ZomeFab: Cost-effective Large-scale Fabrication with Interior Zometool Structure

Ming-Shiuan Chen, I-Chao Shen, Chun-Kai Huang, and Bing-Yu Chen

Abstract— In recent years, personalized fabrication has attracted many attentions due to the widespread of consumer-level 3D printers. However, consumer 3D printers still suffer from shortcomings such as long production time and limited output size, which are undesirable factors to large-scale rapid-prototyping. In order to fabricate a large-scale object, we propose a 3D fabrication method that combines 3D printing and Zometool structure for cost-effective fabrication of large objects. The key of our approach is to utilize compact, ~~solid~~ sturdy and re-usable internal structure (Zometool) to infill fabrications and replace ~~the~~ both time and material-consuming 3D-printed materials. ~~And~~ Moreover, as a result, we can ~~greatly~~ significantly reduce the cost and time by printing ~~a~~ thin 3D external shells only. We design an optimization framework to generate both Zometool structure and printed surface partitions by balancing between several criteria including printability, material saving, and Zometool structure complexity. We demonstrate the effectiveness of the proposed method by a variety of 3D models along with examples of the physically fabricated objects.
Keywords: Geometric Algorithms, Fabrication, Curve, surface, solid, and object representations.

Index Terms—Geometric Algorithms, Fabrication, Curve, surface, solid, and object representations

1 Introduction

The recent widespread of consumer-level 3D printer lead to the emergence of large amount academic and industrial fabrication applications. However, the consumer-level 3D printers have several shortcomings including long printing time, limited output size and the high cost of materials. In order to address these shortcomings, many of the researches are proposed. For saving time and materials, the modern 3D printers allow users to save materials by switching between different fill-rate settings. Meanwhile, different internal patterns are proposed [19] to save materials while maintaining the ~~structure~~ structural soundness. For building a large-scale fabrication, lots of research~~es~~ [22, 10, 20, 13, 32] focus on this problem and share the spirit of partitioning the object into sub-parts that can be fitted into printing volume.

However, the saving of time and material from existing methods are still not sufficient for large-scale shape fabrication. ~~To decrease cost extremely~~ To immensely decrease the cost, our novel idea is to combine 3D printing with another structure. That desire structure must meet several criteria such as (i) easy to assemble, (ii) re-usable, and (iii) robustness, so that the resulted fabricated shape is simple to build and reliable simultaneously. Base on our requirements, the popular modeling system, i.e. *Zometool* [7], is suitable for coarse fabrication. In addition to the above advantages, *Zometool* still has several ~~good~~ excellent characteristics: (i) structural properties, such as stability, expandability and lightness satisfying the requirements for large-scale fabrication. (ii) Independent structure and modularity can parallelize the construction to speed up the building process. Hence, Zometool is a ~~good~~ attractive candidate to replace solid 3D printing material as the ~~inner robust~~ robust inner structure of large-scale structure. Therefore, the goal of this work is to develop a computational method that combines Zometool structure and 3D printing to reduce the time and material costs in large-scale fabrication. Given the desired 3D shape, we design an optimization process to synthesize the inner Zometool structure, which balances between the shape similarity and structure complexity. We leverage Simulated Annealing to explore the huge structure space effectively. Next, with the optimized Zometool structure, we hollowed the shape to obtain the outer shell and partition the outer shell ~~with respect to~~ concerning several criteria including simplicity and printability. We formulate these criteria in a single MRF problem and solve it with graph-cut algorithm [3]. We also design a ~~special~~ particular type of connectors ~~and~~ furthermore, optimize their positions for assembling the inner Zometool structure and outer shell.

There are two primary contributions in this paper:

1. We propose an optimization framework to synthesize the inner Zometool structure to replace the solid printed materials in large-scale fabrication, which greatly reduce the printing cost including time and materials.

2. We design and print a special connector and optimize their layout for better combining both inner Zometool structure and outer printed shell.

2 Related Work

2.1 Computational Fabrication

In recent years, computational fabrication has attracted many attentions in the computer graphics and human-computer interaction research fields [25]. Numerous works are proposed to fabricate shapes (i) with different objectives, e.g., maintain balance [23, 2], reduce size [20], strengthen ~~structure~~ structural soundness [37] and generate specific sounds [31], and (ii) with different materials or building blocks, e.g., Lego [21], planar slices [4], and interlocking puzzles [29].

However, even with the fast developments of assist tools and algorithms, 3D printers still suffer from long production time, excessive material usage, and limited output size. To reduce the usage of print materials, Huang [14] and Wu [35] design devices and algorithms to print shapes in wireframe. Meanwhile, ~~different internal structures are developed~~ several research papers develop different internal structures, e.g., the skin-frame structure [33], the honeycomb-like structure [19], and 2D laser cutting shape proxy [28]. To enable the large shape to be printed using 3D printers, Luo *et al.* [20] developed an iterative planar-cut method, aiming to fit decomposed parts in the 3D printing volume while considering factors such as ~~assembility~~ assemblability and aesthetics. Yao *et al.* [36] proposed a level-set framework for 3D shape partition and packing. Compared to these works, our method fabricates a shape with both Zometool structure and 3D-printed parts, so that we can reduce the fabrication time and cost, with the reusability of Zometool structures.

2.2 Zometool Design and Modeling

Zometool is a mathematically-precise plastic construction set for building a myriad of geometric structures [7], from simple polygons

• *Ming-Shiuan Chen, I-Chao Shen, Chiun-Kai Huang, and Bing-Yu Chen are with National Taiwan University. E-mail: {intere2960, jdily, chinkyell}@cmlab.csie.ntu.edu.tw, robin@ntu.edu.tw*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org.
Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx/

to visualize and model various natural sciences, e.g. DNA molecules. ~~It's~~ It is history dated back to to the 1960s where it started out as a simple construction system inspired by Buckminster Fulleresque geodesic domes, and it evolved from simple toy to versatile modeling tools through years. Although we can use it to construct the complex structure, it's not intuitive for naive users to use and meanwhile time-consuming.

Tools are developed to help users to design the Zometool structures, e.g., vZome [30] and ZomeCAD [8]. These systems provide different ways to grow the structure. ~~However, the difficulties remain when it comes to build a complex structure because it lacks the ability to provide useful suggestions about what kinds of structure to use next in order to build the target shape.~~ However, they cannot provide useful suggestions about what kinds of structure to use next to build the target shape. The difficulties remain when building a complex structure.

~~This~~ Based on the above reason, it motivates works toward automatic construction through computational method. Zimmer [39, 38] approximate and realize freeform surface automatically using Zometool. Zimmer [38] adopt an incremental panels growing strategy to approximate the surface without self-collisions. On the other hand, Zimmer [39] first build up a rough initial Zometool structure and explore the modification space using local operations. ~~The final Zometool structure is obtained by a stochastic optimization framework.~~ They use a stochastic optimization framework to obtain the final Zometool structure.

2.3 Mesh Segmentation

Mesh Segmentation is an important step for decreasing the complexity of the mesh by splitting the original mesh into many small segments. There are a lot of 3D mesh segmentation algorithms which have been proposed, including K-means [27], graph cuts [3, 9], primitive fitting [1], random walks [16], core extraction [15], spectral clustering [18], critical point analysis [17], Shape Diameter Function [26], and SVM planar cut [34].

In our method, we intend to segment shape so that the resulted segments are printable by the 3D printer, and meanwhile is able to be connected to the internal Zometool structure. We achieve this by designing a two stages process that combines the advantages of MRF-based method [3] and SVM planar cut [34]. ~~The final segmented parts can be easily assembled with optimized Zometool structure.~~ The segmented parts are assembled with the optimized Zometool structure to form the final result.

