In this paper, we present a novel method, called ZomeFab, automatically generate both outer shell and inner Zometool structures approximating a given 3D input model. We first performs mesh segmentation to split the complex 3D model into different shape parts. For each part, we fit the smallest cube of Zometool structure as initial inner structure. From the initial Zometool structure, we use Simulated Annealing algorithm to effectively explore the huge structure space. Then, with the optimized Zometool structure, we hollowed the shape to obtain the outer shell, and partition the outer shell with respect to several criteria including simplicity and printability. We formulate these criteria in a single MRF problem and solved it using graph cut algorithm. We also design a special types of connecters and optimize their positions for assembling both structures.

There are two primary contributions in this paper:

1. The proposed method formulate an optimization to compute the inner Zometool structure. Filled by Zometool structure instead of solid printed is greatly reduce the printing cost including time and materials.
2. We design and print a special connector and optimize their layout for better combine both inner Zometool structure and outer printed shell.

The rest of this paper is organized as follows. In Section 2, we surveyed several previous methods for computational fabrication and applications of modeling system Zometool. Section 3, Section 4 and Section 5 describe the three main stages of our method: Zometool construction, surface partition and Fabrication, respectively. In Section 6, we demonstrate results of our method. Finally, Section 7 concludes this paper.

2 Related Work

Computational Fabrication In recent years, computational fabrication has attracted many attentions in the computer graphics and human computer interaction research fields [Shamir et al. 2016]. Numerous works are proposed to fabricate shapes (i) with different objectives, e.g. balance [Prévost et al. 2013; Bächler et al. 2014], size [Luo et al. 2012], structure soundness [Zhou et al. 2013] and sounds [Umetani et al. 2016], and (ii) using different materials or building blocks, e.g. Lego [Luo et al. 2015], planar slices [Cignoni et al. 2014], and interlocking puzzles [Song et al. 2012]. Although the fast development 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 et al. [2016] and Wu et al. [2016] design devices and algorithms to print shapes in wireframe. Meanwhile, different internal structures are developed, the skin-frame structure by Wang et al. [2013], the honeycomb-like structure by Lu et al. [2014], and 2D laser cutting shape proxy by Song et al. [2016]. To enable the large shape to be printed using 3D printers, Luo et al. [2012] developed an iterative planar-cut method, aiming to fit decomposed parts in the 3D printing volume while considering factors such as assemblability and aesthetics. Yao et al. [2015] proposed a level-set framework for 3D shape partition and packing. Compared to these works, our method construct a shape with Zometool structure and 3D-printed parts, so that we can reduce the fabrication time and cost, with the reusability of Zometool structures.

Zometool Design and Modeling Zometool is a mathematically-precise plastic construction set for building a myriad of geometric structures, from simple polygons to visualize and model various natural sciences, e.g. DNA molecules. It’s history dates back to the 1960s where it started out as a simple construction system inspired by Buckminster Fuller’s geodesic domes, and it evolved from simple toy to versatile modeling tools through years. Although we can use it to construct 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 [S.Vorthmann.] and ZomeCAD [E.Schlapp.]. 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 kind of structure to use next in order to build the target shape. This motivates works toward automatic construction through computational method. Zimmer et al. [2014; 2014b] approximate and realize freeform surface automatically using Zometool. Zimmer et al. [2014b] adopt an incremental panels growing strategy to approximate the surface without self-collisions. On the other hand, Zimmer et al. [2014] 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.

Figure 2: Given input shape (a), we initialize the Zometool structure with cubes (b), and obtain the optimized result (c) using Simulated Annealing as described in Section 3.

3 Zometool construction

Our method is aiming for devising an algorithm to construct an object composed of zometool structures and 3D-printed parts with the following objectives:

- **material-effectiveness** We should aim to minimize the overall fabrication cost and time. Since Zometool is substantially faster to build and reusable, we should maximize its usage to reduce 3D printing material in the fabrication.
- **easy-to-assemble** We should reduce the difficulties of assembling both Zometool structure and outer printed shell. In terms of Zometool, we should minimize the usage of nodes and rods, so to reduce the assembly time.

3.1 Initialization

We first partition $S$ into $m$ segments ($S = \{s_1, \cdots, s_m\}$) using Shape Diameter Function (SDF) [Shapira et al. 2008] and clustering implemented in CGAL [CGA]. First we want to fill the inner volume of each segment $s_i$. Although we can use some existing works [Zimmer and Kobbelt 2014b] to generate the inner zometool structure, we intend to use structure that is composed of simple primitives because it’s simplicity and easy-assemblability. Follow Zimmer et al. [2014], we choose to use cube as the basic primitive to initialize the filling. The initialized Zometool structure is denoted as $\widehat{S} = \{Z_0, \cdots, Z_n\}$, where $n$ is the number of segments with embedded zome tools (see Figure 2(b) for sample initialization).

