Screwing Assembly Oriented Interactive Model Segmentation in HMD VR Environment

Abstract

Although different approaches of segmenting and assembling geometric models for 3D printing have been proposed, it is difficult to find any researches which investigate model segmentation and assembly in HMD (Head-Mounted Display) VR environments for 3D printing. In this work, we propose a novel and interactive segmentation method for screwing assembly in the environments to tackle this problem. Our approach divides a large model into semantic parts with a screwing interface for repeated tight assembly. With our method, non-professional users can intuitively segment models larger than a printer's workspace into several components based on a single VR Handle cut and robust Boolean operations. Specifically, after a user places the cutting interface, an automatically created bounding box of the current part is computed for subsequent Boolean segmentations, which results in multi-component semantic parts with planar cuts. Afterwards, the bolt is positioned with an improved K3M image thinning algorithm and is used for merging paired components with union and subtraction Boolean operations respectively. Moreover, we introduce a Swept Boolean based rotation collision detection and location method to guarantee a collision-free screwing assembly. Experiments show that our approach provides a new interactive multi-component semantic segmentation tool, which supports not only repeated installation and disassembly but also tight and aligned assembly.

Keywords: 3D Segmentation, Virtual Reality, 3D Printing, HCI

1 Introduction

Popular digital manufacturing facilities such as 3D printers make it easy for non-professionals to convert virtual digital models into physical objects. In recent years they have been subverting manufacturing and entertainment industries, both of which are spreading to ordinary consumers. However, only a small portion of ordinary users are affordable to buy a desktop 3D printer. The main reasons are: i) It is non-trivial for an ordinary user to create and edit a model, which need to employ most of the available professional 3D packages. However, 3D software often only provides 2D design interfaces to assist terminal users for creating 3D models. ii) The small dimensions of domestic desktop 3D printers limit the sizes of the printable models (Fig. 1). Although the ideas of gluing, connecting, or interlocking could solve the problem to a certain extent [1, 2, 3, 4], it is difficult to satisfy both multiple assemblies and seamless tightness. As a result, such an emerging problem in 3D printing has attracted more and more attention in the community.

The intuitive user interface in the VR environment provides a possible solution to tackle the above mentioned challenges. By integrating VR and 3D printing, we present a novel interactive model segmentation and assembly approach in VR environments for printing large models. To the best of our knowledge, it is the first work on segmenting and assembling models for 3D printing in HMD VR environments.

The main contributions of the paper include:

- A fastener-based, 3D model segmentation method that supports not only repeated disassembly but also tight and aligned assembly;

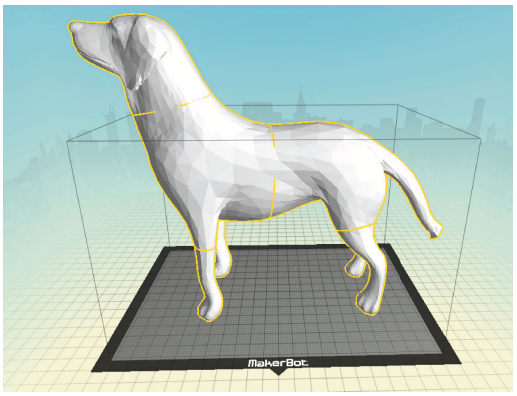


Figure 1: A model exceeding printing dimension

- A new VR-based user interface for segmentation, through which users can segment and assemble models according to their needs in an intuitive way;
- A new rotation collision detection and avoidance method based on cube cage and Boolean operations, which prevents segmented components from colliding with each other in the assembly later.

The remainder of this paper is organized as follows. After introducing the related works on model segmentation especially for 3D printing and shape design in HMD VR environment in Section 2, we provide the overview of our work in Section 3. Then, we describe an automatic bounding box generation method in Section 4, followed by the description of model segmentation based on Boolean operations in Section 5. In Sections 6 and 7, we detail rotation collision avoidance and bolt-nut configuration. Finally, we present experiment results in Section 8, and provide concluding remarks in Section 9.

2 Related Work

In this section, we will briefly review the mostly related research efforts on model segmentation especially for 3D printing and shape design in HMD VR environment.

3D Print-oriented segmentation. 3D shape analysis and component reuse require efficient semantic 3D shape segmentation. Tremendous progresses in 3D shape segmentation has been made in the past decade [5, 6, 7, 8, 9]. With the recent popularity of 3D printing, some segmentation approaches specifically designed for decomposing 3D models have been proposed. Due to transportation requirements and the limited printing volume, large models must be first separated into small components for printing and then assembled later. In the work of [10], an optimization algorithm for the partitioning and packing of a printable model was proposed in a multi-phase level-set framework, in which any specific way of assembly is not discussed. The Chopper algorithm proposed by Luo et al. [3] decomposes large objects into sub-components automatically, and then places the joint rivets between the components automatically through a simulated annealing algorithm. Song et al. [4] proposed a sub-module interlocking assembly, which supports multiple assemblies and disassemblies. Their recent CofiFab system [11] incorporates 2D laser cutting and 3D printing to produce large-scale 3D models, in which the internal structure is quickly generated by laser cutting, the surface details are obtained by fine 3D printing, and the two types of components are finally assembled together. Jadoon et al. [12] presented an flexible interactive tool for partition 3D models, which optimized the *Chopper* framework allowing more segmentation freedom, while detailed assembling was not discussed there.