3 Overview

Given a 3D shape, our method automatically generates inner Zometool structure and outer 3D-printed shells. Our method has the following features:

1. **Large Object.** Our method aims for fabricating the large-scale object which it's height is from 0.3 to 1 meters, which is way larger than the printing volume of ~~common~~ ordinary consumer-level 3D printer.

2. **Fabricability.** Each segment of the outer shell can be printed and fitted inside the volume of consumer-level 3D printers.

3. **Assemblability.** The inner Zometool structure can be easily assembled and connected to the outer printed shells using ~~special~~ specifically designed connectors.

4. **Cost-effectiveness.** Our method maximizes the inner structure and minimize the printing materials. It decreases the fabrication time and the cost of materials.

~~Our method is illustrated in Figure 1.~~ Figure 1 illustrates our method. Given the input shape, we voxelize the inner volume of the input mesh to get the result of an initial Zometool structure. Our goal is to grow the Zometool structure to occupied the maximum inner volume ~~in order~~ to reduce most of the printing materials. Therefore, we design an optimization framework using simulated annealing, and design several local operations to explore the feasible Zometool structure space. After the optimization, we get the maximum inner structure (Section 4).

Next, in order to print the outer shell of input shape, we have to (i) partition the shape into pieces where each piece can be fitted into printing volume, (ii) put connectors at feasible locations so that the inner Zometool structure and the outer shells can be connected robustly, and (iii) keep the salient regions intact. We formulate the partition problem as ~~a~~ an MRF problem and solve it with graph cut algorithm. The intuition of this formulation is that each triangle should be connected to it's closest Zomeball, but simultaneously we want to reduce the number of partitions and maintain the integrity of salient region. To further regularize the resulted partitions, we apply the SVM algorithm to find the hyperplane between different labels and also use the hyperplane for our cut-plane to separate the mesh (Section 5).

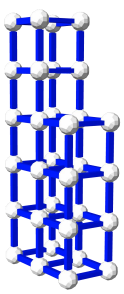
Last, before we separate the mesh by our cut-plane, we generate the inner surface by voxelizing the inner volume, and combine with the outer mesh as the solid mesh. Then, we apply all cut planes on the solid mesh and get the separate pieces. We design the special connector and connect the pieces to the inner Zometool structure. Finally, the pieces with connectors can be printed and assembled to get the final large-scale object (Section 6).

4 Zometool Construction

4.1 Introduction to Zometool

Zometool is widely used as educational toys to replicate complicated scientific structures such as chemical structures. The rods in the standard Zometool system have three different types tenons, i.e. ,blue for rectangles, red for ~~pentagonal~~ pentagons and yellow for triangles. Each type of rod also has three different sizes (see inset). We denote (b_0, b_1, b_2) as three different lengths of blue rods $((r_0, r_1, r_2)$ for red rods, and (y_0, y_1, y_2) for yellow rods). There are ~~totally~~ 62 slots on the Zome-ball, including 30 rectangular slots, 12 pentagonal slots, and 20 triangular slots. Each color of Zometool rods has three sizes, and the ~~growing~~ stretching ratio is related to the golden ratio $\gamma = \frac{1+\sqrt{5}}{2}$. Take blue rods for example, $b_1 = b_0 \cdot \gamma$ and $b_2 = b_0 + b_1$. Red and yellow rods also follow this rule. Moreover, the relative length ratios are different from yellow to blue and red to blue, i.e. , $y_i = \frac{\sqrt{3}}{2} \cdot b_i$ and $r_i = \frac{\sqrt{2+\gamma}}{2} \cdot b_i$. Please refer to [7] for more detail about the underlying mathematical model of Zometool.

4.2 Initialization



Although we can assemble many complicated structures using Zometool, the assemble complexities and time consumption grows ~~pretty~~ fast as the number of Zometool items (rods and balls) used. Our idea is that choosing the simple and repeated unit structure to make up the initial structure, and then growing out from main structure to get better fitting of outer surface. After experimenting different basic structure, such as cubes, triangular pyramids, square ~~based~~ pyramids and pentagonal pyramids, we choose cube as unit building block due to it's smallest assemble time. We follow Zimmer [39] with cube as the basic unit to initialize the filling. The major difference from Zimmer [39] is that we choose b_0 (the length of shortest blue rod) for our edge length of cubes. The reason is that the smaller the cube is, the higher fitting rate we can achieve, and the inner structure will get closer to the outer surface (see inset for sample initialization).

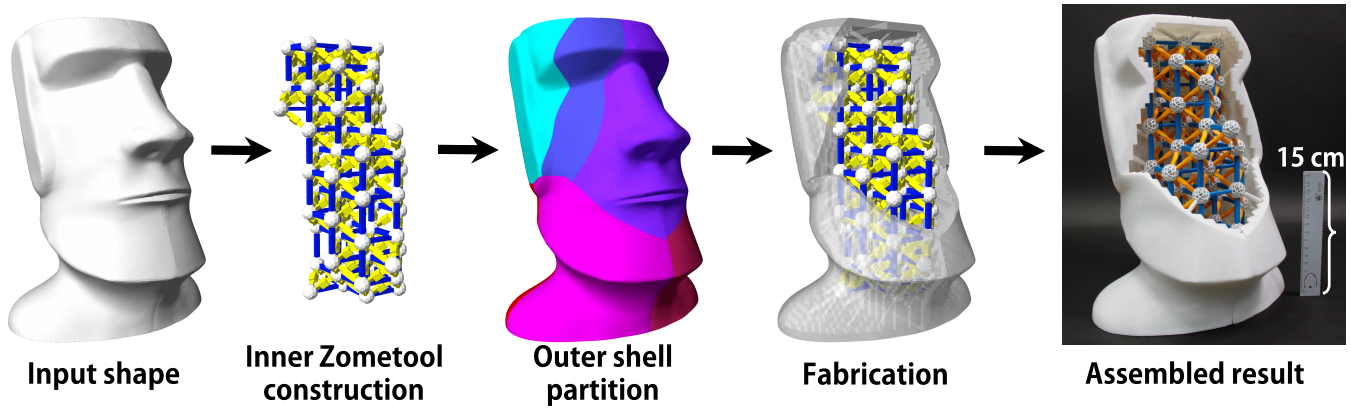


Fig. 1: *Given input shape, we first optimize the inner Zometool structure (Section 4). Guided by optimized Zometool structure, we partition the outer shell (Section 5) and generate connectors for assembling both structures (Section 6). The final fabricated result is obtained by assembling both assembled Zometool structure and printed outer shell.*

4.3 Problem Formulation

We measure the quality of the Zometool structure $\mathbf{Z}$ with an energy E which is composed of 4 terms according to different quality measurements:

$$E(\mathbf{Z}) = w_{\text{dist}} \cdot E_{\text{dist}}(\mathbf{Z}) + w_{\text{reg}} \cdot E_{\text{reg}}(\mathbf{Z}) + w_{\text{val}} \cdot E_{\text{val}}(\mathbf{Z}) + w_{\text{sim}} \cdot E_{\text{sim}}(\mathbf{Z}), \quad (1)$$

(we set w_{dist} = 1.0, w_{reg} = 100.0, w_{val} = 1.0, w_{sim} = 1.0 through all examples shown in this paper.)

