基於龍圖兒(Zometool)之快速大型原型生成

吳宗鴻* 黃群凱* 陳奕麟† 陳炳宇†
國立臺灣大學
*{zonhon, chinkyell}@cmlab.csie.ntu.edu.tw †{yilingchenntu, robin}@ntu.edu.tw

ABSTRACT
最近因為三維印表機的取得容易許多，使得個人化製造的相關領域受到較多關注，但是一般市面上的三維印表機存在兩個大問題：需要較長的列印時間以及輸出受限於印表機大小，這些問題使得市售的3D印表機無法做到快速大型原型生成，在此研究中，我們提出一個有效率的方法來使用龍圖兒(Zometool)做出一個近似原三維模型的結果。為了要使組裝更容易以及使用更少材料，方法中使用了大區塊的形狀抽象化，輸入的三維模型先以分區法分為許多類似圓柱形的區塊，接著將各區塊的邊界轉為環形的龍圖兒結構，之後再以搜尋最短路徑的方法找尋環形之間的連接結構，最後再順著各分區的軸產生一些中間的環來逼近原三維模型，此論文並附上一些實際拼出的結果圖與分析圖表來展示本研究方法的實用性。

Categories and Subject Descriptors
I.3.3 [**Computer Graphics**]: Randomness, geometry and discrete structuresComputational Geometry;

General Terms
Algorithms

1. INTRODUCTION
The production cost of 3D printer has been declining and it is available for everyone, so large amount of applications and requirements for fabrication emerge. As a result, fabrication becomes a significant topic. However, the consumer level 3D printers have some drawbacks: the time consuming process, the limited output size and relatively high cost of materials. Large-scale fabrication is needed for some situations. For instances, the props for stage play are usually composed of wooden boards and lack complete 3D structure. In addition, The prototypes for art exhibition are needed when planning in advance. The large-scale prototypes require fast construction and just fit the coarse outline, but recent 3D printers can not realize these features.

The popular modeling system, *i.e., Zometool* [2], is potentially suitalbe for providing an alternative solution to the abovementioned scenarios. It has several advantages: Firstly, stability, expandability and lightness satisfying the requirements for large-scale fabrication. Secondly, its independent structure and modularity can parallelize the construction to speed up the building process. However, even for 3D models of moderate complexity, novice users may still have difficulty in building visually plausible results by themselves. Therefore, the goal of this work is to develop an automatic system to assist users to realize Zometool rapid-prototyping with a specified 3D shape.

In this paper, we present a novel technique, which optionally accepts moderate user input, to automatically generate Zometool structures approximating a given 3D input model. The proposed method first performs mesh segmentation to split the complex 3D model into a collection of parts of lower complexity resembling generalized cylinders or cones. Then, the oriented bounding box of each segment is computed to extract the representative axis of each segment. The main axis in each segment (named growing axis) have a great influence of constructing Zometool structure, so we implement an algorithm to choose the best axis. It can not be decided only by bounding box, and we also use the segment information to judge the best one. The other important factor is obtaining the feature points. There are two types of features, interjacent ones is in section 3.2 and outward ones in section 3.3. These two types come out with different situation about the number of segments adjacent to a segment(N_a), and outward features are the special cases of the interjacent features when $N_a = 1$. Then we find the Zometool structure with all features, and deal with the balance of approximate solutions and optimized solutions for speed and error.

Recently, several computational schemes have been proposed to accomplish shape approximation with Zometool [2, 3]. In [3], volumetric voxelization is adopted to obtain a rough approximation of the input model, making the method feature-insensitive in nature. Because the resolution of the regular grid is fixed for the whole model, it is difficult to keep the features and reduce the usage of materials at the same time. [2] uses computation intensive optimization to fit the surface and rebuild the structure when encountering deadlocks. The previous methods generate 2-manifold mesh approximation of the given shape while our method aim to achieve more economic usage of building units with higher level of shape abstraction. For example, our approach allows non-manifold structures, and thus can present the features directly with less building units. Our work use flexible structure to build up the result, so we can tolerate a few error to accelerate the optimization. We find the features first to solve this problem, and then get some sample points on surfaces as the new features to construct a similar surface as these works.