3D model processing in immersive VR environment. Due to the developments of head-mounted devices such as Oculus Rift, HTC Vive, and other similar VR devices, more and more applications have started to use such VR systems. The adoption of immersive 3D interfaces for model processing can be dated to decades ago. The early immersive 3DM [13] and FreeDrawer [14] allow users to create polyline or spline based surfaces with simple 3D inputs. In recent CavePainting [15] and TiltBrush systems, users can create colorful art productions. Both “Drawing on Air” [16] and “Lift-Off” [17] study intuitive 3D curve inputs, and the latter also allows users to import a reference image. In the work of [18], visual and haptic feedback is used to provide the sensation of painting on virtual three-dimensional objects us-

ing the MAI Painting Brush++.

To assist young people with disabilities, McLoughlin et al. [19] proposed a method to create an artistic experience through virtual sculpting and 3D printing. Mendes et al. use novel mid-air metaphors to model complex 3D models with Boolean operations [20] and to select out-of-reach objects with iterative refinements [21] in virtual reality. As sketch-based modeling relieves users from tedious operations in professional packages, it has been extended into HMD environments. In the work of [22], Arora et al. analyzed various factors that could affect the human ability to sketch freely in a 3D VR environment. Giunchi et al. presented an approach to search for 3D models based on free-form sketches within a virtual environment [23]. Different from the above works, the focus of our work is to develop an immersive user interface in VR for model segmentation and assembly for 3D printing.

3 Approach Overview

As shown in Fig. 2, given an input Mesh M that need to be divided, the user first places a section P for segmentation using the dominant hand (usually the right hand) handle in a VR environment. If the user is not satisfied with the position of P, the non-dominant hand (usually the left hand) handle can be used to adjust it. Then, a bounding box V is automatically computed on one side of the section according to the segmentation plane C, which surrounds the connected part of the model on the same side completely. After that, two components, M_1 and M_2, are generated with the section as the segmentation interface, and the entire M is composed of M_1 and M_2: $M = M_1 \cup M_2$, which are produced with Boolean intersection/subtraction operations with M and V as input primitives (Equation 1):

$$\begin{cases} M_1 = M \cap V \\ M_2 = M - V. \end{cases} \tag{1}$$

Once M is successfully segmented into M_1 and M_2, the bolt template B will be placed at the segmentation plane C for the component M_1 and the screwing simulation can be previewed. At the same time, a swept volume S of M_1 is

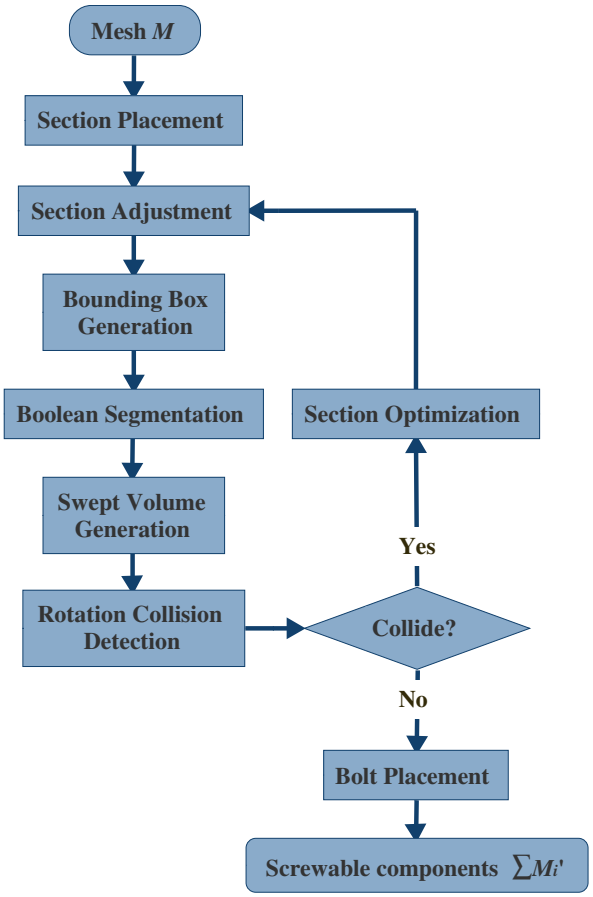


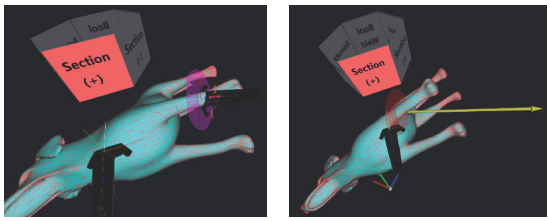
Figure 2: System workflow of a single segmentation

generated for rotation collision detection. If S intersects with the component M_2, the position and orientation of the section need to be optimized. If not, union and subtraction Boolean operations will be applied to M_1 and M_2 respectively, and a pair of screwable components $M_i'(i = 1, 2)$ is created.