3.2 Problem Formulation

We measure the quality of the zometools model with an energy $E$ composed of 4 terms accounting to different quality measurements:

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

(1)

3.2.1 Distance

The distance from $Z$ to $S$ is integrated over all the nodes :

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

(2)

where $P$ is the number of nodes and the normalization factor $d_{\text{norm}}$ is used to relate distance to the fixed length of the structs. We
follow [Zimmer et al. 2014] and define the term \( F(p) \) forbidden zone, which to penalize node points lying too far away from surface. Please see [Zimmer et al. 2014] for more details.

### 3.2.2 Regularity

In order to obtain a regular Zometool structure for simple assembling, we intend to regularize the angles between struts to be exact 90° (see Figure 3).

\[
E_{\text{reg}}(Z) = \frac{1}{|N|} \sum_{p \in N} (\min(\theta_{ij}) - \frac{\pi}{2}),
\]

(3)

### 3.2.3 Valence

We regularize the optimized Zometool structure to have a good valence for simple structure (see Figure 4). We set the target valence as 6 from the initial cube structure.

\[
E_{\text{val}}(Z) = \sum_{i=1}^{P} (V_{p} - 6)^{2},
\]

(4)

where \( V_{p} \) is the valence of each node.

**Figure 3:** Regularity. We penalize the configuration with minimum angle between struts less or greater than 90°.

**Figure 4:** 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)).

### 3.2.4 Simplicity

Let \( N \) be the total number of both nodes and struts, and \( N_{\text{target}} \) be the target complexity. The simplicity term is simply encoded as the quadratic differences from the target complexity:

\[
E_{\text{sim}}(Z) = \frac{1}{N_{\text{target}}} (N - N_{\text{target}})^{2},
\]

(5)

### 3.3 Exploration Mechanism

Searching for the Zometool structure to minimize the energy \( E(Z) \) (Eq. 1) is a non trivial optimization problem since \( E(Z) \) is non convex and contains global terms. Exhaustive search is impractical and thus we adopt a more scalable strategy based on the Metropolis-Hastings algorithm [Hastings 1970]. 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

1: Input: Initialized Zometools \( Z \),
2: relaxation parameter \( T = T_{\text{init}} \)
3: Output: Optimized Zometool \( Z \)
4: procedure \( \text{EXPLORATION}(Z) \)
5: repeat
6: \( Z' \) from \( Z \) with a random local operation.
7: \( p \in [0, 1] \)
8: if \( p < \exp(\frac{E(Z) - E(Z')}{T}) \) and CollisionFree(\( Z \)) then
9: \( Z' \leftarrow Z \)
10: end if
11: Update \( T \leftarrow C \times T \) \( \triangleright \) Update temperature.
12: until \( T < T_{\text{end}} \)
13: end procedure

**Figure 5:** We use four local operations during structure perturbation. (a) Split, (b) Merge, (c) Bridge, and (d) Kill.

#### 3.3.1 Local Perturbation Operation

During the exploration, we proposed four local perturbation operations (Figure 5) to construct the Zometool structure by minimizing Eq. 1.

- **Split** This operator insert a new node and two rods to split the original rod.
- **Merge** This operator insert a new rod to merge two disconnected nodes (two node can travel by two edges).
- **Bridge** This operator insert a new rod to merge two disconnected nodes (two node can’t travel by two edges).
- **Kill** This operator delete a node and two rods.

#### 3.3.2 Cooling schedule

The relaxation parameter \( T \), referred as temperature, controls both the speed and the quality of exploration. Start from initial temperature \( T_{\text{init}} \), 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 experiment. Although the
global minimum convergence is guaranteed using logarithmic cooling schedule [Salamon et al. 2002], we rely on geometric cooling schedule [Henderson et al. 2003]. In our experiment, we set the initial temperature $T_{\text{init}} = 1$, and the decrease rate $C = 0.99$ after every 100 iterations.

## 4 Surface Partition

We aim to decompose the surface into different partitions for 3D printing, and to cover the optimized Zometool structure $Z$ from Section 3. Naively, we can simply compute the distance from each triangle $t$ to all the nodes in $Z$, and assign $t$ to the nearest node as its label. However, inconsistency may arise among adjacent triangles leading to unsatisfactory visual effects (Figure 2) and assembly complexities (numerous small partitions might exist). 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 node, 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 [Boykov and Kolmogorov 2004] to solve it.