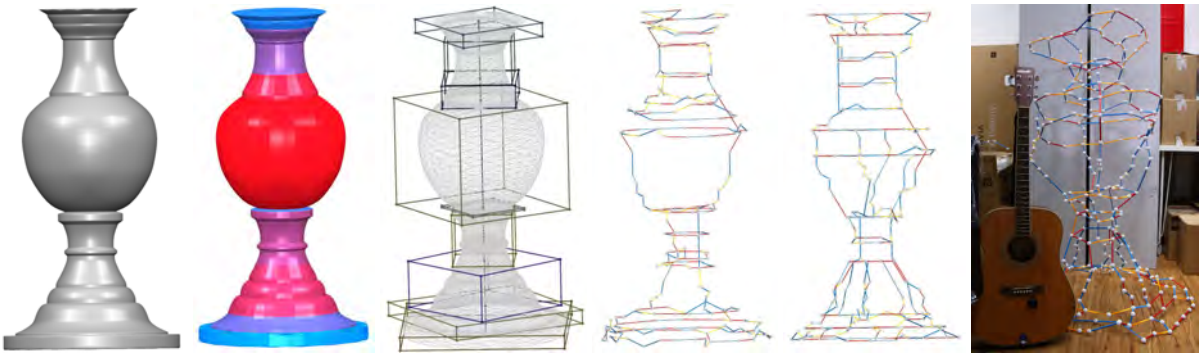

Figure 1: First, we obtain the original model as (a). Second, the system run segmentation as (b). Third, we calculate the oriented bounding box and growing axis (explanation in introduction) as (c). Forth, the system generate a base result as (d). Next, the user can semi-automatically add supports to refine the result as (e). Finally, construct the model with Zometool as (f).

There are two primary contributions in this paper: First, our approach use the non-manifold structure to present the model for emphasizing the features, and minimize the amount of building units at the same time. Second, economic usage of Zometool building units give us an edge over the two papers [2, 3], and accelerate the construction at the same time.

2. OVERVIEW

There is a flow chart at figure **??**, first step is **mesh segmentation**. User has two parameters to control: cluster level is the number of clusters for the soft clustering and smoothness is a factor which indicates the importance of the surface features for the energy minimization. High smoothness results in a small number of segments, since constructing a segment boundary would be expensive. In other words, merging facets that are placed under different clusters is less expensive than separating them and creating boundaries. After mesh segmentation has finished, we can get the points which are the boundary of the segment and cross between two segments (named cross points). We order the points to a ring (named cross ring), and the method is shown in section 3.1.1.

Second step is to **calculate the oriented bounding box and obtain growing axis** for all segments. Growing axis is one of the axis of the oriented bounding box and it decides the main direction for growing the Zometool structure in each segment. The method for judgement is shown in section 3.1.2.

In the next step, we connect all cross rings and the order to execution is decided by N_a. In a segment, we also decide the execution order with the growing axis by projecting the center points of the cross rings to the growing axis. When we connect the rings, we also add features at the same time, and then we add the supports to connect the weak structures. The section 3.2 explains it.

Finally, the system transforms the polylines to the Zometool struts, and we provide the convenient user interface for constructing the result. User can look up the struts informations for each node to speed up the process of building up the result.

3. ZOMETOOL CONSTRUCTION

3.1 Preprocessing

After the model has loaded, the next step is model segmentation. Users can set cluster level and smoothness parameter to get the segmenting result they want, however, the default value can cover in most instances. System will give a appropriate segmenting result unless the user needs a special output just like excessive number of segments.

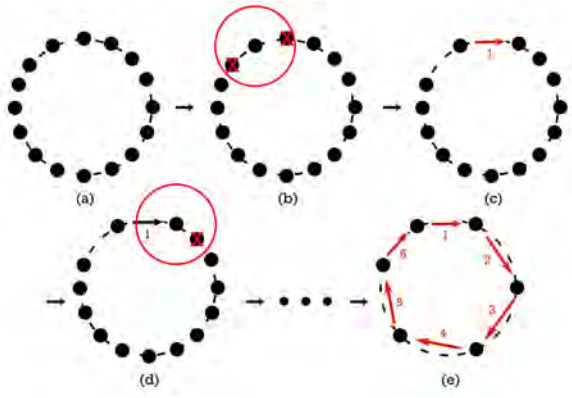

Figure 2: Ring generation. (a) All points. (b) Delete the points with the distance less than the length of the shortest strut. (c) Mark next point. (d) Repeat step(b)(c). (e) Get ring.

