Masonry Shell Structures with Discrete Equivalence Classes

RULIN CHEN, Singapore University of Technology and Design, Singapore
PENGYUN QIU, Singapore University of Technology and Design, Singapore
PENG SONG, Singapore University of Technology and Design, Singapore
BAILIN DENG, Cardiff University, United Kingdom
ZIQI WANG, ETH Zürich, Switzerland
YING HE, Nanyang Technological University, Singapore

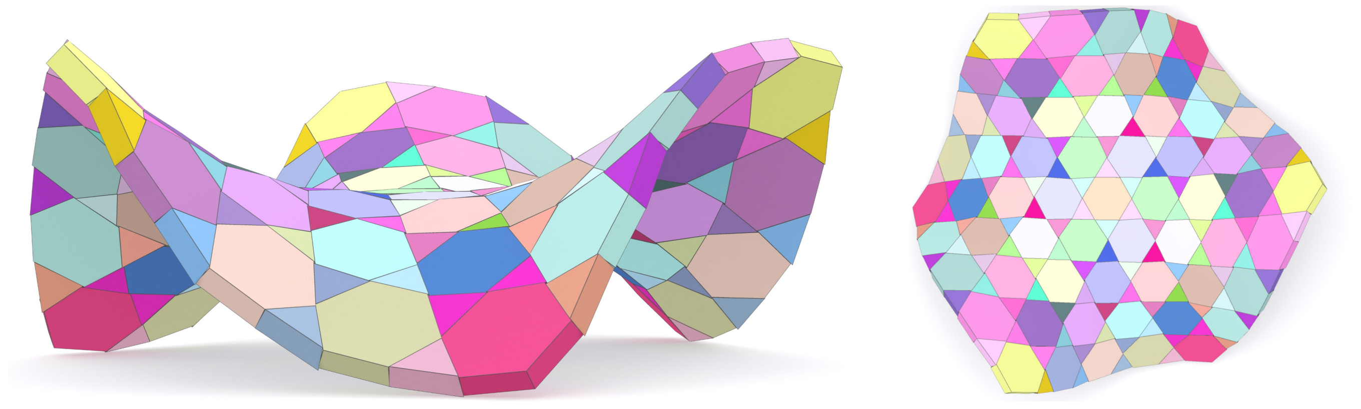
Fig. 1. *A masonry shell structure modeled by our approach, in which 181 shell elements fall into 60 discrete equivalence classes. Shell elements in the same class have exactly the same shape and are rendered with the same color.*

This paper proposes a method to model masonry shell structures where the shell elements fall into a set of discrete equivalence classes. Such shell structure can reduce the fabrication cost and simplify the physical construction due to reuse of a few template shell elements. Given a freeform surface, our goal is to generate a small set of template shell elements that can be reused to produce a seamless and buildable structure that closely resembles the surface. The major technical challenge in this process is balancing the desire for high reusability of template elements with the need for a seamless and buildable final structure. To address the challenge, we define three error metrics to measure the seamlessness and buildability of shell structures made from discrete equivalence classes and develop a hierarchical cluster-and-optimize approach to generate a small set of template elements that produce a structure closely approximating the surface with low error metrics. We demonstrate the feasibility of our approach on various freeform surfaces and geometric patterns, and validate buildability of our results with four physical prototypes. Code and data of this paper are at https://github.com/Linsanity81/TileableShell.

CCS Concepts: • **Computing methodologies** → *Shape modeling*.

Authors' addresses: Rulin Chen, Singapore University of Technology and Design, Singapore, rulin_chen@mymail.sutd.edu.sg; Pengyun Qiu, Singapore University of Technology and Design, Singapore, pengyun_qiu@sutd.edu.sg; Peng Song, Singapore University of Technology and Design, Singapore, peng_song@sutd.edu.sg; Bailin Deng, Cardiff University, United Kingdom, dengb3@cardiff.ac.uk; Ziqi Wang, ETH Zürich, Switzerland, ziqi.wang@inf.ethz.ch; Ying He, Nanyang Technological University, Singapore, yhe@ntu.edu.sg.

Additional Key Words and Phrases: shell structure, discrete equivalence classes, geometry optimization

ACM Reference Format:
Rulin Chen, Pengyun Qiu, Peng Song, Bailin Deng, Ziqi Wang, and Ying He. 2023. Masonry Shell Structures with Discrete Equivalence Classes. *ACM Trans. Graph.* 42, 4, Article xxx (August 2023), 12 pages. https://doi.org/10.1145/3592095

1 INTRODUCTION

Masonry shell structures have been built for centuries around the world in the form of arches, domes and vaults, due to their advantages like large span, lightweight, and high strength [Adriaenssens et al. 2014]. The shapes of masonry shells are generally described by 3D curved surfaces since the thickness of these structures is significantly smaller compared to its width and length. From a geometric perspective, a masonry shell is a geometric tiling of a 3D surface with a number of shell elements that contact one another with no overlaps and no gaps. To tile a freeform 3D surface, it is likely that each shell element has a unique shape when compared to the other elements, requiring each element to be custom manufactured.