4.3.1 Distance

The distance from $\mathbf{Z}$ to S is integrated over all the nodes :

$$E_{\text{dist}}(\mathbf{Z}) = \frac{1}{P \cdot d_{\text{norm}}^2} \sum_{i=1}^{P} \|p_i - \pi(p_i)\|^2 \cdot (1 + F(p_i)), \quad (2)$$

where P is the number of nodes and the normalization factor d_{norm} is used to relate distance to the fixed length of the struts. We follow [39] and define the term $F(p)$ called forbidden zone, which penalizes node points lying too far away from the surface.

Forbidden Zone During energy computation, the distance ~~have~~ has to multiply the additional weight which is called Forbidden zones in [39]. $F(p)$ is defined as a quadratically increasing function which depends on the distance between the triangle centroid position p to the nearest node S. $F(p)$ will be zero when the distance is smaller than d_{min} and will be F_{max} when the distance is larger than d_{max}. After a series of tests, finally, we get the best parameters. We set d_{min} = 16.0 ($\frac{1}{3}$ length of b_0) , d_{max} = 47.3 (length of b_0) and F_{max} = 70.0 for all the examples in this paper.

4.3.2 Regularity

In order to obtain a regular Zometool structure for simple assembling, we intend to regularize the angles between struts to be exact 90° and penalize angle that is too small or too big (see Figure 2).

$$E_{\text{reg}}(\mathbf{Z}) = \frac{1}{|\mathcal{N}_i|} \sum_{p_j \in \mathcal{N}_i} (\min(\theta_{ij}) - \frac{\pi}{2}), \quad (3)$$

where $\mathcal{N}_i$ are the adjacent struts of strut i.

4.3.3 Valence

We regularize the optimized Zometool structure to have a good valence for a simple structure (see Figure 3). ~~In order to~~ To maximize ~~each Zome-ball's utility and minimize the complexity~~ the utility and minimize the complexity of each Zomeball, we have to set a baseline

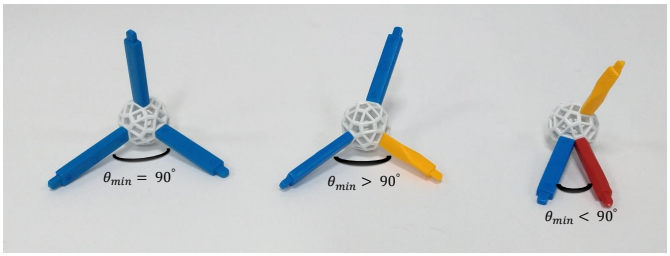


Fig. 2: ***Regularity.*** *We penalize the configuration with minimum angle between struts smaller or greater than* 90°.

number of rods as a constraint. We set the target valence as 6 from the initial cube structure.

$$E_{\text{val}}(\mathbf{Z}) = \sum_{i=1}^{P} \frac{(V_p - 6)^2}{6}, \quad (4)$$

where V_p is the valence of each node.

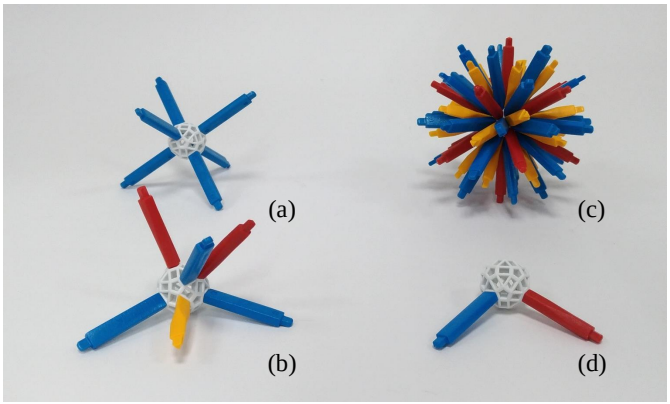


Fig. 3: ***Valence.*** *We encourage the valence of each Zometool node to be 6 (as in configuration (a) and (b)). We penalize the valence that is not 6 (configuration (c) and (d)).*

4.3.4 Simplicity

Let N is the total number of both nodes and struts, and N_{target} be the target complexity. The simplicity term is encoded as the quadratic

differences from the target complexity:

$$E_{\text{sim}}(\mathbf{Z}) = \frac{1}{N_{\text{target}}}(N - N_{\text{target}})^2, \quad (5)$$

4.4 Exploration Mechanism

Searching for the Zometool structure to minimize the energy $E(\mathbf{Z})$ (Eq. 1) is a ~~non trivial~~ non-trivial optimization problem since $E(\mathbf{Z})$ is ~~non convex~~ non-convex and contains global terms. Exhaustive search is impractical and thus we adopt a more scalable strategy based on the Metropolis-Hastings algorithm [11]. In a nutshell, this algorithm makes a random exploration of the solution space by iteratively perturbing the current solution with a certain probability depending on the energy variation between the two solutions and a relaxation parameter T. Following, we describe our local perturbation operators and relaxation scheme. Algorithm 1 details the major steps of our optimization algorithm.

Algorithm 1 Exploration mechanism

Input: Initialized Zometools $\bar{\mathbf{Z}}$,
relaxation parameter $T = T_{init}$
Output: Optimized Zometoos $\mathbf{Z}$
procedure EXPLORATION($\mathbf{Z}$)
  repeat
    generate $\mathbf{Z}'$ from $\mathbf{Z}$ with a random local operation.
    draw a random value $p \in [0,1]$
    if $p < \exp(\frac{E(\mathbf{Z})-E(\mathbf{Z}')}{T})$ and CollisionFree($\mathbf{Z}$) **then**
      update $\mathbf{Z}' \leftarrow \mathbf{Z}$
    end if
    Update $T \leftarrow C \times T$ ▷Update temperature.
  until $T < T_{end}$
end procedure

4.4.1 Local Perturbation Operation

During the exploration, we proposed four local perturbation operations (Figure 4) to construct the Zometool structure by minimizing Eq. 1.

Split This operator inserts a new node and two rods to split the original rod.

Merge This operator inserts a new rod to merge two disconnected nodes (two nodes can travel by two edges).

Bridge This operator inserts a new rod to merge two disconnected nodes (two nodes ~~can't~~ cannot travel by two edges).

Kill This operator deletes a node and two rods.

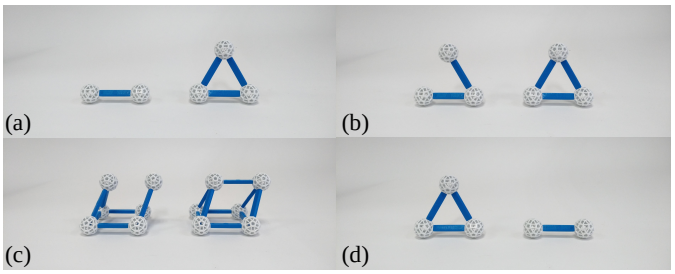


Fig. 4: *We use four local operations during structure perturbation. (a) Split, (b) Merge, (c) Bridge, and (d) Kill.*

Operation validity The operation will make the structure have some difference from the previous round. However, not every round is valid. Our method of judgement which is collision detect is very simple. If the rods or Zome-ball get too close, the collision will happen and the structure will not be assembled. So this procedure will check whether ~~the~~ distance of Zome-balls to Zome-balls, rods to Zome-balls and rods to rods are too close.