We expect the segmentation results would be general cylinders or cones, and we adopt the shape diameter function [1] to segment the input model, because we will slice the model for getting the rings to compose the features, and general cylinders and cones can easily predict the middle slice and the end. It can decrease the complexity and speed up the process, so it is suitable for rapid fabrication.

3.1.1 Ring Generation

Now we have groups of cross points, and we need orders for the points to create the Zometool structure, so we make the cross rings.

First we delete some points to keep the least distance between any two points. The distance is set the length of the shortest Zometool strut. Then we pick one of them to be the head, and start finding the closest point one by one as Figure 2. In this way, cross points would be threaded to a cross ring.

3.1.2 Axis Decision

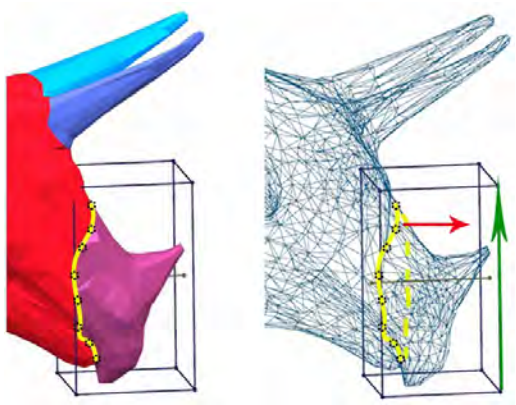

Figure 3: Example of the special axis selection. In the segment, the longest axis of bounding box(as green arrow) may not be the best axis, and the axis draw (red arrow) in the Figure is the most appropriate. Yellow curve is the cross ring, and its normal vector is red arrow. Because the $N_a = 1$, we just use the normal vector to be the growing axis ,and not the longest OBB axis(green arrow).

The segmentation and cross ring generation are complete, and then we calculate the oriented bounding box(OBB) for each segments. Because the process of finding OBB is fast and we just need a approximate solution, so we just use brute-force method to scan all directions with the alternation of 5 degree to get the OBB. There are three axis for each bounding box, but the system needs a main axis to make sure that the system can get the most feature points from the original model for constructing Zometool structure, because if the growing axis has the closest direction to the normal vectors of faces which are made from cross rings, the result of slicing the model would be well-distributed.

In most cases, the longest axis would be the best axis to choose, but sometimes the longest one is not appropriate as shown in Figure 3. We assume segments to be general cylinders, so we approximate the cross ring with a plane and calculate the normal vector of it. when N_a=1, there is only a normal vector of cross ring, we compare the three axis of the OBB, and use the closest axis. when N_a>1, We have to separately count the number of fitting as N_a=1, and if there is only a counter of axis more than two, it means all normal vectors direct the same, so we select that axis; and if there are more than two counters has the number more than two or all counters have the number less than two, it means that the growing axis can not be decide by normal vectors and we just use the longest axis to be the growing axis. Let $\mathcal{P}$ be the points on the cross ring, $\bar{\mathbf{p}}$ be the centroid of $\mathcal{P}$. The normal vector $\mathbf{n}$ of the approximate plane of $\mathcal{P}$ is obtained by the following equation:

$$\mathbf{n} = \sum_{\mathbf{p}_i \in \mathcal{P}} \left\{ \frac{\mathbf{p}_i - \bar{\mathbf{p}}}{|\mathbf{p}_i - \bar{\mathbf{p}}|} \times \frac{\mathbf{p}_{i+1} - \bar{\mathbf{p}}}{|\mathbf{p}_{i+1} - \bar{\mathbf{p}}|} \right\}. \quad (1)$$

We then calculate the angle between $\mathbf{n}$ and the three axes of oriented bounding box. The axis with the minimum value is chosen as the representative axis.

