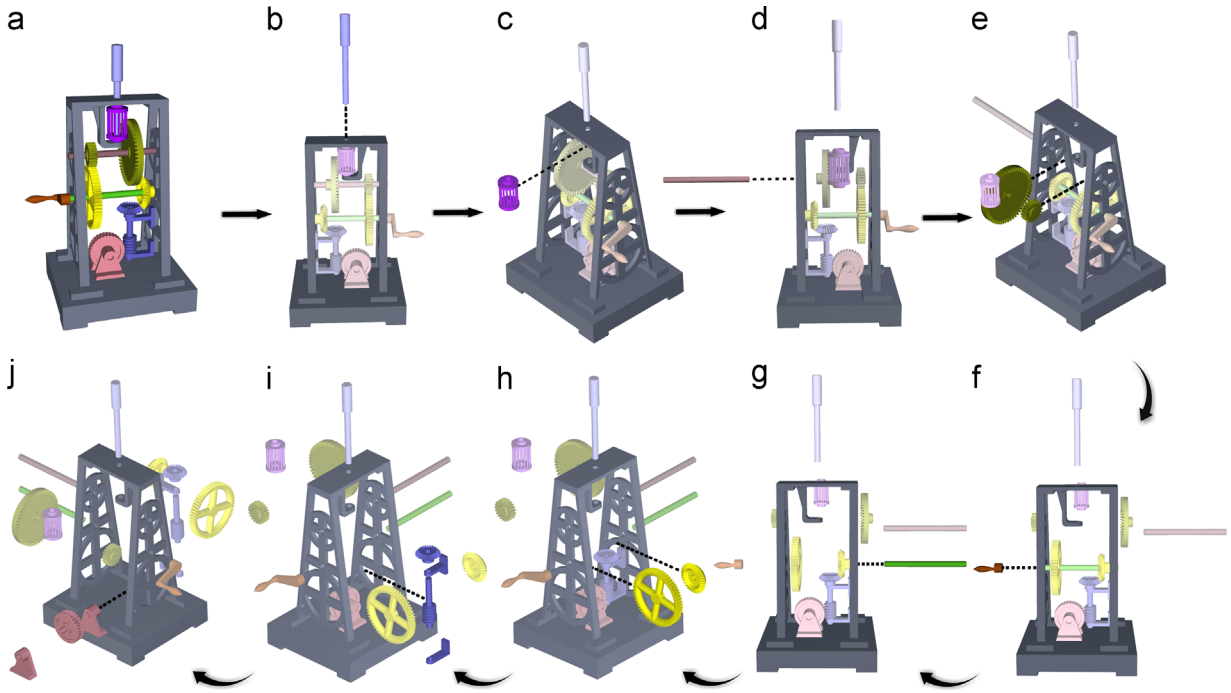

Fig. 8. (a) Input geometry model of F15, the individual parts of which have been clustered into several different groups. (b)–(j) Illustrating the step-by-step disassembly operations of this model. In this sequence, we show our visualization methods of rendering, highlighting and camera movement.

parts and render the other parts in a desaturated manner, which is similar to [22]. Figs. 1 and 8 give two such examples.

7. Results

In this section, we present selected disassembly results using our framework. We have tested a wide range of 3D models, from mechanical assemblies to furniture models, as shown in Figs. 1, 8 and 9. Our framework is implemented in C++ using the open-source platform *Graphite*.[1] We use the open-source library PQP [31] for computing the distance between parts, and we adapt the code of [32] for surface fitting. The experimental results shown in this paper are conducted on an Intel Dual Core 2.40 GHz CPU with 4 GB memory and a 64-bit Windows 7 operating system.

All the disassembly results generated by our framework followed the principles proposed in Section 2. The base part and fixed parts are specified by users, except in the stool model. We use animated illustrations to visualize the disassembly sequences, which are shown in our accompanying video. The disassembly processes in Figs. 1 and 8 are illustrated by generating disassembly instructions step-by-step, which shows how each part is to be separated from the remainder of the assemblies. In Fig. 9, we demonstrate three static figures for each model by employing a non-photorealistic rendering style. First, each part of an input model is identified and visualized with a special color. In the second figure, we cluster the individual parts into meaningful sub-assemblies based on the algorithm described in Section 4.2. Finally, we generate the exploded views in which each part is in its final separated position. The view angles are automatically selected.

Comparisons. Notably, using the upright direction information and fixed parts can improve the stability of the disassembly process, compared with previous works. For example, for the middle part of the clamp in Fig. 9(c), the method of Agralawa et al. [4] or Li et al. [6] may disassemble the dark green fastener first, as this fastener can be removed without violating the blocking constraints. Consequently, without any holding device, the parts above this fastener will collapse because of gravity. In our framework, we disassemble this fastener later after the parts above it using a top-down approach. In addition, we allow users to specify the fixed parts and keep such parts unchanged throughout the disassembly process. This technique also guarantees the stability effectively, as shown in Fig. 8.