In this paper, we study a new problem of modeling freeform shell structures where the shell elements fall into a set of discrete equivalence classes. These shell structures have clear advantages of reducing the fabrication cost and simplifying the physical construction due to reuse of template shell elements. We model our shell elements as convex polyhedrons since convex shapes and planar faces make it easy to fabricate, e.g., by molding. Shell elements falling into the same discrete equivalence class should have exactly

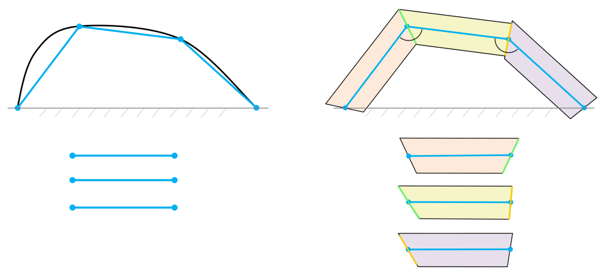

Fig. 2. *Given an input surface (in black), we can approximate it with three polygons (in blue) with the same shape. However, three shell elements generated from the three polygons, respectively, with the same thickness may have different shapes due to different side faces (in green and orange) of the elements.*

the same shape. That means the shell elements should have the same top face that defines the shell's appearance as well as the same side faces that are used for contacting with other elements in the final structure. Existing approaches [Fu et al. 2010; Liu et al. 2021; Singh and Schaefer 2010] for modeling freeform surfaces using discrete equivalence classes of polygons are insufficient to address our problem since they ignore the shell elements' thickness and do not model the elements' side faces; see Figure 2 for a 2D illustration.

Given a 3D freeform surface as an input, our goal is to compute a small set of template shell elements whose instances can produce a *seamless* and *buildable* shell structure that closely resembles the surface. The major challenge in our problem arises from two conflicting goals. On the one hand, we want to use a small set of template shell elements to build the freeform shell structure, aiming to maximize reusability of the templates. On the other hand, we aim for a seamless and buildable tiling of the freeform surface with the template elements, where planar contacts between instance elements of the templates should be preserved and gaps as well as overlaps between the instance elements should be minimized. This seamless and buildable tiling is achieved in conventional shell structures by using a large number of template elements, where each template is used only for once in many cases (i.e., no reuse of the template).

To address the challenge, we make the following technical contributions:

- We parameterize the geometry of a shell structure with discrete equivalence classes, and define three error metrics to measure their seamlessness and buildability.
- We propose a hierarchical approach to clustering shell elements in the parameter space, which is able to cluster elements of various shapes including triangles, quads, n-gons, and a mix of them.
- We develop a hierarchical approach to optimizing the geometry of clustered shell elements, enabling to significantly reduce the number of templates while preserving the final structure's seamlessness and buildability.

2 RELATED WORK

Modeling surfaces with discrete equivalence classes. One typical application of modeling surfaces with discrete equivalence classes is to rationalize freeform architectural surfaces. To simplify the problem, existing works assume a fixed topology of the input surface with only triangles [Huard et al. 2015; Singh and Schaefer 2010] or quads [Fu et al. 2010], and generate the discrete equivalence classes by clustering polygons and optimizing mesh vertex positions. Instead of directly reusing the polygons, other works aim to reuse molds that are used to fabricate curved panel elements [Eigensatz et al. 2010] or triangle-based point-folding elements [Zimmer et al. 2012], in which elements of the same shape are fabricated with molding and then cut into different sizes and forms for making the architectural surface. The above works optimize specific discrete equivalence classes for different input surfaces, a process known as *post-rationalization* [Austern et al. 2018]. In contrast, Liu et al. [2021] addressed the *pre-rationalization* problem of modeling various input surfaces using predefined discrete equivalence classes of triangles by developing a fabrication-error-driven remeshing algorithm.

Similar to the above works that post-rationalize a freeform surface [Fu et al. 2010; Huard et al. 2015; Singh and Schaefer 2010], our modeling approach also involves clustering and optimizing polygons in a freeform surface to minimize intra-cluster variance. However, there are two differences. First, we propose a hierarchical approach to clustering and optimizing polygons in a freeform surface with various topologies, including surfaces with triangles, quads, n-gons, and a mix of them. Second, our modeling approach clusters and optimizes not only the polygons but also the dihedral angles between the polygons, aiming to facilitate the clustering of shell elements in the later stage.

Modeling structures with discrete equivalence classes. Modeling 3D structures with predefined discrete equivalence classes, also called tileable blocks [Wang et al. 2021b], is a well-known problem. A number of computational approaches have been developed to approximate a given 3D shape with Lego bricks [Luo et al. 2015; Testuz et al. 2013] or a Zometool construction set with nine struts of different lengths and one universal joint [Zimmer and Kobbelt 2014; Zimmer et al. 2014]. These tileable blocks, such as universal building blocks [Chen et al. 2018] and the Zometool construction set [Shen et al. 2020] have also been used to build infill structures for cost-effective 3D fabrication. Besides these well-known tileable blocks, researchers have developed new tileable blocks for modeling 3D structures, including interlocking voxels [Zhang and Balkcom 2016], SL blocks [Shih 2016], and various space filling blocks [Akleman et al. 2020; Krishnamurthy et al. 2020; Subramanian et al. 2019].