### 4.1 Optimization energy

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

$$E(f) = \sum_{t \in T} D(t, f_t) + \sum_{t, s \in \mathcal{N}} S(t, s, f_t, f_s),$$

(6)

and we optimize this function using multi-label graph-cut algorithm proposed by Boykov et al. [Boykov and Kolmogorov 2004]. In our setting, the entire outer nodes of $Z$ are complete possible label set $\mathcal{L}$. This $E(f)$ consists of three separate terms, i.e. data, smoothness and label costs. Next, we will explain each term in more detail.

### 4.2 Data cost

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

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

(7)

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

### 4.3 Smoothness cost

Smoothness term measures the spatial consistency of neighboring elements.

$$S(t, s, f_t, f_s) = \begin{cases} 0, & \text{if } l_t = l_s, \\ -\log(\theta_{t,s}/\pi)\varphi_{t,s}, & \text{otherwise} \end{cases}$$

(8)

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

## 5 Fabrication

We partitioned the surface of the input shape as pieces for fabrication in Section 4. However, the input mesh just have the outer surface, the valid pieces will have both outer and inner surface for 3D printer. After we have the inner surface, the pieces have to generate connection to the zometool structure. After the process we can get all pieces can fabricate, and then we assembled them by connecting to the optimized Zometool structure. In order to generate fabricatable shape, it is practically leave a minimum thickness of the outer shell, and this thickness differs from printer to printer. Thus, we start to prepare the shape to be fabricated by generating a inner surface with a predefined thickness, and the remaining operations are performed within this inner surface.

### 5.1 Inner surface

There are many potential ways to generate inner surface. The simplest method is shrink the mesh along the vertex normals. Although it is simple, it solely sometimes leads to the triangle flip over the surface. In order to prevent this problem, we use the voxelized mesh to handle it. First, we shrink the mesh along vertex normals with one radius of zometool ball. Second, voxelize the shrinked mesh. Finally, choose the outer surface of the voxelized result and combine the original mesh then we get the new mesh have outer and inner surface. The inner surface generation process is illustrated in Figure 6.

![Figure 6: Inner surface generation method.](image)

### 5.2 Generate connector

After get the inner surface, we are able to use the partitioned results from graph cut to cut the surface into pieces. Still, we need to connect the inner Zometool surface with these cutted pieces. Two potential methods for building these connectors are: (i) we can dig holes on the surface and use the Zometool sticks to connect both inner and outer structure (Figure 7(a)), (ii) we can grow Zometool tenons on the surface instead (Figure 7(b)).

#### 5.2.1 Dig holes

If we want to dig some connecting holes on surface, we can’t directly dig it because it might break the outer surface. To deal with it, we can put a new ball we called ”virtual ball” on the surface. Then we can dig the hole on the virtual ball and thus the outer shell will not be broken. The downside of this approach is that shapes are usually printed with support materials, and the dugged hole are usually filled with these materials Due to the low precision issues for most of the consumer 3D printers, it is impractical to generate clean holes for Zometool tenon to plug in. Hence, we design an alternative approach to generate the connectors.
5.2.2 Grow tenons on surface

Zometool’s tenon is a very small object, fit perfectly on the zometool ball and make the structure be very strong. However, the size of tenon is a strong challenge for the 3D printer due to its low precision. Beside, same object will be printed differently under different orientations because the way of support printing. In order to verify our printer (Ultimaker 3) 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 (Figure 8). 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 surface, it is also easier to clean the printed support materials on the printed tenons. We search the nearest node on inner Zometool structure on split piece by graph cut for each triangle. There are sixty-two slots on the Zometool ball, it means we can get sixty-two directions to grow the tenons. We choose the direction which can grow most tenons on the surface to make a strong connection on the inner Zometool structure.

6 Results

Table 1: Material usage for each result. Including Zometool and printing pieces.

<table>
<thead>
<tr>
<th>Model</th>
<th>Blue</th>
<th>Red</th>
<th>Yellow</th>
<th>Ball</th>
<th>Printing pieces</th>
</tr>
</thead>
<tbody>
<tr>
<td>Maoi</td>
<td>S</td>
<td>M</td>
<td>L</td>
<td>S</td>
<td>M</td>
</tr>
<tr>
<td>Squiral</td>
<td>122</td>
<td>17</td>
<td>3</td>
<td>12</td>
<td>4</td>
</tr>
</tbody>
</table>


Figure 7: Experiment: (a) dig hole (b) grow tenon on surface.