3.2 Structure Construction

Now we only have the cross rings, so we connect the rings in this section. In a segment with $N_a > 1$, the vacancy between cross rings might exist some features on the original model, we call it

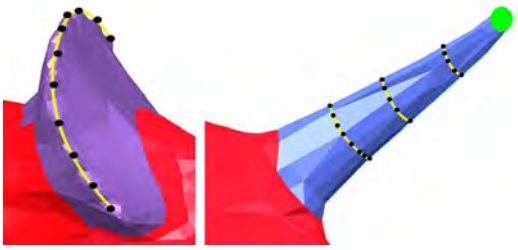

Figure 4: Additional features on the model . The yellow rings are interjacent features and the green point is outward feature. Interjacent features add a ring of struts to fit, and outward features may add a ring or a point according to the end.

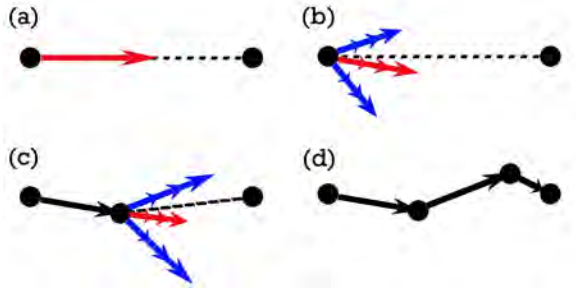

Figure 5: Demonstration of path connection. (a) Calculate the vector between end point and start point (V_m), (b) Use V_m to find the closest direction on node (D_m as red arrows) and get the near directions of the D_m (blue arrows). Each direction has three sizes of struts. (c) Add the best vector to path and loop the step(b). , (d) Get the whole path. The detailed approach is in section 3.4.

interjacent features. The following three functions construct the structure and fit the interjacent features: **path connection**, **slice ring generation**, and **support addition**, and the detailed descriptions are in the following subsections.

Path Connection

This function connect two points and calculate the path of the Zometool from start point to end point as figure 5, and there are two approaches to find the path. The implementation of quick approach and optimized approach are in the section 3.4. After using the approaches above, We obtain intermediate nodes between start point and end point. Then we use the **slice ring generation** to fit the surface.

Slicing Ring Generation

The figure 6 demonstrate it. For fitting the surface, we slice the model with the plane which have the normal vector as growing axis on the segment to get the sample points, and then use the **ring generation** in section 3.1.1 to string the sample points.

Support Addition

The rings created by previous subsections only connect like a string, but they are not stable because the rings only connect with one Zometool path. As we know, one point on the ring is connected, so we get the farthest point relative to that point to link to the whole Zometool structure by searching the closest point on it. Show in figure 7.

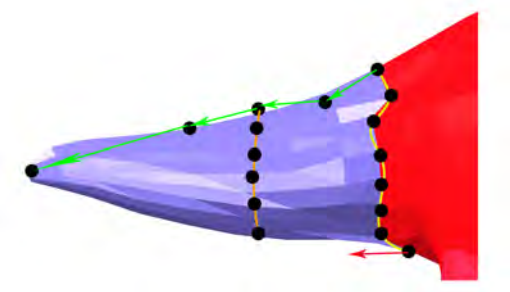

Figure 6: Demonstration of Slicing Ring Generation. Yellow curve means the cross ring, and the growing axis is the red arrow. There is a path as green arrows, and we will slice the model to get the interjacent features just like the orange curve.

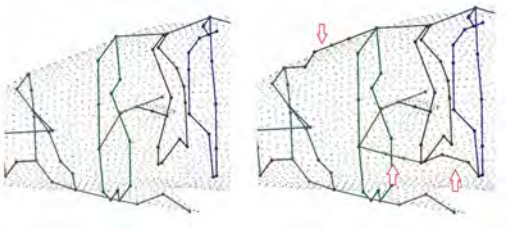

Figure 7: Difference between adding support or not. The two rings in the middle only connect with a strut, so it is not stable. After we add the support automatically, the structure is stronger than before.