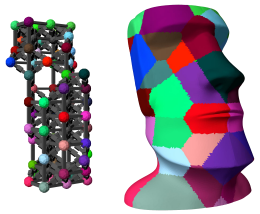
4.4.2 Cooling schedule

The relaxation parameter T, referred as temperature, controls both the speed and the quality of exploration. Start from an initial temperature T_{init}, we decrease the temperature close to zero as iteration tends to infinity. The decreasing process is referred as cooling, and different cooling schedules ~~are~~ exists ~~for~~ for experiments. Although the global minimum convergence is guaranteed using logarithmic cooling schedule [24], we rely on geometric cooling schedule [12]. In our experiment, we set the initial temperature $T_{\text{init}} = 1$, and the decrease rate $C = 0.99$ after every 100 iterations.

5 Object Partition

ichao: Figure ?? and Figure 6 can be merged into one.

As mentioned ~~in~~ at the beginning of this paper, most of the consumer-level 3D printers have limited printing volumes. In order to print our large-scale object, it is necessary to decompose the object into different partitions. Meanwhile, unlike traditional surface partition methods that only take surface features into consideration, we also need to consider the relationship between the outer surface partitions and the inner optimized Zometool structure. More specifically, we will place connectors between the outer and inner structure in order to connect them (we will describe the design of the connector in more detail in Section ??). Naively, we can simply compute the distance from each triangle t to all the nodes in $\mathbf{Z}$, and assign t to the nearest node as it's label. However, inconsistency may arise among adjacent triangles, leading to unsatisfactory visual effects and assembly complexities (numerous small partitions might exist, see inset).



To address this issue, we formulate the problem as a multi-label graph cut minimization. As each triangle t can potentially correspond to different Zometool nodes, it gets assigned data cost for different corresponding nodes. Given n elements, k labels and $n \cdot k$ costs, finding the minimum assignment is a combinatorial problem and typical NP-hard. We employ Boykov [3] to solve it.

After the partition, we further regularize the boundaries between pieces by performing a multi-class classification. The ~~major~~ reason is that we seek smooth boundaries in order to ease the assembling process. We follow Wang [34] and use the Support Vector Machine (SVM) [5] classifier to find our cut-plane. We will describe the detail formulation and implementation of MRF problem and how to find the cut planes in the following paragraph.

5.1 Surface Partition

Optimization energy We compute the assignment function f that assign labels to each triangle t, where $t \in T$, such that the labeling f minimize the following energy $E(f)$:

$$E(f) = w_{data} * \sum_{t \in T} D(t, f_t) + w_{smoothness} * \sum_{t,s \in N} S(t, s, f_t, f_s), \quad (6)$$

and we optimize this function using multi-label graph-cut algorithm proposed by Boykov [3]. In our setting, the entire outer nodes of $\mathscr{Z}$ are complete possible label set L. This $E(f)$ consists of two separate terms, i.e., data and smoothness. Next, we will explain each term in more detail.

5.1.1 Data cost

Data cost measures how well a triangle t covers a node $p \in P$. This cost is simply defined as the distance of the nearest node to the triangle.

$$D(t, f_t) = -\omega \log(\mathscr{P}(p|t)), \quad (7)$$

where $\mathscr{P}(p|t)$ is the probability of that triangle t belong to the node label p, and ω is a constant that regulates the influence of the data term in the total energy. Here, we simply define $\mathscr{P}(p|t)$ as $1/d(t,p)$, where $d(t,p)$ is the distance of the node to the triangle.

5.1.2 Smoothness cost

Smoothness term measures the spatial consistency of neighboring elements.

$$S(t,s,l_t,l_s) = \begin{cases} 0, & \text{if } l_t = l_s, \\ -\log(\theta_{t,s}/\pi)\varphi_{t,s} + w_{saliency} * E_{saliency}(t), & \text{otherwise} \end{cases} \quad (8)$$

where $\theta_{p,q}$ and $\varphi_{p,q}$ are the dihedral angle and distance between triangle p and q, respectively. With the smoothness term, two adjacent triangles are likely to have consistent labels.

User-guided saliency We observed that there are many salient regions on each of the object we used, e.g., we don't want the partition seam to go through the eyes and nose on the face ~~Figure 5~~ (see Figure 5). ~~In order to~~ To preserve the salient region, we ask users to draw the region he thinks it's important to preserve, and we formulate this as part of the smoothness cost, which prevents ~~the partition seam~~ partition seams to cut through salient region ~~Figure 5~~ (Figure 5)..

$$E_{saliency}(t) = \begin{cases} 1, & \text{triangle is marked as salient region,} \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

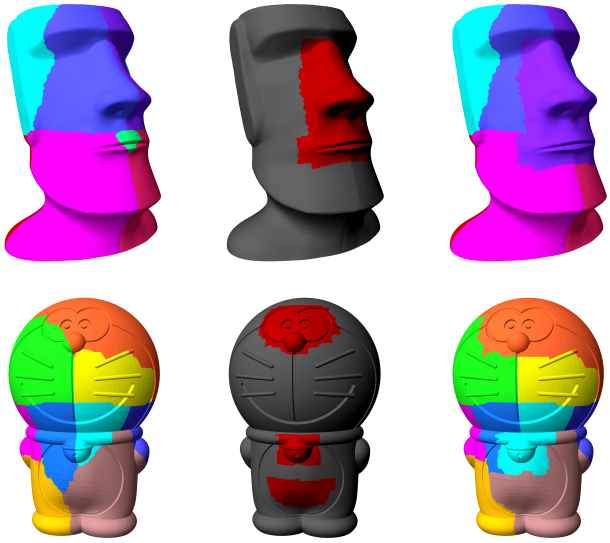


Fig. 5: *User-guided saliency. Origin result of graph cut may split some feature on the mesh. (Left of the figure) In order to solve it, we ask user to mark the unique feature. (Red region in the middle of the figure) We add those information for graph cut, and precent those regions from separating into multiple parts. (Right of the figure)*

5.1.3 Our result

We solve the labeling problem Eq. 6 with graph-cut and obtain 11 labels on the Maoi head object Figure 6(a). Compare to the naive method (Figure **??**), we obtain much smaller partition numbers (11 v.s. 60), and each of them ~~are~~ is with better shape and size.

5.2 Object Cut

In order to cut the physical object into pieces, we have to find the cut planes that separate the space occupied by the object. However, the boundaries between optimized partitions from the previous step ~~is~~ are not regularized enough for directly used as the separating plane. It is very difficult to find a plane in 3D space because of the varied plane normal vectors. ~~In order to~~ To obtain the desired cut planes, we analogize our problem to finding the separating plane in solving multi-class classification using Support Vector Machine (SVM). We use each triangle as a data sample, with it's location as the feature vector and the optimized label from the previous step as it's class. In short, SVM algorithm intend to find the best hyperplane which is the one that represents the largest separation, or margin, between the two adjacent classes. ~~And~~ Therefore, we use the obtained hyperplanes from SVM as our cut planes (Figure 6(b)).

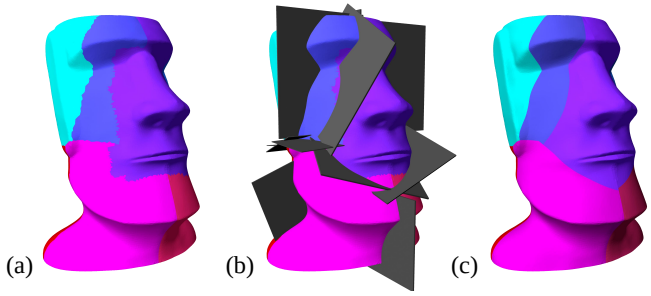


Fig. 6: *(a) Result of graph cut (b) Cut-plane generated by SVM classifier (c) Cutting result*

6 Fabrication

Chi: here need some discussion. The object to be printed should be solid mesh~~, i.e. we have to fill in the inner space of the object with materials as well.~~ . In our method, we use the Zometool structure to fill in most of the inner spaces, while still retaining the minimum thickness of the outer shell ~~due to the common setting of 3D printers.~~ to form the input shape. The minimum thickness is defined due to the common setting of 3D printers. Thus, we prepare ~~object~~ segmented solid meshes to be fabricated by generating ~~a inner surface~~ pieces with a predefined thickness, and attach connectors on ~~this inner surface to combine the inner and outer structure.~~ those pieces to combine the Zometool structure.