Furthermore, while the parts of a model in previous works [4,6] can only move along the three principal axes of the model, our method enables each part to be separated in any reasonable direction. For the green part in Fig. 1, we first extract the contact faces between this part and others. We then fit three cylinders to these contact faces, and thus we identify this part as a cylinder part. As mentioned earlier in Section 4.1, the removal directions of this part are the two directions of this cylinder's axis.

Performance. We evaluate the performance of our method in Table 1, including the statistics of each model (e.g., the number of parts) and the computation time for each stage. As it shows, the time mainly depends on the part analysis and the contact graph computation, which are related to the mesh size and the number of parts respectively. It also demonstrates that our method can deal with complex models such as the Aparelho Divisor model, which is composed of as many as 81 parts.

8. Conclusions and future work

In this paper, we present an approach for the disassembling complex 3D models and visualizing the disassembly processes. The key techniques of our work include an improved version of the contact graph that is used for determining the removal directions of parts based on shape analysis, an automated method to partition the input complex model into groups of sub-assemblies and two constraints for stable disassembly.

[1] http://alice.loria.fr/WIKI/index.php/Graphite.

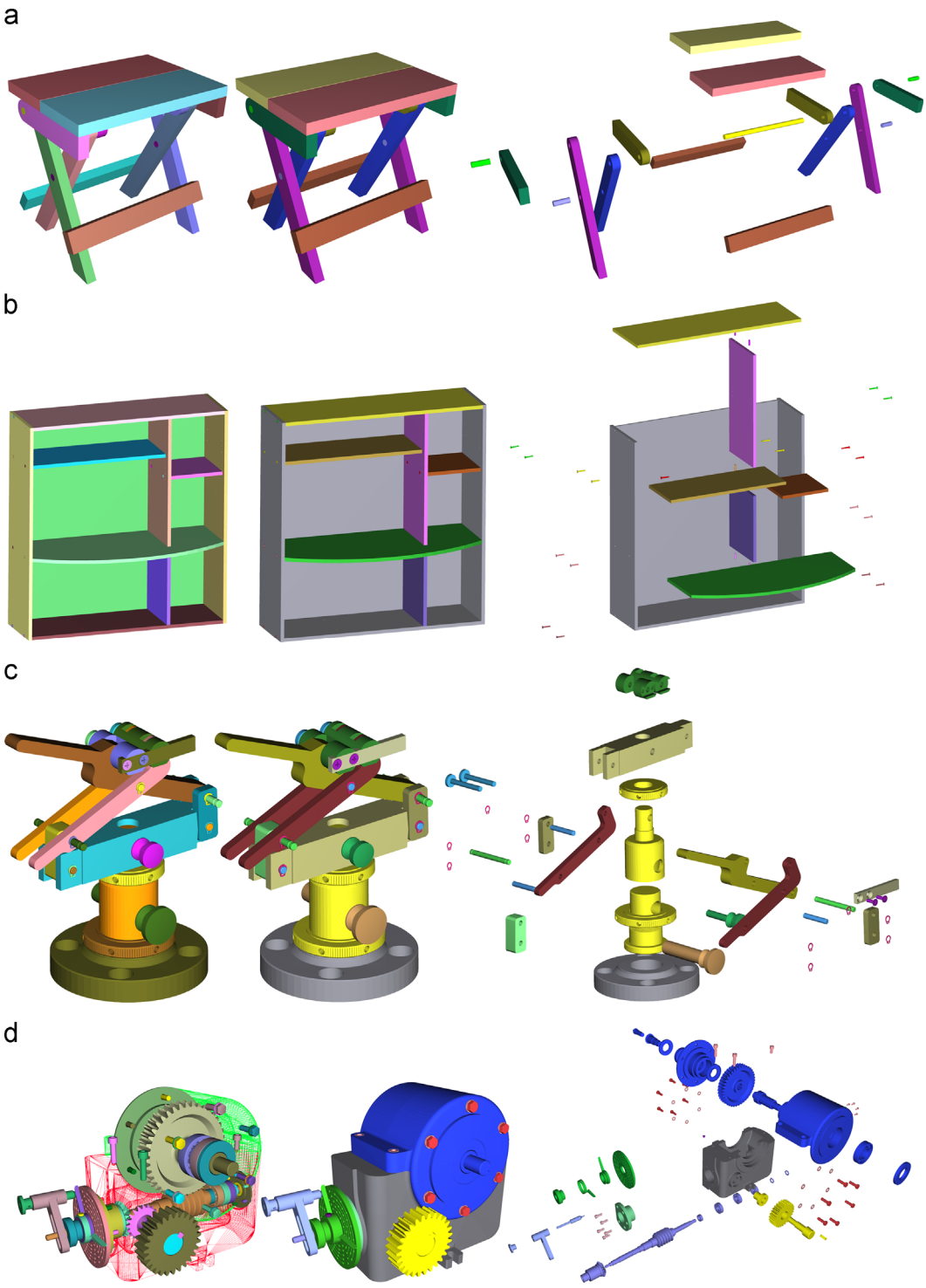

Fig. 9. A gallery of examples generated by our framework: (a) Stool, (b) Cabinet, (c) Clamp, and (d) Aparelho Divisor. For each model, left: shows its distinct parts in a single color. Middle: parts are partitioned into different groups. Right: displays disassembly result with each part in its final position. To display the internal structure of Aparelho Divisor model, we render its containers in a wire-frame mode. (For interpretation of the references to color in this figure caption, the reader is referred to the web version of this paper.)