3.3 Outward Features

When there is a outward feature in the segment, it means that segment has $N_a = 1$ and the feature is at the end away from the crossing ring. Figure 4 shows an example. The shape of the segment might be a cylinder or a cone, but our approach can deal with these two situation in the same way, because of the non-manifold structure. We first use the growing axis to find the farthest point from the cross ring, and use **slice ring generation** to get the ring (or a point when the sample points are too close) at end. Then use the **path connection** to connect the end ring and cross ring.

3.4 Zometool Path Search

We have a lot of feature points on the model, but it is impossible that the points can be connected directly by Zometool structure, so the system should find out the result to connect any two nodes. The following paragraphs show the methods to connect points using Zometool struct. Both approaches use the vector (V_m) from start point to end point to find the closest direction on the node as the initial guess, and it correspond to a direction (D_m) on node, named core direction.

Quick approach. Test all directions near the D_m and all size of the struts. In figure 8 shows the close directions for any direction, and quick approach use both one and two layer directions to find the best place to be close to the end point. Let P_s be the start point, P_e be the end point, D be all directions in two layers according to D_m, $L(d, s)$ be the three lengths of the struts with direction (d) and strut size (s), s is from 0 to 2 corresponding to three size of struts. The best direction and size of strut are decided by the equation below:

$$\min_{d \in D, s \in \{0,1,2\}} \|P_e - (P_s + d * L(d, s))\| \tag{2}$$

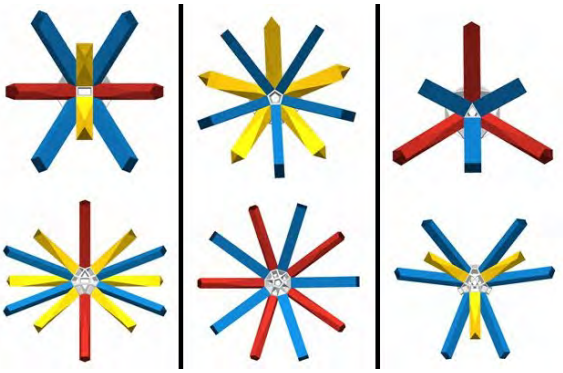

Figure 8: All close directions. The center hole is the core direction, and columns from left to right represent the blue, red, and yellow strut as the core direction. The upper row represent layer 1, and the row below is layer 2.

Although we brute-force all the possibilities to find the answer, but this approach only run all circumstances once to get the closest position. The above equation is just one step to approach the end point. This method will loop until the path is close enough within the threshold, and the following equation shows the stopping inequality:

$$\left\| P_e - (P_s + \sum_{i \in V} v_i) \right\| < T, \tag{3}$$

where V refers to the vectors in the path, and T is the threshold. This approach may generate more error because it only get the local minimum, but it is enough for the situation which is not care about completely matching to the end point.

Optimized approach. Use A-star algorithm to optimize the result, and the energy is the summation of the path walk though and the distance from the node on the end of path to the end point. We only use one layer close directions to decrease the amount of calculation, because most of the directions on layer two increase the energy rapidly. The termination criteria is judging that if the distance from the node on the end of path to the end point is less than the threshold(default 0.3cm) we set. The following equation shows the energy (E) of the A-star:

$$E = \sum_{i \in V} \|v_i\| + \left\| P_e - (P_s + \sum_{i \in V} v_i) \right\|. \tag{4}$$

E decides the order to add next path, and the smaller E will be ordered in the front, and the stopping inequality is the same as quick approach but threshold is very close to zero. Quick approach and optimized approach both can create the Zometool struts between two points, but these two method are quite apart in time consuming, so we mix them to balance the time and error. When the distance between start point and end point is more than the longest struct, we use the quick approach first and run optimized approach on the last part, so we can get a good enough result in a short time.