6.1 Inner surface

To construct the solid mesh as the outer shell, we need to obtain the inner surface and use the original surface as the outer surface. There are many potential ways to generate the inner surface. The simplest method is that shrinking the mesh along the vertex normals. However, it sometimes generates the surface with flipped triangles that stick out of the outer surface.

To prevent this problem, we instead voxelize the original object, and remove the voxels that cover the original surface. ~~And~~ Moreover, we use the ~~outer surface~~ outermost surface of the remaining voxels as our inner surface for solid mesh (Figure 7).

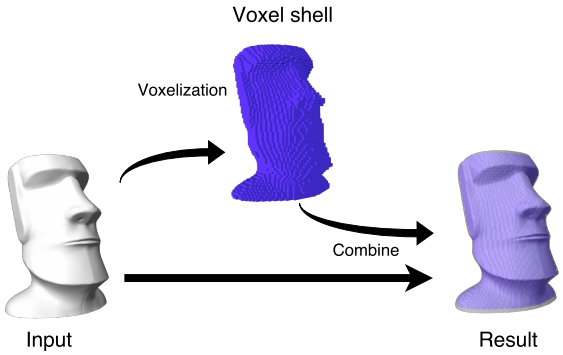


Fig. 7: *Inner surface generation method.*

6.2 Generate connector

With the inner surface, we need to place connectors on it to connect between inner Zometool structure and outer shells. Two potential designs for building these connectors are:

1. Dig holes on the surface and use the Zometool rods to connect both inner and outer structure (Figure 8 (a)).

2. Grow Zometool tenons on the inner surface instead (Figure 8 (b)).

We will elaborate how we implement both designs and discuss the pros and cons of both designs in the following paragraph.

6.2.1 Dig holes

As observed, naively digging holes on the inner surface is likely to break the outer surface. Instead, we have to grow additional structure that replicates Zome-ball, which is called "virtual ball" on the inner surface. And we dig holes on this virtual ball to connect with the Zometool structure using ~~z~~ Zometool struts. However, as the objects usually printed with support materials generated by 3D printers, we observed that these support materials ~~will greatly~~ would hugely reduce the quality of these holes. The major reason is that the holes are filled with the support materials and it is ~~really hard~~ quite difficult to ~~clean~~ remove all of them. As a result, the Zometool struts cannot be inserted well (Figure 8 (a)). To address this issue, practically we can change the printing directions, e.g., place the holes face upward to the printing plate so that no support materials will be printed inside the holes. However, this means that the outer surface is attached to the support materials, and the final printed appearance will be pretty poor. The issue cannot be handled well given the limited precision of consumer-level 3D printer. Hence, we propose an alternative connector design.

6.2.2 Grow tenons on surface

Zometool's tenon is a very small object, it fit perfectly on the Zome-ball and makes the structure pretty robust. However, the size of tenon is a strong challenge for the 3D printer due to it's limited precision. Beside, same objects will be printed differently under different orientations because the way of support printing. In order to verify our 3D printer (Ultimaker 3 [1]) is able to print the tenons, we design an exhaustive experiment as follow: we use Ultimaker 3 to print three different tenon of Zometool (rectangle, pentagon, triangle), each print in twenty-seven directions. As a result, we found out that even under the lowest precision ("fast print" mode in Ultimaker 3), the printed tenons can still perfectly fit into the slots on the Zometool balls. Compared to dig holes on the inner surface, it is also easier to ~~clean~~ remove the printed support materials on the printed tenons. Given it's easier ~~clean up~~ cleanup and more robust structure, we use this design to connect the inner Zometool structure and outer shells (see Figure 8 (b)).

And we decide how many tenons on each outer shell with the following process: We shoot rays from each slot on a single Zome-balls in the optimized Zometool structure **S**, and record whether it intersects with the surface. We repeat this examination on all of the 62 slots on the Zome-balls, and we grow tenons on the surface along the direction with the most ray-surface intersections.

7 Result

7.1 Experiment environment

We implement ZomeFab in C++ and Python on desktop PC with 3.4GHz CPU and 16GB memory. The outer surfaces are printed by Ultimaker 3, a low-cost FDM 3D printer with 0.2m x 0.2m x 0.2m printing volume and PLA material. The time it takes to assemble the fabricated parts for these objects range from few minutes to around an hour, depending on the number of parts.

[1] https://ultimaker.com/en/products/ultimaker-3

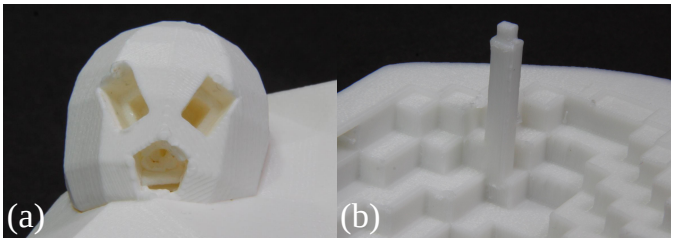


Fig. 8: *Connector design : Dig holes on inner surface (a) and grow tenons on surface (b). We can see that the materials in the digged hole can not be cleaned well in (a), so the Zometool struts can not be inserted well.*

7.2 Evaluation

We evaluate the material cost and fabrication time between "Zomefab" and solid-printed object (denoted as "baseline" method). We use CURA [6], a slicer software for 3D printing, to simulate the fabrication time and used materials for solid-printed object. Meanwhile, we subdivide the shape using a octree, and stop the subdivision once each sub-shape fit into the printing volume. To evaluate our method, we report two different infill methods called "hollow" and "solid". Shapes printed under "hollow" use only 20% infill rate, where "solid" use 100% infill rate. Note that we only use 20% infill rate to fabricate the results shown in this paper, and the "solid" results are simulated using CURA.

7.2.1 Material cost

As shown in Table 1, our method greatly saves materials from 24% to 64% under "hollow" setting, and from 68% to 85% under "solid" setting. As expected that our method brings more benefits when infill rate grows. The material cost are listed as follow: 0.56 USD/meter (Ultimaker official PLA material price), Zometool rod: 0.19 USD/rod and Zome-ball: 0.29 USD/ball (Zometool official price) for our experiments.

7.2.2 Printing time

We report printing time evaluation under both single 3D printer scenario and multiple 3D printer scenario. In multiple 3D scenario, we assume there are enough 3D printers so that we can print all pieces simultaneously. In Table 2, we demonstrate the statistical comparison of printing time between single 3D printer and multiple 3D printers. In single 3D printer scenario, the fabrication time of our method under "hollow" configuration is longer than the baseline method. The main reason is because the pieces of baseline method fit the printing volume better (regularly cutted), where the printing paths can be optimized without the time waste on printing redundant support materials. And the pieces of our method is highly non-regular with connectors, so the printing paths is more complicated compare to baseline pieces. Note that as the infill rate grows, we can discover that the growing rate of our method is way less than the baseline method, and is able to save from 52% to 68% of printing time. In multiple 3D printers scenario, we assume that we have enough 3D printers, so the reported time of our method is the longest printing time among all pieces. The statistics shows that our method printing time is absolutely faster than baseline in different fill rate using multiple 3D printers. If the user just has single 3D printer and wants to get a exquisite result, our method can also get lower cost and fabrication time.