The current framework has several limitations. First, our system can only handle input models with clean geometry. Second, similar to most works on generating assembly sequences and exploded views, all parts can be removed only via single linear translations. We cannot deal with complex moving paths, e.g., a screw usually has to be removed by a helix motion, and removing a nut from inside a car engine follows a sequence of linear translations. Third, user assistance is required to specify the base part to compute the upright direction, which prevents our framework from being fully automatic. We would like to address these issues in our future work.

Acknowledgments

We would like to thank the anonymous reviewers for their detailed comments. The F15 model in Fig. 8 was downloaded from the “Antique Mechanism Teaching Models Digital Museum” of

Table 1
Performance statistics. Here N_t is the number of triangles in the input geometry model, N_p is the parts number, and N_g is the number of partitioned groups. T_p, T_c and T_h respectively represent the cost time for part analysis, computing the contact graph and computing the part hierarchy. The total time T_{total} is listed on the right.

Model	Model statistics			Run time (s)			
	N_t (K)	N_p	N_g	T_p	T_c	T_h	T_{total}
Bench vice	15	9	7	1.49	5.33	0.109	6.93
Trolley	23	25	10	1.55	10.64	0.391	12.58
F15	92	20	12	61.1	51.9	1.2	114.2
Stool	3.3	17	10	0.093	2.75	0.047	2.89
Cabinet	30	36	18	1.8	27.5	0.7	29.9
Clamp	74	36	15	13.6	106.9	1.52	122.0
Aparelho divisor	203	81	18	80.6	586.8	10.9	678.3

NCKU, and the other models were obtained from GrabCAD.[2] This work was partially funded by the National Natural Science Foundation of China (nos. 61271431, 61172104, 61271430 and 61202324), and the National Science Foundation.