4 Automatic Bounding Box Generation

4.1 Initial Section Placement and Adjustment

For the input mesh M, as shown in Fig. 3a, the user can edit the scale of the brush and place the section P directly with the dominant hand handle. Although the computation of the segmentation plane C is independent of the brush size, the user can tune the brush scale for convenience. In order to obtain a robust segmentation



(a) Initial placement of the section (circular purple plate)

(b) Section adjustment (circular red plate)

Figure 3: Place the section with both hands.

(a) Segmentation interface

(b) Simplified K3M

Figure 4: Segmentation center location.

contour, the section center should be as close as possible to the intended segmentation interface. Therefore, the non-dominant hand handle can be adopted to edit the position and orientation of the section P, as shown in Fig. 3b.

4.2 K3M-based Section Center Calculation

Both the bounding box generation and the bolt location depend on the center and the normal of the segmentation plane C, which coincides with its planar polygonal contour in 3D. Therefore, the normal vector can be obtained as the cross product of two normalized edges of its contour polygon, and its central position can be determined as follows.

The three-dimensional planar polygon is firstly transformed into a two-dimensional polygon and discretized into an image (Fig. 4a). Then the approximated center is calculated based on the simplified K3M algorithm [24] (refer to Fig. 4b). Although the K3M algorithm has certain advantages including retaining the right angle at the linear interconnection and producing a single-pixel wide skeleton, it is mainly used to generate a single-pixel line skeleton. In this work we extend it so that it can be applied for the bolt-nut placement at the segmentation interface in our system.

Similar to onion peeling, the center of C can be obtained by gradually eroding the segmentation interface from the outside to the inside. The boundary pixels are peeled off layer by layer, and the last pixel is served as the center. The original K3M algorithm requires seven steps for each iteration. During each iteration, it determines the number of the neighborhood points of its boundary pixels. It can be simplified into two iterations. The image boundary point is obtained at the first step (*Phase* 0), and the boundary points are deleted (converted to the background) at the second step (*Phase* 1). Finally, there is only one pixel left in the image, which is the center.

Phase 0: Traverse all the foreground pixels in the image, and mark them as boundary points when they have 1$\sim$7 neighboring pixels.

Phase 1: Traverse all the boundary points. When a boundary point has 2$\sim$7 neighboring pixels, its mark will be removed and it will be converted into a background pixel.

Ending condition: Count all the foreground pixels in the image, and stop the iterations when only one pixel is left in the image; otherwise, return to *Phase* 0.

4.3 Flooding-based Bounding Box Generation

In our approach, Boolean operations are utilized to segment the initial model, and the second primitive in a Boolean operation is the half-mesh agent which consists of the surface triangles of a compact bounding box.

It is nontrivial for a user to manually place a suitable bounding box given a complex model to be segmented. Therefore, we present an automatic component bounding box creation method after the placement of a cut plane. The procedure takes the cut position and its normal vector as input, and performs a flooding algorithm to recursively produce a compact bounding surface, which avoids users' complicated interactions.

Our automatic generation approach is inspired by the work of [25]. First, the bounding box enclosing the model M is initialized, which

is voxelized with a preset resolution. Then, the voxels are classified into three categories, that is, the *outer voxels* outside M, the *feature voxels* intersecting the mesh surface, and the *inner voxels* inside M. Finally, the bounding box V is generated by extracting the outer faces of the feature voxels.

The classification of voxels is derived from the signed distance field [26] of the voxel vertices to M. For each voxel corner c, there is the closet point p on M. When p lies on some facets of M, the normal of p can be directly calculated. Otherwise, if p is at the vertices or edges of M, an angle weighted pseud-onormal n_A is applied [26]. Therefore, the sign of c can be computed by Equation 2. If all vertices of a voxel are inside M, the voxel is an inner voxel with sign value -1. If only a part of the vertices of a voxel are inside M, it is a feature voxel with sign value 0. If all vertices of a voxel are outside M, the it is an outer voxel with sign value 1. They together make up the *tri-value distance field* [25]:

$$c \begin{cases} outside\ \ M, & if\ \ \boldsymbol{n}_A \cdot (c-p) > 0 \\ inside\ \ M, & if\ \ \boldsymbol{n}_A \cdot (c-p) < 0 \\ on\ \ M, & if\ \ \boldsymbol{n}_A \cdot (c-p) = 0 \end{cases} \tag{2}$$