Rather than developing predefined tileable blocks for pre-rationalization, Brütting et al. [2021] addressed a post-rationalization problem of designing a bespoke kit of parts with beams of different lengths and ball-like joints for modeling multiple input frame structures. Compared with pre-rationalization, post-rationalization allows to model structures with various forms without being restricted by the shapes of predefined tileable blocks. In our paper, we address a new post-rationalization problem: computing a small set of template elements to model a seamless and buildable shell structure that closely resembles a given freeform surface.

Optimizing masonry shell structures. Optimization of masonry shell structures is an active area of research in the computer graphics community. Some researchers [Wang et al. 2019; Whiting et al. 2009,

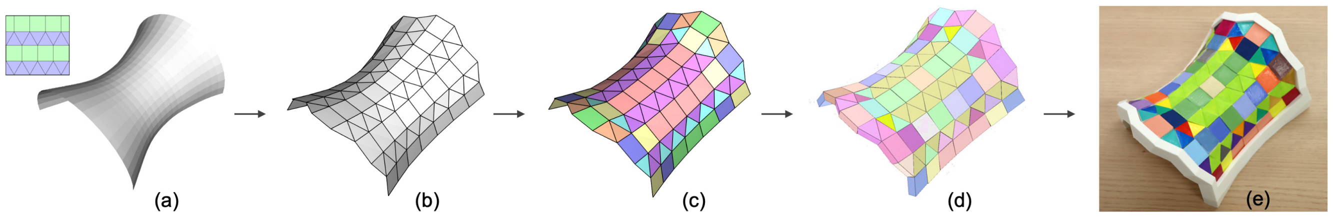

Fig. 3. *Overview of our approach. (a) Given a 3D guiding surface and a 2D tessellation, (b) we initialize a base polygonal mesh* **T** *by mapping the 2D tessellation onto the 3D surface using the as-rigid-as-possible algorithm [Liu et al. 2008]. (c) Next, we optimize the mesh vertices to cluster the polygons as well as dihedral angles between the polygons. (d) Finally, we optimize the augmented angles to cluster the shell elements for modeling a shell structure* **M′** *with discrete equivalence classes. (e) A 3D printed prototype that validates the structure's buildability.*

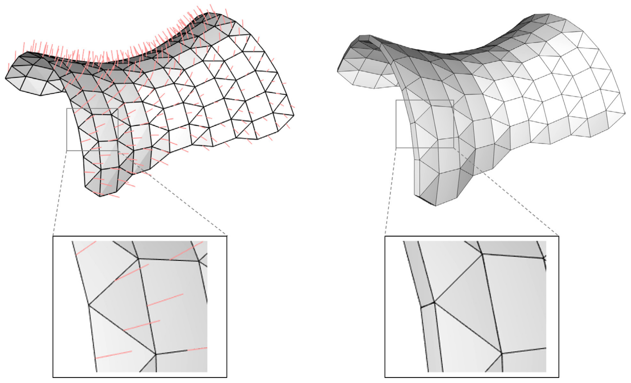

Fig. 4. *Parameterizing the geometry of (right) a shell structure using (left) a base polygonal mesh with augmented vectors (in red).*

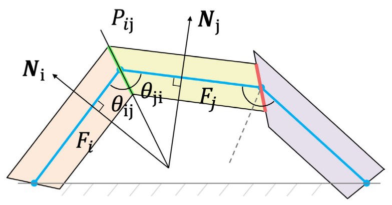

Fig. 5. *Two constraints on the augmented angles* $\{\theta_{ij}\}$*; see Equations 1 and 2.*

2012] directly optimize geometry of shell structures to guarantee its structural stability while others [de Goes et al. 2013; Liu et al. 2013; Ma et al. 2019; Miki et al. 2015; Panozzo et al. 2013; Tang et al. 2014; Vouga et al. 2012] design self-supporting shapes using geometry processing methods. Different from the above works that optimize for structural stability, we are the first to optimize the geometry of shell structures for *reusability of the shell elements*. While self-supporting is outside the scope of this paper, the above approaches can be potentially used in combination with our method to create self-supporting shell structures that are cost-effective to produce in practice.

3 MODELING SHELL STRUCTURES

In this section, we model and parameterize the geometry of a shell structure with discrete equivalence classes, and propose three error metrics to evaluate seamlessness and buildability of the shell structure.

Parameterizing shell structures. We parameterize the geometry of a shell structure **M** with *convex* elements by using a 3D polygonal mesh **T** where each edge is augmented with a vector; see Figure 4. Given a 3D freeform surface **S**, we first remesh it to obtain a 3D polygonal mesh **T**, called a *base mesh*, in which each polygonal face is planar and convex. Specifically, we obtain the base mesh **T** by parameterizing the surface **S** using the as-rigid-as-possible algorithm [Liu et al. 2008] and then mapping a 2D tessellation with convex polygons onto the surface. Since the tessellation is unbounded in 2D, users are allowed to adjust its location, orientation, and scale, relative to the surface **S**, to specify its portion that is mapped to the surface [Song et al. 2013]. Users are also allowed to choose a desired 2D tessellation for the mapping, and the 2D tessellation is preferred to be a monohedral, dihedral, or trihedral tiling to limit the number of distinct tiles in the tessellation; see Figure 3(a&b).