7.3 Zometool Use

Table 3 shows the quantity of Zometool rods and balls used in each result. In Section 4, we use smallest blue rods to make the zome-cube as the unit structure in order to make the best-fit initial structure. The bigger the object is, more smallest blue rods (b_0) will be used. We also observed that, blue rods and yellow rods are more likely to be used interchangeably since we encourage regularity during simulated annealing optimization. Meanwhile, the rule of Zometool indicates

that the longest rod only can be replaced by one shortest and middle rod in the same color. So it seldom introduce red rods into the structure, which resulted in the used number of rods imbalance shown in Table 3.

Except the smallest yellow and blue rods, the other rods are used few or none in our simulated annealing rule.

REFERENCES

[1] M. Attene, B. Falcidieno, and M. Spagnuolo. Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer*, 22(3):181–193, 2006.

[2] M. Bächer, E. Whiting, B. Bickel, and O. Sorkine-Hornung. Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 33(4):96:1–96:10, 2014.

[3] Y. Boykov and V. Kolmogorov. An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision. *IEEE transactions on pattern analysis and machine intelligence*, 26(9):1124–1137, 2004.

[4] P. Cignoni, N. Pietroni, L. Malomo, and R. Scopigno. Field-aligned mesh joinery. *ACM Trans. Graph.*, 33(1):11:1–11:12, 2014.

[5] C. Cortes and V. Vapnik. Support-vector networks. *Machine learning*, 20(3):273–297, 1995.

[6] Cura: 3d printer slicing application. https://ultimaker.com/en/products/ultimaker-cura-software.

[7] T. Davis. The mathematics of zome, 2007.

[8] E.Schlapp. ZomeCAD. [online].

[9] A. Golovinskiy and T. Funkhouser. Randomized cuts for 3d mesh analysis. In *ACM transactions on graphics (TOG)*, page 145. ACM, 2008.

[10] J. Hao, L. Fang, and R. E. Williams. An efficient curvaturebased partitioning of largescale stl models. *Rapid Prototyping Journal*, 17(2):116–127, 2011.

[11] W. K. Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[12] D. Henderson, S. H. Jacobson, and A. W. Johnson. *The Theory and Practice of Simulated Annealing*. Springer US, Boston, MA, 2003.

[13] R. Hu, H. Li, H. Zhang, and D. Cohen-Or. Approximate pyramidal shape decomposition. *ACM Trans. Graph.*, 33(6):213:1–213:12, Nov. 2014.

[14] Y. Huang, J. Zhang, X. Hu, G. Song, Z. Liu, L. Yu, and L. Liu. Framefab: Robotic fabrication of frame shapes. *ACM Trans. Graph.*, 35(6):224:1–224:11, Nov. 2016.

[15] S. Katz, G. Leifman, and A. Tal. Mesh segmentation using feature point and core extraction. *The Visual Computer*, 21(8-10):649–658, 2005.

[16] Y.-K. Lai, S.-M. Hu, R. R. Martin, and P. L. Rosin. Fast mesh segmentation using random walks. In *Proceedings of the 2008 ACM symposium on Solid and physical modeling*, pages 183–191. ACM, 2008.

[17] H.-Y. S. Lin, H.-Y. M. Liao, and J.-C. Lin. Visual salience-guided mesh decomposition. *IEEE Transactions on Multimedia*, 9(1):46–57, 2007.

[18] R. Liu and H. Zhang. Segmentation of 3d meshes through spectral clustering. In *Computer Graphics and Applications, 2004. PG 2004. Proceedings. 12th Pacific Conference on*, pages 298–305. IEEE, 2004.

[19] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, and B. Chen. Build-to-last: Strength to weight 3d printed objects. *ACM Trans. Graph.*, 33(4):97:1–97:10, July 2014.

[20] L. Luo, I. Baran, S. Rusinkiewicz, and W. Matusik. Chopper: Partitioning models into 3D-printable parts. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)*, 31(6), Dec. 2012.

[21] S.-J. Luo, Y. Yue, C.-K. Huang, Y.-H. Chung, S. Imai, T. Nishita, and B.-Y. Chen. Legolization: Optimizing lego designs. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia 2015)*, 34(6):222:1–222:12, 2015.

[22] H. Medelln, T. Lim, J. Corney, J. Ritchie, and J. Davies. Automatic subdivision and refinement of large components for rapid prototyping production. *Journal of Computing and Information Science in Engineering*, 7(3):249–258, 9 2007.

[23] R. Prévost, E. Whiting, S. Lefebvre, and O. Sorkine-Hornung. Make It Stand: Balancing shapes for 3D fabrication. *ACM Transactions on Graphics (proceedings of ACM SIGGRAPH)*, 32(4):81:1–81:10, 2013.

[24] P. Salamon, R. Frost, and P. Sibani. *Facts, conjectures, and improvements for simulated annealing*. Society for Industrial and Applied Mathematics, 2002.

[25] A. Shamir, B. Bickel, and W. Matusik. Computational tools for 3d printing. In *ACM SIGGRAPH 2016 Courses*, SIGGRAPH '16, pages 9:1–9:34, New York, NY, USA, 2016. ACM.

[26] L. Shapira, A. Shamir, and D. Cohen-Or. Consistent mesh partitioning and skeletonisation using the shape diameter function. *The Visual Computer*, 24(4):249, 2008.

[27] S. Shlafman, A. Tal, and S. Katz. Metamorphosis of polyhedral surfaces using decomposition. In *Computer graphics forum*, pages 219–228. Wiley Online Library, 2002.

[28] 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 Transactions on Graphics (SIGGRAPH 2016)*, 35(4), 2016. Article 45.

[29] P. Song, C.-W. Fu, and D. Cohen-Or. Recursive interlocking puzzles. *ACM Transactions on Graphics (SIGGRAPH Asia 2012)*, pages 128:1–128:10, December 2012.

[30] S.Vorthmann. vZome [online].

[31] N. Umetani, A. Panotopoulou, R. Schmidt, and E. Whiting. Printone: Interactive resonance simulation for free-form print-wind instrument design. *ACM Trans. Graph.*, 35(6):184:1–184:14, Nov. 2016.

[32] J. Vanek, J. A. G. Galicia, B. Benes, R. Mech, N. A. Carr, O. Stava, and G. S. P. Miller. Packmerger: A 3d print volume optimizer. *Comput. Graph. Forum*, 33(6):322–332, 2014.

[33] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, and X. Liu. Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph.*, 32(6):177:1–177:10, 2013.

[34] W. M. Wang, C. Zanni, and L. Kobbelt. Improved surface quality in 3d printing by optimizing the printing direction. In *Computer Graphics Forum*, pages 59–70. Wiley Online Library, 2016.

[35] R. Wu, H. Peng, F. Guimbretière, and S. Marschner. Printing arbitrary meshes with a 5dof wireframe printer. *ACM Trans. Graph.*, 35(4):101:1–101:9, July 2016.

[36] M. Yao, Z. Chen, L. Luo, R. Wang, and H. Wang. Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph.*, 34(6):214:1–214:11, Oct. 2015.

[37] Q. Zhou, J. Panetta, and D. Zorin. Worst-case structural analysis. *ACM Trans. Graph.*, 32(4):137:1–137:12, July 2013.