Actually, the feature voxels F of the model component M_1 are the main source for producing the bounding box, and a voxel flooding algorithm is utilized for finding all inner voxels and feature voxels Y of M_1, from which F is selected. Firstly, a local coordinate system is constructed by taking the center of the segmentation interface C as the origin and its normal vector as the y-axis (Fig. 5). After that, we search for inner voxels and feature voxels of the first layer by starting from the center voxel of C as a seed. It is followed by recursive searching in the x-axis and z-axis directions, from which all inner voxels and feature voxels Y_1 from the first layer right above C are found and marked to avoid repeated lookups. Then, we use the inner voxels and feature voxels right above Y_1 as the seeds and then search for inner and feature voxels recursively in the x-axis, y-axis, and z-axis directions, and find all other inner voxels and feature voxels Y_2 of M_1 from all the remaining layers, $Y = Y_1 \cup Y_2$. Therefore, F is finally obtained by combining the feature voxels F_2 in Y_2 and

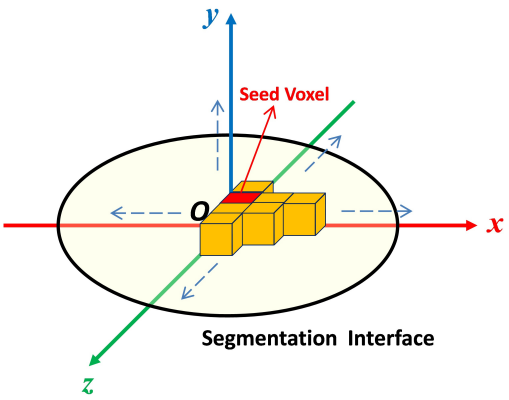


Figure 5: Flooding algorithm.

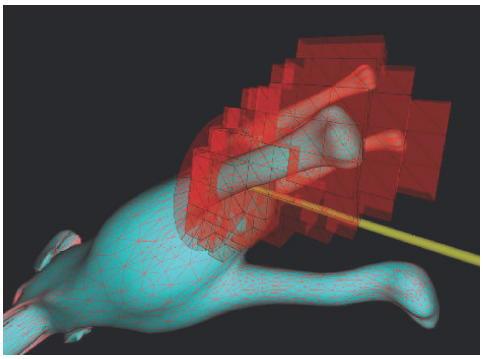


Figure 6: The final bounding box.

Y_1, that is, $F = Y_1 \cup F_2$.

The bounding box V is essentially the outer faces of F, which are adjacent to an outer voxel and a feature voxel. So the extraction and generation of V can be carried out by checking the voxel values adjacent to each face of each feature voxel based on the tri-value distance field (Fig. 6).

5 Model Segmentation based on Boolean Operations

5.1 Model Segmentation Principle

Although users can segment the input model M at any position and in any direction, in order to ensure the validity of the model segmentation and the convenience of subsequent assembly, the principles for users to follow are provided as follows:

1. Segmentation position: The segmentation position (i. e., segmentation center) of the

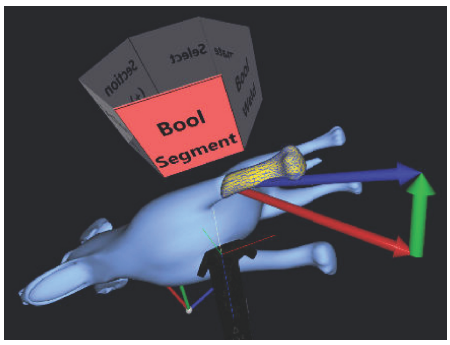


Figure 7: Boolean operation.

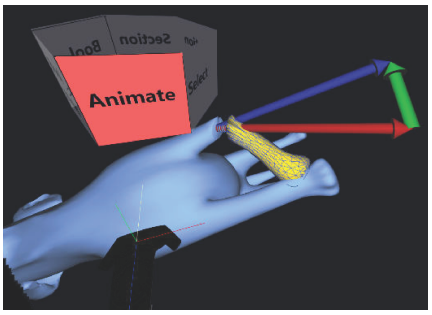


Figure 8: Screwing process simulation.

model M should be as close as possible to its skeleton center, preventing the model from being segmented into too many parts, which is also not conducive to assembly. At the same time, segmentation should be placed at the positions with locally similar cylindrical shapes, which benefits bolt-based assembly and reinforcement.

2. Segmentation direction: Although the model can be segmented in any direction in principle, the two segmented components M_1 and M_2 should be located in different half spaces with the segmentation interface as the boundary to reduce the possibility of collision during screwing.

5.2 Robust Boolean Operations

Three phases of our system are based on Boolean operations. That is, the Boolean-based segmentation of the model (Fig. 7), the use of Boolean operation on testing whether the swept volume S of one component intersects with another component for collision detection, and integrating bolts and nuts to the segmented components in the subsequent sections.