We construct a shell element S_i for each face F_i in the mesh **T**. To represent the side faces of shell elements, we augment each edge $\mathbf{e}_{ij}$ in the base mesh **T** with a unit vector $\mathbf{n}_{ij}$ that is orthogonal to $\mathbf{e}_{ij}$; see Figure 4 (left). Each edge $\mathbf{e}_{ij}$ of the mesh **T** together with the augmented vector $\mathbf{n}_{ij}$ define a 3D partitioning plane P_{ij}, whose normal is $\mathbf{e}_{ij} \times \mathbf{n}_{ij}$. For each face F_i, we intersect all the 3D planes associated with its edges to construct convex geometry of the corresponding shell element S_i. Since the intersected geometry is generally infinite, we trim each element S_i using offset planes with normal $\pm\mathbf{N}_i$ and offset distance $\pm 0.5\tau$, where $\mathbf{N}_i$ is the normal of face F_i and τ is the shell thickness; see Figure 4 (right).

According to our modeling strategy, each side face of a shell element induces a dihedral angle $\theta_{ij} \in [0, \pi)$ between the base polygon F_i and the partitioning plane P_{ij}; see Figure 5. We call these dihedral angles $\{\theta_{ij}\}$ *augmented angles*. In particular, these augmented angles are subject to the following constraint:

$$\theta_{ij} + \theta_{ji} = \alpha_{ij} \tag{1}$$

where $\alpha_{ij} \in [0, 2\pi)$ is the dihedral angle between two neighboring base polygons F_i and F_j; see again Figure 5. Moreover, when choosing each partitioning plane P_{ij}, it has to be close to the plane bisecting the dihedral angle α_{ij}. This makes the shell structure approximate the guiding surface nicely with planar shell elements that

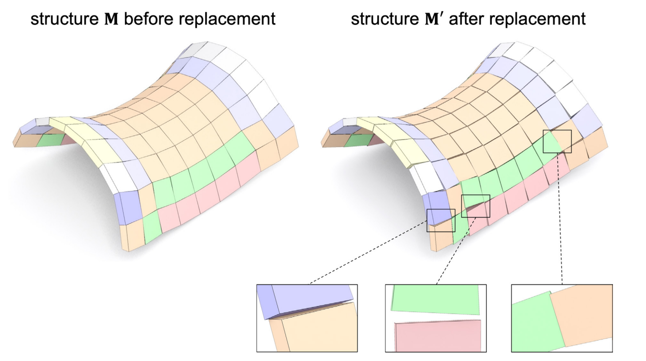
structure **M** before replacement
structure **M′** after replacement

Fig. 6. *We (left) cluster elements in a shell structure* **M** *and (right) model a shell structure* **M′** *with discrete equivalence classes by replacing each clustered element in the structure* **M** *with a template, resulting in undesirable contacts, gaps, and overlaps between the elements (see the zooming views).*

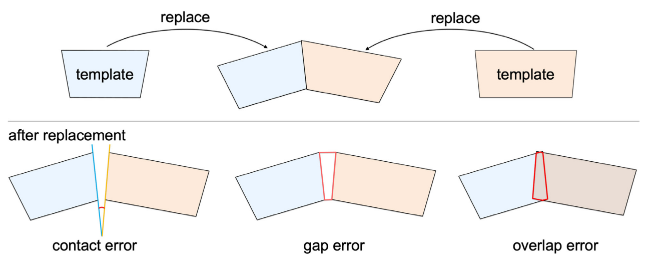

Fig. 7. *We propose (bottom) three error metrics to evaluate seamlessness and buildability of shell structures where (top) the original elements are replaced with a small set of templates.*

have the same thickness. This constraint is formulated as:

$$\left|\theta_{ij} - \alpha_{ij}/2\right| < \beta \tag{2}$$

where β is a threshold set as 10° in our experiments. Figure 5 shows two partitioning planes in green and red respectively, where the red plane does not satisfy Equation 2. This results in visibility of the purple element's side face, hurting the structure's aesthetics.

Thanks to our parameterization method, shell elements contact one another exactly with their planar side faces in the structure **M**, making the structure seamless and buildable; see again Figure 4 (right). However, the problem with the structure **M** is that each shell element is likely to have a unique shape, making them unable to be reused. To model a shell structure with discrete equivalence classes, we partition the N shell elements in the structure **M** into K_S clusters according to the elements' shape similarity; see Figure 6 (left). We generate a template shell element for each cluster and replace each original element in the structure **M** with the corresponding template. By doing this, we model a shell structure **M′** with discrete equivalence class; see Figure 6 (right) and Section 4. The reusability of shell elements in the structure **M′** is measured using N/K_S.

Three error metrics. The cost of reusing shell elements is that the structure **M′** may have undesirable contacts, gaps, and overlaps between the elements, making the structure **M′** not seamless and/or not buildable; see again Figure 6 (right). These errors are a result of the shape difference between each original element and the template used for replacement. We propose the following metrics to measure these errors:

- *Contact error* occurs when the two side faces of two adjacent shell elements are not parallel; see Figure 7 (left). We measure the error by computing the dihedral angle between the two side faces.
- *Gap error* is caused by a mismatch between the two side faces of two adjacent elements, resulting in a separation or gap between them; see Figure 7 (middle). We measure the error by computing the gap volume and normalize it by the average volume of all the elements in the structure **M′**.
- *Overlap error* is caused by a mismatch between the two side faces of two adjacent elements, resulting in one element passing through or penetrating the other; see Figure 7 (right). We measure the error by computing the overlap volume and normalize it by the average volume of all the elements in the structure **M′**.