[38] H. Zimmer and L. Kobbelt. Zometool rationalization of freeform surfaces. *IEEE Transactions on Visualization and Computer Graphics*, 20(10):1461–1473, Oct 2014.

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

Mesh	Infill Method	Fabrication Method	Material Cost (USD)			Efficiency (Saved)
			3D printing	Zometool	Overall (sum)	Material
Moai	Hollow	Zomefab	82.56	70.57	153.13	13.83%
		Baseline	177.71		177.71	
	Solid	Zomefab	170.9	70.57	241.47	65.11%
		Baseline	692.07		692.07	
Squirrel	Hollow	Zomefab	100.38	77.55	177.93	21.74%
		Baseline	227.37		227.37	
	Solid	Zomefab	213.96	77.55	291.51	66.17%
		Baseline	861.66		861.66	
Doraemon	Hollow	Zomefab	101.38	76.34	177.72	21.91%
		Baseline	227.59		227.59	
	Solid	Zomefab	227.78	76.34	304.12	64.25%
		Baseline	850.79		850.79	
Totoro	Hollow	Zomefab	72.32	61.48	133.80	25.20%
		Baseline	178.88		178.88	
	Solid	Zomefab	148.26	61.48	209.74	77.44%
		Baseline	657.31		657.31	
Iron Man	Hollow	Zomefab	73.99	57.76	131.75	34.12%
		Baseline	199.97		199.97	
	Solid	Zomefab	149.67	57.76	207.43	72.93%
		Baseline	766.24		766.24	

Table 1: *ZomeFab's performance on saving material as compared to a baseline method.*

Mesh	Infill Method	Fabrication Method	Single 3D Printer Fabrication Time (hours)			Multiple 3D Printer Fabrication Time (hours)			Efficiency (Saved)	
			3D printing	Zometool	Overall (sum)	3D printing	Zometool	Overall (max)	Single	Multiple
Moai	Hollow	Zomefab	293.97	2.5	296.47	72.58	2.5	75.08	30.70%	22.50%
		Baseline	434.08		434.08	96.88		96.88		
	Solid	Zomefab	523.20	2.5	525.70	108.13	2.5	110.63	77.75%	68.80%
		Baseline	2362.28		2362.28	354.57		354.57		
Squirrel	Hollow	Zomefab	355.05	3.0	358.05	53.82	3.0	56.82	22.04%	8.49%
		Baseline	459.28		459.28	62.07		62.07		
	Solid	Zomefab	643.65	3.0	646.65	93.97	3.0	96.97	76.57%	77.94%
		Baseline	2759.72		2759.72	439.65		439.65		
Doraemon	Hollow	Zomefab	356.60	4.0	360.60	53.83	4.0	57.83	37.88%	39.45%
		Baseline	580.48		580.48	88.90		88.90		
	Solid	Zomefab	643.65	4.0	647.65	93.97	4.0	97.97	57.71%	77.74%
		Baseline	1531.42		1531.42	440.13		440.13		
Totoro	Hollow	Zomefab	273.68	3.0	276.68	29.37	3.0	32.37	27.76%	31.46%
		Baseline	383.02		383.02	47.23		47.23		
	Solid	Zomefab	468.82	3.0	471.82	51.67	3.0	54.67	77.68%	83.78%
		Baseline	2113.42		2113.42	337.07		337.07		
Iron Man	Hollow	Zomefab	282.30	2.0	284.30	47.57	2.0	49.57	34.98%	18.67%
		Baseline	437.25		437.25	60.95		60.95		
	Solid	Zomefab	477.03	2.0	479.03	72.12	2.0	74.12	80.65%	79.92%
		Baseline	2475.22		2475.22	369.10		369.10		

Table 2: *ZomeFab's performance on time as compared to a baseline method.*

Mesh	Zometool										Printing pieces
	Blue			Red			Yellow			Ball	
	S	M	L	S	M	L	S	M	L		
Moai	112	0	0	0	0	0	140	0	0	73	8
Squirrel	144	22	0	8	4	0	119	3	1	85	13
Doraemon	143	26	1	2	5	1	89	1	1	87	15
Totoro	95	0	0	0	0	0	137	0	0	60	18
Iron Man	93	0	0	0	0	0	124	0	0	57	10
Owl	115	0	0	0	0	0	159	0	0	70	15
Pig	61	12	0	10	8	0	53	6	0	47	10
Slime	132	0	0	0	0	0	182	0	0	80	14
Lion	78	0	0	0	0	0	101	0	0	50	8
Bunny	215	0	0	0	0	0	266	0	0	126	10

Table 3: *Material usage for each result. Including Zometool and printing pieces.*

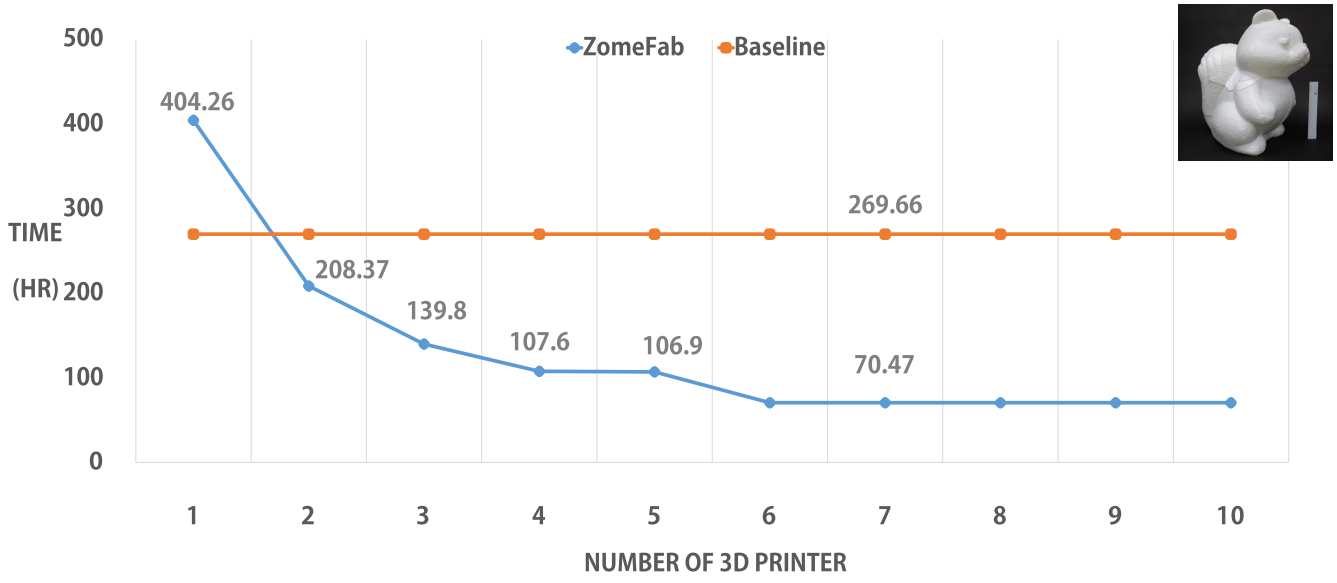


Fig. 9: *Printing time of Squirrel with different number of printers. If user use single 3D printer to print the result, our method's printing time will be more than baseline. The chart shows if the number of 3D printers are more than 2, the printing time will lower than baseline. If user has more than 6 3D printers, the printing time only has to wait the pieces which has the longest printing time.*