Due to the importance of Boolean operations in our system, we chose a rather robust method for mesh intersection calculation and Boolean operation in libigl [27]. To perform the Boolean operations of two triangle meshes $TriMesh_A$ and $TriMesh_B$, the unified "mesh configuration" [28] is firstly calculated to find the intersections of all triangles, and the new edges and vertices will be added at the intersection lines accurately. Then the voxels surrounded by the configured surface are marked according to their "winding number vectors". Finally, the boundaries of the corresponding voxels are extracted using specific Boolean operations (Union, Intersection, etc.).

6 Collision detection and optimization

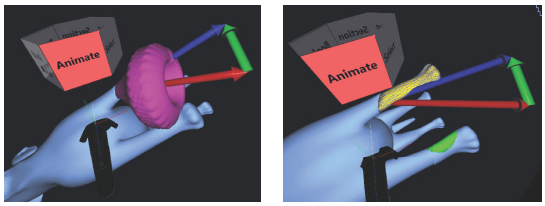
6.1 Rotation simulation and collision detection

After placing an improper segmentation, two of the components may collide with each other when screwing. To this end, we provide a solution for rotation collision detection and optimization. Moreover, the process of screwing the bolt is displayed with a previewing animation.

In order to simulate the screwing process of the components and to detect the collision, the bolt needs to be placed in the expected position temporarily. For components M_1 and M_2 sharing the same segmentation interface, the bolts and nuts can be assembled normally on either side. But in general, users are used to cutting a small part of the model, so we place the bolt B at the segmentation center O of M_1 regularly, and the normal vector of the bolt is aligned with the normal vector n_P of the segmentation interface. The component M_1 with a bolt is represented as M_1'.

The simulation of the component screwing process is rotating and moving M_1' with time t forth and back along the normal of the segmentation interface, similar to the process of screwing the bolt by hand in reality, as shown in Fig. 8.

The swept volume S is generated by screwing M_1. Using the swept volume algorithm provided in libigl [27], S is essentially the union of



(a) Swept volume (b) Collision region

Figure 9: Collision detection.

a moving solid object M_1 [29] (Equation 3):

$$S = \bigcup_{t\in[0,1]} f(t)M_1, \tag{3}$$

where $f(t)$ is a rigid motion over time t of M_1.

Since the surface of the swept volume generated by M_1 and $f(t)$ is a piecewise-ruled surface, which cannot be represented exactly by a triangle mesh. An approximate swept volume can be computed based on signed distances [29] (Fig. 9a), and an offset parameter can be set to approximate the exact swept volume.

Whether an intersection I between S and another component M_2 exists can be determined by a Boolean intersection operation (Equation 4), where I is the potential 3D collision region (Fig. 9b). The existence of I means that the position of the section is not proper, and the segmented components cannot be assembled after 3D printing. Therefore, the position and orientation of the section need to be optimized. If there is no collision between the segmented components during the screwing process, the bolt can be placed at the segmentation interface directly for 3D printing.

$$I = S \cap M_2. \tag{4}$$

6.2 Segmentation optimization

As mentioned above, if the collision region I exists, the position and orientation of the section P need to be optimized. We provide two optimization schemes, an intelligent version and an interactive one.

The intelligent optimization scheme focuses on optimizing the orientation of section P. The normal vector of the section is the same as the normal vector of the segmentation interface, n_P.

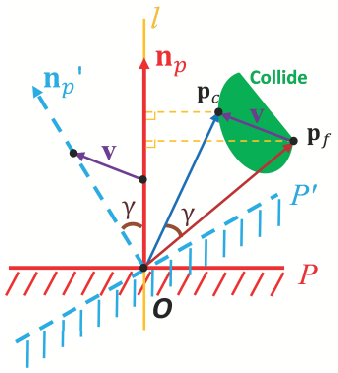


Figure 10: Section optimization diagram.

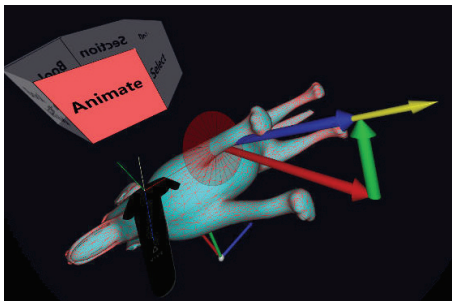


Figure 11: Optimized section.

As shown in Fig. 10, a four-step algorithm is employed to rectify the failed section direction:

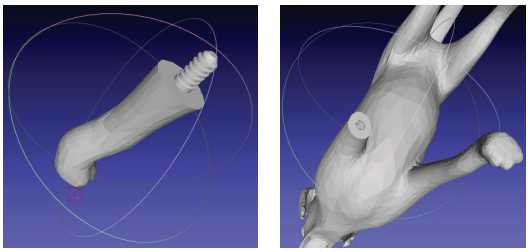
1. Compute the line l coinciding with n_P as its direction which passes the center O of the segmentation interface.

2. Find the closest point p_c and the farthest point p_f away from line l in the collision region I.

3. Calculate the angle γ between $\overrightarrow{Op_c}$ and $\overrightarrow{Op_f}$ as the rotation angle of P, taking $v = \overrightarrow{Op_c} - \overrightarrow{Op_f}$ as the rotating orientation of P.