It is worth noting that either an overlap or gap error can occur at each contact, depending on whether two elements penetrate each other at the contact or not. To evaluate these errors, we compute the above three types of errors for each contact in the structure **M′**, and calculate the average and maximum for each error. We consider a shell structure **M′** is seamless and buildable only when the following conditions are satisfied:

$$\begin{aligned} C_{\text{avg}} &< t_{\text{c\_avg}}, \quad C_{\max} < t_{\text{c\_max}} \\ G_{\text{avg}} &< t_{\text{g\_avg}}, \quad G_{\max} < t_{\text{g\_max}} \\ O_{\text{avg}} &< t_{\text{o\_avg}}, \quad O_{\max} < t_{\text{o\_max}} \end{aligned} \tag{3}$$

where C_{avg} and $C_{\max}$ are the contact errors, G_{avg} and $G_{\max}$ are the gap errors, and O_{avg} and $O_{\max}$ are the overlap errors, $t_{\text{c\_avg}}$, $t_{\text{c\_max}}$, $t_{\text{g\_avg}}$, $t_{\text{g\_max}}$, $t_{\text{o\_avg}}$, and $t_{\text{o\_max}}$ are user specified thresholds (set as 2°, 10°, 0.005, 0.05, 0.005, and 0.05 in all our experiments).

In Section 5, we optimize vertices of the base mesh **T** as well as the augmented angles $\{\theta_{ij}\}$ (subject to Equations 1 and 2) consecutively to generate a small set of template elements for producing a shell structure **M′** that satisfies the conditions in Equation 3; see Figure 3(c&d). We show that our modeled shell structure **M′** is buildable by making a physical prototype; see Figure 3(e).

4 CLUSTERING SHELL ELEMENTS

In this section, our goal is to partition the N shell elements $\{S_i\}$ in the structure **M** into K_S clusters according to their shape similarity. A typical approach to clustering shell elements is to define a metric that measures similarity between the shell elements and then to cluster the shell elements based on the metric. We introduce this typical approach, point out its limitations for addressing our problem, and then present a new hierarchical approach to clustering shell elements.

Typical approach and its limitations. Given two shell elements S_i and $\tilde{S}_i$, we denote their base polygons as F_i and $\tilde{F}_i$, respectively. We assume that the two base polygons F_i and $\tilde{F}_i$ are planar and have the same number of vertices, denoted as L. We measure the similarity of two shell elements S_i and $\tilde{S}_i$ by generalizing the metric

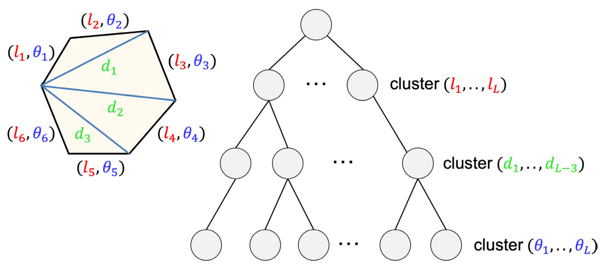

Fig. 8. *Our hierarchical approach to clustering shell elements. Left: the geometry of each shell element is represented by edge lengths $(l_1,..,l_L)$, diagonal lengths $(d_1,..,d_{L-3})$, and augmented angles $(\theta_1,..,\theta_L)$. Right: a three-level hierarchy of clusters computed by our approach, where the leaf nodes represent final clusters of shell elements.*

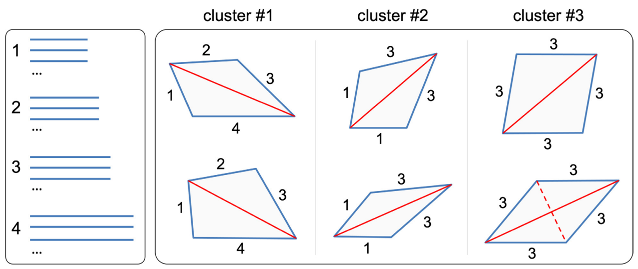

Fig. 9. *Our hierarchical approach to clustering planar polygons in a base mesh (i.e., the first two hierarchies of clustering). (Left) We cluster all the edges in the mesh according to their length. (Right) We first cluster polygons accordingly to the assigned edge cluster IDs and then further partition each cluster of polygons into sub-clusters based on the length of the diagonal(s) (in red color).*

of measuring similarity of polygons in [Fu et al. 2010; Singh and Schaefer 2010]:

$$s(S_i, \tilde{S}_i) = \min_k \sum_{l=1}^{2L} \|\mathbf{v}_l - T_k(\tilde{\mathbf{v}}_l)\|^2, \tag{4}$$

where $\mathbf{v}_l$ and $\tilde{\mathbf{v}}_l$ are vertices on shell elements S_i and $\tilde{S}_i$, respectively, and T_k represents a rigid transformation. $s(S_i, \tilde{S}_i) = 0$ indicates that the two shell elements have exactly the same shape. We search for the best possible registration between the two shell elements among the L different ways to correspond vertices from the two elements. Note that we do not allow the transformation that flips the top and bottom faces of one element to correspond to the other element since a shell element's top and bottom faces usually have different appearance in practice. We find the best rigid transformation T_k in Equation 4 using the method of [Arun et al. 1987].