4. RESULTS AND DISCUSSION

In this section, we demonstrate the results of our system on a number of examples as the figure 10 and table 1. Our system automatically generate the base result, and we provide the function for user to refine the output. The base result guarantee the connectivity and the basic stability. If the users need a stabler result, they just need to

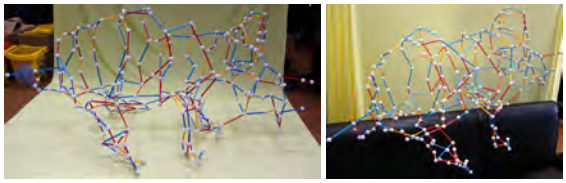

Figure 9: Zometool structure of triceratops.

find out pairs of points they want to connect, and the system will automatically create the path for Zometool structure.

We choose the Computational Geometry Algorithms Library (CGAL) to deal with the segmentation. The package of triangulated surface mesh segmentation in CGAL provides an implementation of the algorithm relying on the shape diameter function [1].

The data of the Zometool struts and nodes are set by the real size, and there are some important parameters: 1. Minimum diameter of a ring is set to be the shortest strut plus the diameter of node, because if the diameter of a ring is smaller than the smallest strut, it does not exist the structure to construct a ring, and it judges the whether the ring should be shrink to a point. 2. Minimum size of connect two points is the same as the minimum diameter of a ring because the shortest strut is the same, and it is the condition to know if two points has a strut to connect. 3. Maximum size of connect two points is set to be the longest strut plus the diameter of node, and it is ised to judge to be the boundary to use the quick approach or optimize approach. 4.optimized threshold is set to be 0.3cm for the Zometool has a few flexibility to tolerate this error in our non-manifold structure.

Performance. Table 1 shows the model information used in this paper and the performance measured on a desktop PC equipped with an Intel i7 3770 3.40GHz CPU and 20GB RAM. The optimization time is proportional to the number of vertices,and it usually takes only seconds to obtain the result.

Limitation and future work. The proposed still has several limitations. First, it sometimes cannot find the most appropriate growing axis when local mesh vertices are isotropically distributed and the orientation check is ambiguous. In this case, we simply select the longest axis of the OBB. Second, our approach mainly deal with the connectivity and feature maintenance, and we provide semi-automatic structural supports to enhance the stability. It can be a future work to consider the structure stability, and generate results of different fineness to give users to select the sparse or complex result they needs.

5. CONCLUSION

We have presented an system for large-scale rapid-prototyping that enables users to semi-automatically generate a Zometool structure with a 3D model input. The main property of the approach is non-manifold based structure to rapidly fit the features and decrease the amount of building units in the same time. In contrast to the previous works about Zometool, we can use less building units to construct the model in the same size, because of our sparse structure, so we can also build up the result faster. And we provide a convenient GUI for user to speed up the process of construct the result. The proposed technique can be use as the large prototype in exhibition, the the props for stage play, and all situations which need the prototype constructed rapidly.

6. 致謝

本論文感謝科技部經費補助，計畫編號：MOST103-2221-E-002-158-MY3、MOST103-2811-E-002-040 與 MOST103-2218-E-002-030。

7. REFERENCES

[1] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *Vis. Comput.*, 24(4):249–259, 2008.

[2] H. Zimmer and L. Kobbelt. Zometool rationalization of freeform surfaces. *IEEE Trans. Vis. Comput. Graph.*, 20(10):1461–1473, 2014.

[3] H. Zimmer, F. Lafarge, P. Alliez, and L. Kobbelt. Zometool shape approximation. *Graph. Models*, 76(5):390–401, 2014.