4. Transform the section center to the center O of the segmentation interface, and rotate the section with angle γ along v, thus obtaining an optimized section P', as shown in Fig. 11.

Since the intelligent optimization scheme above can only avoid the current collision step, which means the optimized segmentation of a new section may still cause a new collision in other regions, the intelligent optimization can be



(a) Leg with a bolt (b) Body with a nut

Figure 12: Bolt and nut placement of a dog model.

performed iteratively until no collision occurs any more.

However, if no collision-free orientation can be found after too many iterations (a predefined maximum steps), an interactive operation is required and the collision positions will be visualized for reference. Users can place the section at a new location along the skeleton manually to satisfy the requirements with the non-dominant hand handle.

7 Bolt-nut configuration

Based on the segmentation interface center O, we can get the minimum distance R from O to the interface boundary. The radius r of the bolt and the nut should satisfy

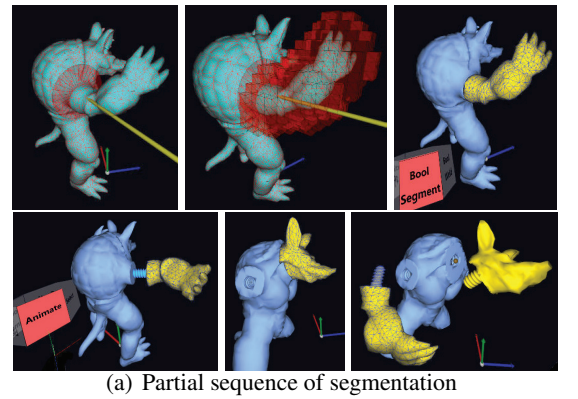
$$\frac{r}{R} \in [\alpha, \beta], \tag{5}$$

where α is used to avoid creating too small bolts, and β is used to avoid too thin walls of the nut hole, which may lead to collapse during screwing. In practical applications, we set $\frac{r}{R} = \frac{2}{3}$ in most cases. Users can adjust the value according to different requirements. In our experiments we empirically set $\alpha = \frac{1}{5}$, $\beta = \frac{4}{5}$, which have satisfactory results.

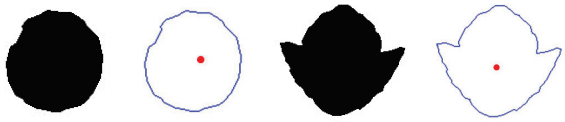
Attaching bolts and nuts onto the components $M_i\,(i = 1, 2)$ by Boolean union and subtraction operations will create a pair of screwable components (Equation 6):

$$\begin{aligned} M_1' &= M_1 \cup B, \\ M_2' &= M_2 - B. \end{aligned} \tag{6}$$

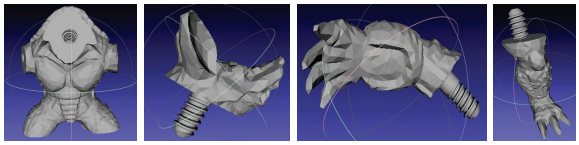
The example of a dog model with $\frac{r}{R} = \frac{1}{3}$ is shown in Fig. 12.



(a) Partial sequence of segmentation



(b) Segmentation interface with its center



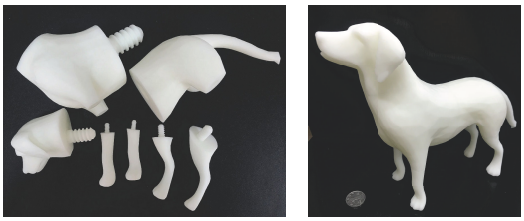
(c) Segmented components

Figure 13: Armadillo segmentation.

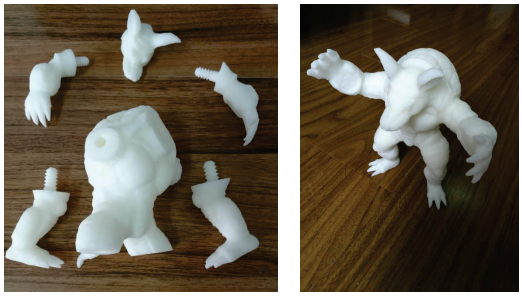
8 Experiments and Discussion

To validate the effectiveness of the proposed algorithm, we developed a prototype system based on OpenVR using HTC Vive helmet. Our experiments are performed on a desktop PC, with 4.0 GHZ Intel i7-6700K CPU, 8G memory and Nvidia GTX 1070 graphics card. In addition to the dog model above, another example of the Armadillo model segmented using our system is shown in Fig. 13. Fig. 13a shows a part of the segmentation. After each pair of component segmentation, the center of the segmentation interface is calculated automatically (Fig. 13b), which will be used for the placement of bolts and nuts. Components after segmentation can be seen in Meshlab [30], as shown in Fig. 13c.