A typical approach to clustering the shell elements is to minimize intra-cluster variances measured using the above metric, e.g., by using k-means clustering. However, this typical approach is not suitable for our problem since it clusters shell elements in the explicit geometry space (i.e., polyhedron) instead of the parameter space (i.e., base polygon with augmented angles). We prefer to cluster shell elements and optimize their geometry in the parameter space due to its low dimension as well as independence among the parameters. In detail, a base polygon's shape can be defined by $2L-3$ parameters where $2L$ parameters define positions of the L vertices of a planar polygon and -3 is due to the three degrees of freedom (1 for rotation and 2 for translation) to transform the planar polygon. Hence, the dimension of the parameter space is $3L-3$, including $2L-3$ parameters that define the base polygon's shape and L augmented angles. In particular, the $2L-3$ parameters of a base polygon are independent from the L augmented angles. In contrast, the dimension of the explicit geometry space is $6L$ since the polyhedral shape has $2L$ 3D vertices. These $6L$ variables are highly correlated due to planarity of the polyhedron's faces as well parallelism between the polyhedron's top and bottom faces.

Our hierarchical approach. We propose a hierarchical approach to clustering shell elements in the parameter space, which does not rely on the similarity metric in Equation 4. Our insight is that shell elements in the same cluster should have the same or similar base polygons (i.e., base polygons are in the same cluster). We represent the $2L-3$ parameters of each base polygon using the lengths of L edges and $L-3$ diagonals shooting from the same vertex; see Figure 8 (left). Similarly, for base polygons in the same cluster, the lengths of their corresponding edges are the same or similar. Inspired by this insight, we propose to cluster shell elements using a three-level hierarchy based on edge lengths $(l_1, ..., l_L)$, diagonal lengths $(d_1, ..., d_{L-3})$, and augmented angles $(\theta_1, ..., \theta_L)$, respectively; see Figure 8 (right).

For the first hierarchy, instead of directly clustering shell elements based on edge lengths $(l_1, ..., l_L)$, we propose to first cluster all the edges in the base mesh $\mathbf{T}$ based on their lengths and then to cluster shell elements according to the clustered edges. By this, we can perform clustering in a space of dimension 1 instead of dimension L. One necessary condition for this approach to work well is that all the edges in the base mesh $\mathbf{T}$ are well clustered, which is indicated by a small value of the function

$$E_{\text{edge}} = \sum_k \sum_i \|e_k^i - \bar{e}_k\|^2, \quad 1 \le k \le K_E \tag{5}$$

where e_k^i is the length of i-th edge in the k-th cluster, $\bar{e}_k$ is the length of the centroid edge of the k-th cluster, and K_E is the number of clusters of edges. To satisfy this condition, we penalize E_{edge} when optimizing the base mesh $\mathbf{T}$; see Equation 8. Moreover, we try different K_E's, starting from a small value such as 1, to find a K_E that leads to a sufficiently small E_{edge}.

For the third hierarchy, the challenge of clustering shell elements based on the ordered list of augmented angles $(\theta_1, ..., \theta_L)$ is that these angles are not independent but subject to Equations 1 and 2. Concerning Equation 1, a necessary condition to achieve a good clustering performance is that there is a small number of clusters of dihedral angles $\{\alpha_{ij}\}$ in the base mesh $\mathbf{T}$ and the dihedral angles in each cluster are very close to one another. This condition is indicated by a small value of the function

$$E_{\text{dihed}} = \sum_k \sum_i \|\alpha_k^i - \bar{\alpha}_k\|^2, \quad 1 \le k \le K_D, \tag{6}$$

where α_k^i is the i-th dihedral angle in the k-th cluster, $\bar{\alpha}_k$ is the centroid dihedral angle of the k-th cluster, and K_D is the number of clusters of dihedral angles. To satisfy this condition, we penalize E_{dihed} when optimizing the base mesh $\mathbf{T}$; see Equation 8. Similarly, we also try different K_D's to find a K_D that leads to a sufficiently small E_{dihed}. Concerning Equation 2, we limit the possible values of augmented angles to a small set of discrete values $\{\bar{\alpha}_i - \bar{\alpha}_k/2\}$ where $1 \leq i,\ k \leq K_D$, thanks to the clustering of dihedral angles $\{\alpha_{ij}\}$. By this, we significantly reduce the space of augmented angles, enabling a discrete optimization on these angles in Section 5.2.

Our hierarchical approach to clustering shell elements is performed as follows:

(1) *Clustering edge lengths* $(l_1, ..., l_L)$. Since all the edges in the base mesh $\mathbf{T}$ have been partitioned into K_E clusters, we assign a cluster ID $\in \{1, ..., K_E\}$ for each edge of a polygon F_i, and then cluster all the polygons based on the assigned edge cluster IDs; see Figure 9. Two polygons are put in the same cluster if they have exactly the same edge cluster IDs up to the L different ways to correspond vertices from the two polygons. Ambiguity occurs when we try to correspond the vertices of two symmetric polygons; see cluster #3 in Figure 9. In this case, we resolve the ambiguity by finding the right correspondence between the vertices via minimizing a similarity metric [Fu et al. 2010; Singh and Schaefer 2010] of the two polygons F_i and $\tilde{F}_i$.