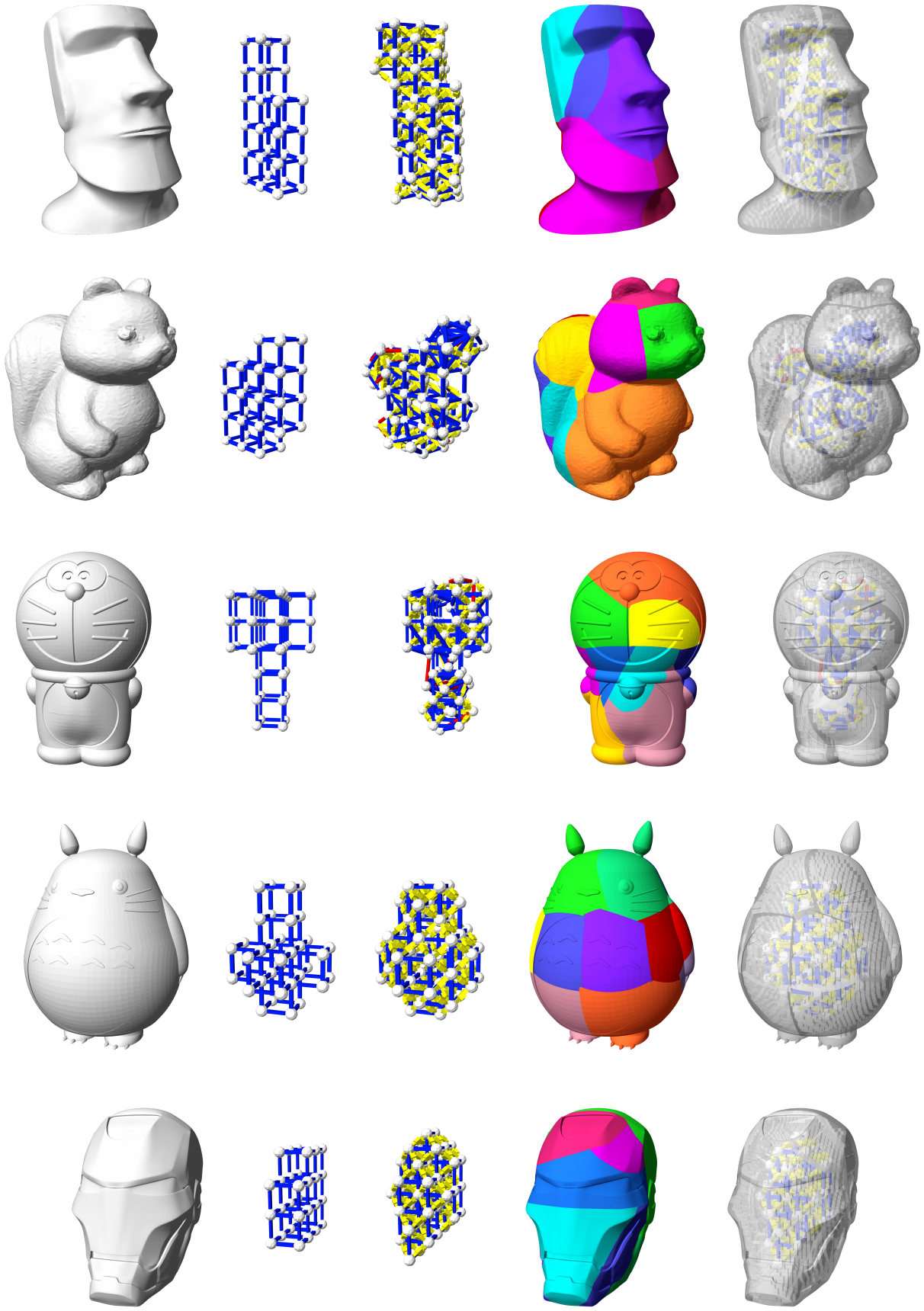


Table 4

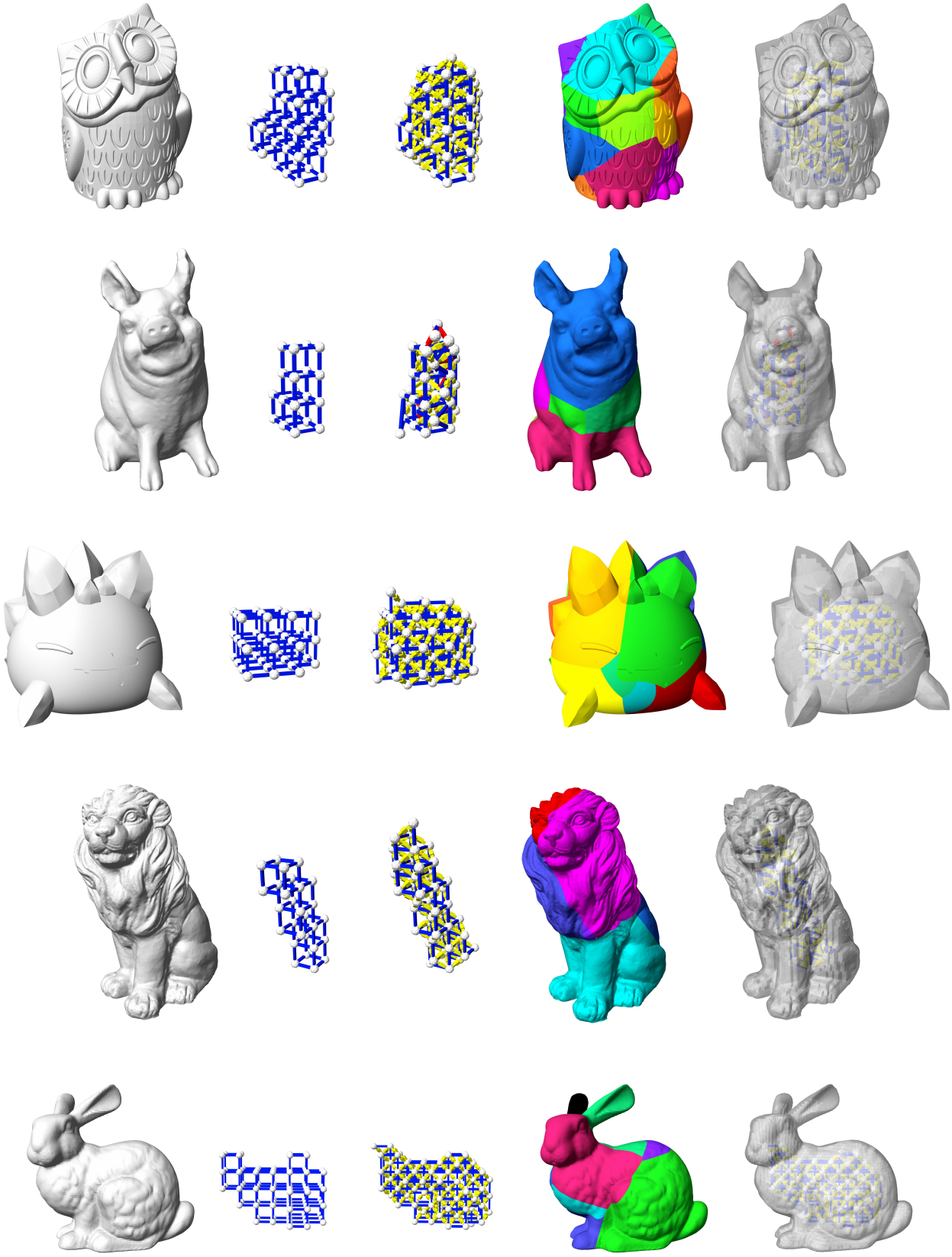


Table 5