As shown in Fig. 14, a single model can be segmented multiple times as needed, and all 3D printed models can be repeatedly disassembled and assembled. The assembled components with very tiny gaps are firmly connected with each other. According to our tests, if the model is too small, the screw threads of the generated bolts and nuts are so fine that they may collapse easily during the screwing assembly.



(a) Dog components (b) Assembled dog



(c) Armadillo components (d) Assembled Armadillo

Figure 14: 3D printing and assembly of model components.

The larger the model is, the better the result will be by using our system, which is just the requirement for large model segmentation before printing. Since we use robust Boolean operation based on "mesh configuration" and swept volume algorithm in the libigl library, our method is not real-time. But it does not matter as in our interactive system, a single Boolean operation takes only about 1∼2 seconds. Although the calculation of swept volume may take several seconds, it is only calculated once so it has little effect on the user's experience.

9 Conclusion

We have presented a VR-based segmentation and assembly approach for printing 3D models, which is suitable for dividing large-size models into small components and printing them separately. Pairs of bolt fasteners will be generated at the segmentation interfaces, which supports repeated seamless and firm assembly. In the segmentation procedure, users wear a VR helmet with high immersive experience, which provides convenient user interaction. Non-professional users can segment a 3D model on demand directly with VR handles.

Some steps in our system are based on Boolean operations using libigl library, which may affect the performance of our proposed approach. In order to achieve a real-time performance, an approximate GPU parallel Boolean operation [31] can be used for interactive display in our future interaction system, and its final processing of the model can use robust Boolean operations based on "mesh configuration".

Our system also provides collision detection and optimization of the segmented components, which guarantees successful assembly of component pairs with bolts and nuts.

However, there are still some limitations in our current approach, which can be further optimized in the future. First of all, in order to increase the robustness of the bolt assembly, a mechanical analysis can be used to determine the wall thickness and bolt diameter, which prevents the bolt from breaking during the screwing process. Secondly, the optimization schemes for the section in the case of collision can be improved. For the intelligent optimization scheme, it takes too long time to compute the swept volume. Therefore, the optimal orientation and position of the section should be automatically searched locally. For the collision detection during the screwing, we can refer to the 2D projection method in [32].

References

[1] Juraj Vanek, Jorge A. Garcia Galicia, Bedrich Benes, Radomír Mech, Nathan A. Carr, Ondrej Stava, and Gavin S. P. Miller. Packmerger: A 3d print volume optimizer. *Comput. Graph. Forum*, 33(6):322–332, 2014.

[2] Ruizhen Hu, Honghua Li, Hao Zhang, and Daniel Cohen-Or. Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213:1–213:12, 2014.

[3] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik. Chopper: partitioning models into 3d-printable parts. *ACM Trans. Graph.*, 31:1–9, 2012.

[4] P. Song, Z. Fu, L. Liu, and C. W. Fu. Printing 3d objects with interlocking parts. *Computer Aided Geometric Design*, 35-36:137–148, 2015.

[5] Rui S. V. Rodrigues, Jos F. M. Morgado, and Abel J. P. Gomes. Part-based mesh segmentation: A survey. *Computer Graphics Forum*, 37(6):235–274, 2018.

[6] Zhenyu Shu, Chengwu Qi, Shi-Qing Xin, Chao Hu, Li Wang, Yu Zhang, and Ligang Liu. Unsupervised 3d shape segmentation and co-segmentation via deep

learning. *Computer Aided Geometric Design*, 43:39–52, 2016.

[7] Truc Le, Giang Bui, and Ye Duan. A multi-view recurrent neural network for 3d mesh segmentation. *Computers & Graphics*, 66:103–112, 2017.

[8] Li Yi, Vladimir G. Kim, Duygu Ceylan, I-Chao Shen, Mengyan Yan, Hao Su, Cewu Lu, Qixing Huang, Alla Sheffer, and Leonidas J. Guibas. A scalable active framework for region annotation in 3d shape collections. *ACM Trans. Graph.*, 35(6):210:1–210:12, 2016.

[9] David George, Xianghua Xie, and Gary K. L. Tam. 3d mesh segmentation via multi-branch 1d convolutional neural networks. *Graphical Models*, 96:1–10, 2018.

[10] Miaojun Yao, Zhili Chen, Linjie Luo, Rui Wang, and Huamin Wang. Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph.*, 34(6):214:1–214:11, 2015.

[11] P. Song, B. Deng, Z. Wang, Z. Dong, W. Li, C. W. Fu, and L. Liu. Cofifab: coarse-to-fine fabrication of large 3d objects. *ACM Trans. Graph.*, 35:1–11, 2016.

[12] Aamir Khan Jadoon, Chenming Wu, Yong-Jin Liu, Ying He, and Charlie C. L. Wang. Interactive partitioning of 3d models into printable parts. *IEEE Computer Graphics and Applications*, 38(4):38–53, 2018.