(2) *Clustering diagonal lengths* $(d_1, ..., d_{L-3})$. For polygons in each cluster of the first hierarchy, we compute their similarity based on the lengths of $L-3$ diagonals, where the diagonals have to be shot from a pair of corresponding vertices in the two polygons. Our similarity metric of diagonal lengths is defined as

$$s(F_i, \tilde{F}_i) \;=\; \|\mathbf{d} - \tilde{\mathbf{d}}\|^2, \tag{7}$$

where $\mathbf{d}$ and $\tilde{\mathbf{d}}$ are vectors concatenating $L-3$ diagonal lengths of polygons F_i and $\tilde{F}_i$, respectively. We partition each cluster in the first hierarchy into multiple clusters in the second hierarchy using k-means and the similarity metric. Note that we cannot cluster diagonal lengths in the same way as clustering edge lengths in the first hierarchy since we rely on the correspondence of edges between two polygons to choose diagonals for the clustering. Nodes in the second hierarchy in Figure 8 (right) represent clusters of polygons in the base mesh $\mathbf{T}$. We denote the number of clusters of polygons as K_F.

(3) *Clustering augmented angles* $(\theta_1, ..., \theta_L)$. Since we have discretized the possible values of augmented angles, we put shell elements in the same cluster only when they have exactly the same ordered list of augmented angles $(\theta_1, ..., \theta_L)$.

Thanks to the hierarchical strategy, our clustering approach is scalable and can deal with thousands of elements in a few seconds; see Figure 16. Moreover, our clustering approach is able to cluster elements with various shapes such as triangles, quads, n-gons, and a mix of them; see Figure 12.

Modeling template shell elements. We model a template shell element for each cluster of elements, resulting in K_S templates with distinct shapes. We compute the template as the centroid of each cluster in the explicit geometry space instead of the parameter space since alignment of the shell elements has to be performed in the former space. In detail, we align all the elements in a cluster using

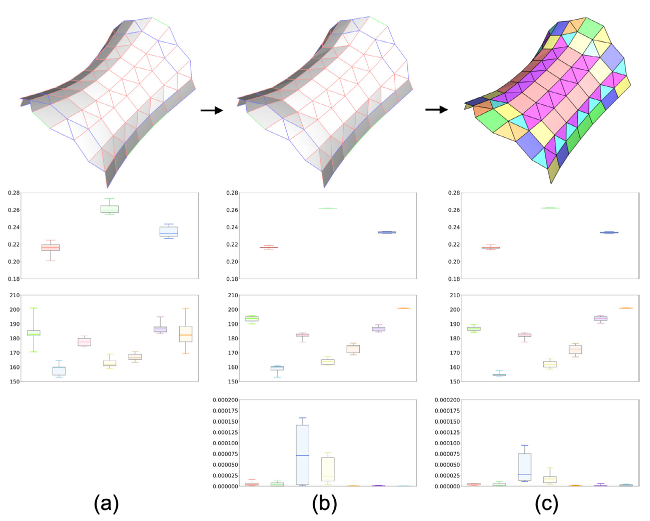

Fig. 10. *Optimizing a base mesh* $\mathbf{T}$ *for* $K_E = 3$ *and* $K_D = 7$*, resulting in* $K_F = 7$ *clusters of polygons. Box plots of clustered edge lengths, dihedral angles, and base polygons (represented as similarity metrics to the cluster centroid) are provided at the bottom. (a) Input base mesh. (b) Reduce intra-cluster variances for edge lengths and dihedral angles by minimizing* E_1*. (c) Reduce intra-cluster variances for base polygons by minimizing* E_2*.*

rigid transformations computed with Equation 4 and average corresponding vertices of all the transformed elements to obtain the template's geometry. In case the template contains some non-planar faces, we fit a plane for each of these non-planar faces and intersect the fitted planes as well as the planes of the remaining planar faces to obtain the final geometry of the template.

5 OPTIMIZING SHELL ELEMENTS

A shell structure $\mathbf{M}'$ with discrete equivalence classes usually requires a large number (K_S) of template shell elements to ensure that the structure is seamless and buildable by satisfying the conditions in Equation 3. In this section, our goal is to find an as-small-as-possible number (K_S) of template shell elements such that the resulting shell structure $\mathbf{M}'$ still satisfies the conditions in Equation 3, aiming to maximize reusability of the shell elements.

We achieve the goal by optimizing the geometry of shell elements in the structure $\mathbf{M}'$. Directly optimizing the geometry of the shell elements is challenging due to the large search space. Thanks to our hierarchical approach to clustering shell elements in Section 4, we are able to optimize the shell elements hierarchically. Our idea is to minimize the number (K_S) of clusters of shell elements by minimizing the nodes in each level of the hierarchy tree; see again Figure 8 (right). In our optimization, we first optimize the base mesh $\mathbf{T}$ to minimize the number (K_E) of clusters of edges as well as the number (K_D) of clusters of dihedral angles. Then, we optimize the base mesh $\mathbf{T}$ to minimize the number (K_F) of clusters of base polygons. Lastly, we optimize the augmented angles $\{\theta_{ij}\}$ to minimize the number (K_S) of clusters of shell elements. We introduce the first two steps of our optimization in Section 5.1 and the final step in Section 5.2.