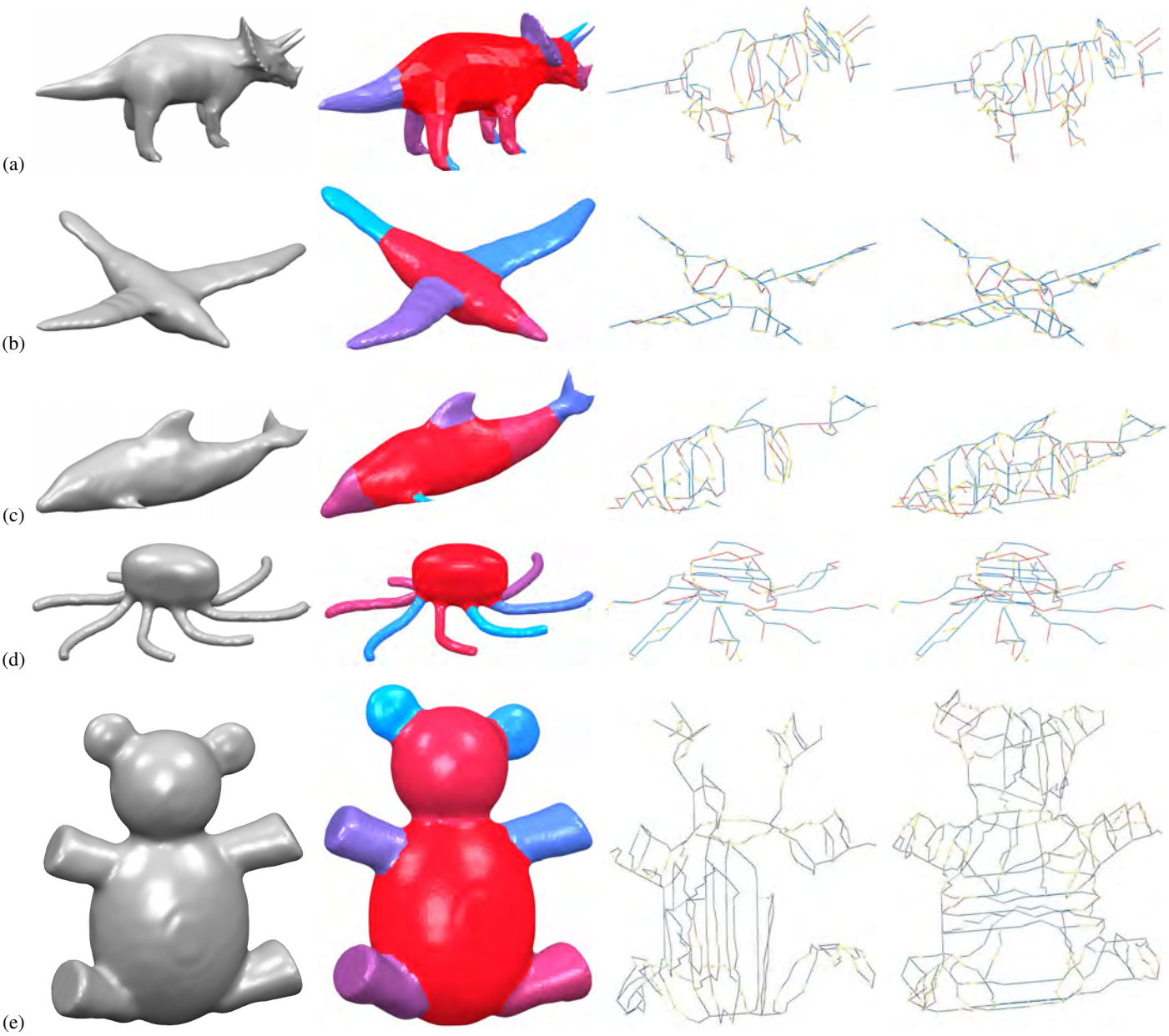

Figure 10: Some results:(a)triceratops (b)bird (c)dolphin (d)octopus (e)teddy. The column left to right are the original models, segmentation result, base results, and the refined results by user, respectively. The length of longest axis of (a) (d) are 100 cm and (e) is 120 cm, and the detailed data is in the table 1.

					base		refine	
model	# of Vertex	# of Edges	Segmenting Time	Processing Time	node	strut	node	strut
triceratops	2832	8490	2.515s	0.867s	187	213	610	679
bird	3478	10428	2.341s	0.31s	93	116	121	270
dolphin	5216	15642	3.524s	0.342s	137	154	223	420
octopus	5944	17832	4.165s	0.524s	144	164	171	557
teddy	13826	41472	9.487s	1.729s	394	434	831	1385
vase	9299	27891	10.619s	0.367s	288	315	601	675

Table 1: The table records the time to segmentation, create the base results, and the numbers of building units of base results and refined results.