[13] Jeff Butterworth, Andrew Davidson, Stephen Hench, and Marc Olano. 3dm: A three dimensional modeler using a head-mounted display. In *Proceedings of the 1992 Symposium on Interactive 3D Graphics, SI3D '92*, pages 135–138, 1992.

[14] Gerold Wesche and Hans-Peter Seidel. Freedrawer: a free-form sketching system on the responsive workbench. In *VRST*, pages 167–174, 2001.

[15] Daniel F. Keefe, Daniel Acevedo Feliz, Tomer Moscovich, David H. Laidlaw, and Joseph J. LaViola Jr. Cavepainting: a fully immersive 3d artistic medium and interactive experience. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics, SI3D*, pages 85–93, 2001.

[16] D. F. Keefe, R. C. Zeleznik, and D. H. Laidlaw. Drawing on air: input techniques for controlled 3d line illustration. *IEEE Trans. Vis. Comput. Graph.*, 13:1067–1081, 2007.

[17] Bret Jackson and Daniel F. Keefe. Lift-off: Using reference imagery and freehand sketching to create 3d models in VR. *IEEE Transactions on Visualization and Computer Graphics*, 22(4):1442–1451, 2016.

[18] Mai Otsuki, Kenji Sugihara, Azusa Toda, Fumihisa Shibata, and Asako Kimura. A brush device with visual and haptic feedback for virtual painting of 3d virtual objects. *Virtual Reality*, 22(2):167–181, 2018.

[19] Leigh McLoughlin, Oleg Fryazinov, Mark Moseley, Mathieu Sanchez, Valery Adzhiev, Peter Comninos, and Alexander A. Pasko. Virtual sculpting and 3d printing for young people with disabilities. *IEEE Computer Graphics and Applications*, 36(1):22–28, 2016.

[20] Daniel Mendes, Daniel Medeiros, Maurício Sousa, Ricardo Ferreira, Alberto Raposo, Alfredo Ferreira, and Joaquim A. Jorge. Mid-air modeling with boolean operations in VR. In *2017 IEEE Symposium on 3D User Interfaces, 3DUI 2017, Los Angeles, CA, USA, March 18-19, 2017*, pages 154–157, 2017.

[21] Daniel Mendes, Daniel Medeiros, Maurício Sousa, Eduardo Cordeiro, Alfredo Ferreira, and Joaquim A. Jorge. Design and evaluation of a novel out-of-reach selection technique for VR using iterative refinement. *Computers & Graphics*, 67:95–102, 2017.

[22] Rahul Arora, Rubaiat Habib Kazi, Fraser Anderson, Tovi Grossman, Karan Singh, and George W. Fitzmaurice. Experimental evaluation of sketching on surfaces in VR. In *Proceedings of the 2017 CHI Conference on Human Factors in Computing Systems, Denver, CO, USA, May 06-11, 2017.*, pages 5643–5654, 2017.

[23] Daniele Giunchi, Stuart James, and Anthony Steed. 3d sketching for interactive model retrieval in virtual reality. In *Proceedings of the Joint Symposium on Computational Aesthetics and Sketch-Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering*, Expressive '18, pages 1:1–1:12, 2018.

[24] K. Saeed, M. Tabedzki, M. Rybnik, and M. Adamski. K3m: a universal algorithm for image skeletonization and a review of thinning techniques. *Int. J. Appl. Math. Comput. Sci.*, 20:317–335, 2010.

[25] C. Xian, H. Lin, and S. Gao. Automatic generation of coarse bounding cages from dense meshes. In *IEEE International Conference on Shape Modeling & Applications*, pages 21–27, 2009.

[26] J. A. Baerentzen and H. Aanaes. Signed distance computation using the angle weighted pseudonormal. *IEEE Trans. Vis. Comput. Graph.*, 11:243–253, 2005.

[27] A. Jacobson, Daniele. Panozzo, C. Schller, O. Diamanti, Q. Zhou, S. Koch, and et al. libigl: a simple c++ geometry processing library. `http://libigl.github.io/libigl/`, 2017.

[28] Q. Zhou, E. Grinspun, D. Zorin, and A. Jacobson. Mesh arrangements for solid geometry. *ACM Trans. Graph.*, 35:1–15, 2016.

[29] A. Garg, A. Jacobson, and E. Grinspun. Computational design of reconfigurables. *ACM Trans. Graph.*, 35:1–14, 2016.

[30] P. Cignoni, M. Callieri, M. Corsini, M. Dellepiane, F. Ganovelli, and G. Ranzuglia. Meshlab: an open-source mesh processing tool. In *Eurographics Italian Chapter Conference*, pages 129–136, 2008.

[31] H. Zhao, C. C. L. Wang, Y. Chen, and X. Jin. Parallel and efficient boolean on polygonal solids. *Visual Computer*, 27:507–517, 2011.

[32] T. Sun and C. Zheng. Computational design of twisty joints and puzzles. *ACM Trans. Graph.*, 34:1–11, 2015.