Figure 8: Experiment: print different types of zometool struts: (a) rectangle (b) pentagon (c) triangle. We designed an experiment to make sure the 3D printer can print tiny object such as the Zometool tenon in all possible configurations. And the same size tenon won’t be deformed by different printing route.

6.1 Experiment environment

We implement ZomeFab in C++ and execute it on desktop PC with 3.4GHz CPU and 16GB memory. The assemble process are showed in Figure 9 and Figure 10. The inner structure is build by Zometool which can construct a strong structure easily. The outer surface are printed by Ultimaker3, a low-cost FDM printer with 0.2m x 0.2m x 0.2m printing volume and PLA material. Table 1 shows the using amount of Zometool and printing pieces.

6.2 Evaluation

We evaluate the material cost and fabrication time between Zomefab and solid mesh with simple partition. We use CURA (version 2.3.1), a slicer software for 3D printing, help us evaluate it. The software will evaluate the fabrication time and amount of material if the volume of input mesh is printable. In Table 1, we demonstrate four experiments for each model, print hollowed with 20% infill and print solid by Zomefab and baseline that partitions a solid object into 3D printable parts. All of experiments show that our method cost less material than baseline. However, the fabrication time will more than baseline in 20% infill rate. We propose that our mesh’s complexity is higher than baseline testing mesh. It will cause the printing route be complicated. On the other hand, if the user want to get a exquisite result, our system can get lower cost and fabrication time.
<table>
<thead>
<tr>
<th>Model</th>
<th>Infill Method</th>
<th>Fabrication Method</th>
<th>Material Cost (US$)</th>
<th>Fabrication Time (hours)</th>
<th>Efficiency (Saved)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>3D printing</td>
<td>Zometool</td>
<td>Overall (sum)</td>
<td>3D printing</td>
</tr>
<tr>
<td>Maoo</td>
<td>Hollow</td>
<td>Zomefab</td>
<td>110.43</td>
<td>110.43</td>
<td>130.25</td>
</tr>
<tr>
<td></td>
<td>Solid</td>
<td>Baseline</td>
<td>183.89</td>
<td>183.89</td>
<td>110.43</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zomefab</td>
<td>95.15</td>
<td>70.57</td>
<td>165.72</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Baseline</td>
<td>693.96</td>
<td>693.96</td>
<td>639.00</td>
</tr>
<tr>
<td>Squirrel</td>
<td>Hollow</td>
<td>Zomefab</td>
<td>438.48</td>
<td>3.0</td>
<td>441.48</td>
</tr>
<tr>
<td></td>
<td>Solid</td>
<td>Baseline</td>
<td>693.96</td>
<td>693.96</td>
<td>639.00</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Zomefab</td>
<td>83.71</td>
<td>70.57</td>
<td>154.28</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Baseline</td>
<td>76.12</td>
<td>76.12</td>
<td>76.12</td>
</tr>
</tbody>
</table>

Table 2: ZomeFab’s performance on saving material and time as compared to a baseline method.

Figure 9: Result fabricated and assembly : Maoi

7 Conclusion

ZomeFab is a system which combines Zometool and 3D printing to fabricate a large-scale 3D object. In this approach, we aimed to represent an input 3D model by an inner structure and pieces of outer surfaces. The inner structure is greatly saving cost of 3D printing compare to directly print partitioned input model. The outer surface still remain the fine geometric characteristic of 3D printer. Thanks to the Zometool reusability, the long term cost of fabrication quietly decreased. However, Zometool is a complex geometric system which can build thousands of structure. Our method generate a Zometool result which can simply build and also well fit outer surface. User can get the pretty well result by using Zomefab.

7.1 Limitation and Future work

Zomefab relied on the input mesh which have the large inner volume and using Zometool as inner structure. The smallest Zometool cube (4.7cm x 4.7cm x 4.7cm) is limited. Therefore, in order to get the initial structure have to make sure input mesh’s inner volume must larger than one unit cube. In the future, we can change the unit cube into different and smaller shape to get better structure fitting inside the mesh.

8 致謝

本論文感謝科技部經費補助，計畫編號：MOST103-2212-E-002-158-MY3 與 MOST105-2218-E-002-011。

References


E. Schlappe. ZomeCAD. [online].


Figure 10: Result fabricated and assembly: Squirrel

ACM Transactions on Graphics (Proc. SIGGRAPH Asia) 31, 6 (Dec.).


S. VORTHMANN. vZome [online].