References

[1] R. Knoth, M. Brandstotter, B. Kopacek, P. Kopacek, Automated disassembly of electr(on)ic equipment. In: 2002 IEEE international symposium on electronics and the environment, 2002. p. 290–4.
[2] Shyamsundar N, Gadh R. Geometric abstractions to support disassembly analysis. IIE Trans 1999;31(10):935–46.
[3] Lambert A. Disassembly sequencing: a survey. Int J Prod Res 2003;41 (16):3721–59.
[4] Agrawala M, Phan D, Heiser J, Haymaker J, Klingner J, Hanrahan P, et al. Designing effective step-by-step assembly instructions. ACM TOG (Proc SIGGRAPH) 2003;22(3):828–37.
[5] Li W, Ritter L, Agrawala M, Curless B, Salesin D. Interactive cutaway illustrations of complex 3D models. In: ACM TOG (Proceedings of the SIGGRAPH), vol. 26, 2007. p. 31:1–11.
[6] Li W, Agrawala M, Curless B, Salesin D. Automated generation of interactive 3D exploded view diagrams. In: ACM TOG (Proceedings of the SIGGRAPH), vol. 27, 2008. p. 101:1–7.
[7] Driskill E, Cohen E. Interactive design, analysis, and illustration of assemblies. In: Proceedings of the 1995 symposium on interactive 3D graphics, 1995. p. 27–34.
[8] Song P, Fu C-W, Cohen-Or D. Recursive interlocking puzzles. ACM TOG (Proc SIGGRAPH Asia) 2012;31(6):128:1–10.
[9] Xin S, Lai C, Fu C, Wong T, He Y, Cohen-Or D. Making burr puzzles from 3D models. ACM TOG (Proc SIGGRAPH) 2011;30(4):97:1–8.
[10] Tatzgern M, Kalkofen D, Schmalstieg D. Compact explosion diagrams. In: Proceedings of the NPAR, 2010. p. 17–26.
[11] Mitra NJ, Wand M, Zhang H, Cohen-Or D, Bokeloh M. Structure-aware shape processing. In: EUROGRAPHICS State-of-the-art Report, 2013.
[12] Mitra NJ, Guibas LJ, Pauly M. Partial and approximate symmetry detection for 3D geometry. ACM TOG (Proc SIGGRAPH) 2006;25(3):560–8.
[13] Wang Y, Xu K, Li J, Zhang H, Shamir A, Liu L, et al. Symmetry hierarchy of man-made objects. In: Computer graphics forum, vol. 30, 2011. p. 287–96.
[14] Mitra NJ, Pauly M, Wand M, Ceylan D. Symmetry in 3D geometry: extraction and applications. In: EUROGRAPHICS state-of-the-art report.
[15] Fu H, Cohen-Or D, Dror G, Sheffer A. Upright orientation of man-made objects. ACM TOG (Proc SIGGRAPH) 2008;27(3):42:1–7.
[16] Mehra R, Zhou Q, Long J, Sheffer A, Gooch A, Mitra NJ. Abstraction of man-made shapes. In: ACM TOG (Proceedings of the SIGGRAPH Asia), vol. 28, 2009. p. 137:1–10.
[17] Gal R, Sorkine O, Mitra NJ, Cohen-Or D. iWIRES: an analyze-and-edit approach to shape manipulation. ACM TOG (Proc SIGGRAPH) 2009;28(3):33:1–10.
[18] Xu W, Wang J, Yin K, Zhou K, Van De Panne M, Chen F, et al. Joint-aware manipulation of deformable models. In: ACM TOG (Proceedings of the SIGGRAPH), vol. 28, 2009. p. 35:1–9.
[19] Zheng Y, Fu H, Cohen-Or D, Au OK-C, Tai C-L. Component-wise controllers for structure-preserving shape manipulation. In: Computer graphics forum (Proceedings of the EUROGRAPHICS), vol. 30, 2011. p. 563–72.
[20] Jain A, Thormählen T, Ritschel T, Seidel H-P. Exploring shape variations by 3D-model decomposition and part-based recombination. Comput Graphics Forum (Proc EUROGRAPHICS) 2012;31(2):631–40.
[21] Gelfand N, Guibas L. Shape segmentation using local slippage analysis. In: Computer graphics forum (Proceedings of the SGP), 2004. p. 214–23.
[22] Mitra NJ, Yang Y-L, Yan D-M, Li W, Agrawala M. Illustrating how mechanical assemblies work. ACM TOG (Proc SIGGRAPH) 2010;29(4):58:1–58:11.
[23] Mitra NJ, Yang Y-L, Yan D-M, Li W, Agrawala M. Illustrating how mechanical assemblies work. Commun ACM 2013;56(1):106–14.
[24] Zhu L, Xu W, Snyder J, Liu Y, Wang G, Guo B. Motion-guided mechanical toy modeling. ACM TOG (Proc SIGGRAPH) Asia 2012;31(6):127:1–10.
[25] Umetani N, Igarashi T, Mitra NJ. Guided exploration of physically valid shapes for furniture design. ACM TOG (Proc SIGGRAPH) 2012;31(4):86:1–86:11.
[26] Seligmann D, Feiner S. Automated generation of intent-based 3D illustrations. In: ACM SIGGRAPH computer graphics, vol. 25, 1991. p. 123–32.
[27] Dennision JA, Johnson CD. Technical illustration: techniques and applications. Goodheart-Wilcox.
[28] Pomares J, Puente S, Torres F, Candelas F, Gil P. Virtual disassembly of products based on geometric models. Comput Ind 2004;55(1):1–14.
[29] Romney B, Godard C, Goldwasser M, Ramkumar G, et al. An efficient system for geometric assembly sequence generation and evaluation. Comput Eng 1995:699–712.
[30] Sbert M, Plemenos D, Feixas M, Gonzlez F. Viewpoint quality: measures and applications. In: Eurographics workshop on computational aesthetics in graphics, visualization and imaging, 2005. p. 185–92.
[31] Larsen E, Gottschalk S, Lin M, Manocha D. Fast distance queries with rectangular swept sphere volumes. In: Proceedings of the IEEE international conference on robotics and automation, 2000. ICRA'00, vol. 4, 2000. p. 3719–26.
[32] Yan D-M, Wang W, Liu Y, Yang Z. Variational mesh segmentation via quadric surface fitting. Comput Aided Des 2012;44(11):1072–82.

[2] http://grabcad.com/library.