5.1 Optimizing Base Mesh

In Section 4, we assume that edges in the base mesh **T** are well clustered for clustering base polygons. We also assume dihedral angles in the base mesh **T** are well clustered in order to cluster shell elements based on the augmented angles. To this end, we partition the edges and dihedral angles in the mesh **T** into K_E and K_D clusters respectively, and then optimize vertices of the mesh **T** to improve the clustering performance. We start running this optimization with small values of K_E and K_D, gradually increase them, and repeat the optimization until it converges. The objective function of our optimization is a weighted sum of different target terms:

$$E_1 = \lambda_1 E_{\text{edge}} + \lambda_2 E_{\text{dihed}} + \lambda_3 E_{\text{planar}} + \lambda_4 E_{\text{surf}} + \lambda_5 E_{\text{smth}}. \quad (8)$$

Here the terms E_{edge} and E_{dihed} are defined in Equations 5 and 6 and measure the clustering performance of the edges and dihedral angles, respectively. The other terms [Bouaziz et al. 2012] are defined as

$$\begin{aligned} E_{\text{planar}} &= \sum_i P(F_i), \\ E_{\text{surf}} &= \sum_i \|\mathbf{v}_i - \mathbf{c}(\mathbf{v}_i)\|^2, \\ E_{\text{smth}} &= \sum_i \| \sum_{\{i,j\}\in \mathbf{E}} \omega_{ij}(\mathbf{v}_j - \mathbf{v}_i)\|^2, \end{aligned}$$

where $P(F_i)$ measures the planarity of face F_i as the sum of squared distance from the vertices of F_i to their best fitting plane, $\{\mathbf{v}_i\}$ denotes the vertices of the mesh **T**, $\mathbf{c}(\mathbf{v}_i)$ is the closest point on the original base mesh to the vertex $\mathbf{v}_i$, and **E** denotes the set of mesh edges. We empirically set the weights as $\lambda_1 = 1$, $\lambda_2 = 2$, $\lambda_3 = 1$, $\lambda_4 = 1$, and $\lambda_5 = 1$ in our experiments to balance the impact of different target terms.

After the optimization, we cluster the polygons in the mesh **T** based on the lengths of their edges and diagonals using the approach in Section 4. For each of the K_F clusters of polygons, we generate a template polygon using an approach similar to generating a template element. To improve the performance of clustering the polygons, we run another optimization on the mesh **T** with the objective function:

$$E_2 = E_1 + \lambda_6 E_{\text{polygon}} \quad (9)$$

with

$$E_{\text{polygon}} = \sum_k \sum_i s(F_k^i, \bar{F}_k), \quad 1 \le k \le K_F,$$

where $s(\cdot)$ is a function defined in Equation 7, F_k^i is the i-th polygon in the k-th cluster, and $\bar{F}_k$ is the template polygon of the k-th cluster. We set $\lambda_6 = 1$ in our experiments.

We solve the two optimizations using the ShapeOp library [Bouaziz et al. 2012, 2014]. In the supplementary material, we explain how the target terms, E_{edge}, E_{dihed}, E_{planar}, and E_{polygon}, are represented using vertices $\{\mathbf{v}_i\}$ of the base mesh **T**. Figure 10 shows an example of running our two-stage optimization on the base mesh with fixed K_E and K_D, showing that the performance of clustering edges, dihedral angles, and polygons is significantly improved by our optimization on the base mesh.

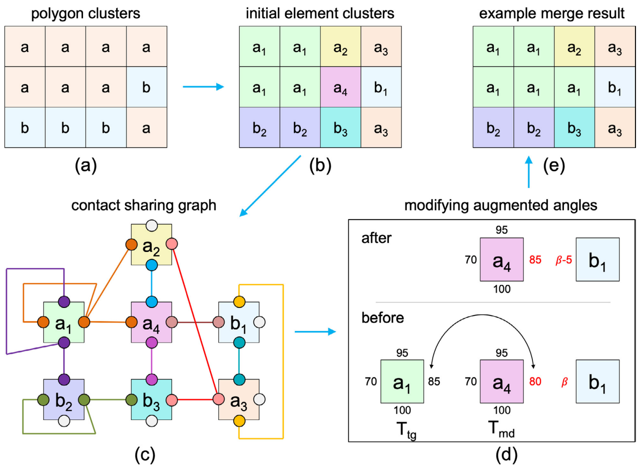

Fig. 11. *Optimizing augmented angles. (a) Clustering of polygons in the optimized mesh* **T**, *i.e., two templates a and b. (b) Partition each polygon cluster into multiple element clusters by initializing the augmented angles. (c) Build a contact sharing graph to guide the modification of augmented angles to reduce K_S. To simplify understanding of the graph, we assume instances of each template in (b) have the same orientation in the structure. (d&e) An example valid cluster merge operation, where we modify the augmented angle of a_4's right face from 80° to 85° such that a_4's geometry is exactly the same as that of a_1. We also modify the left face of b_1 to satisfy Equation 1.*

5.2 Optimizing Augmented Angles

In this step, our task is to maximize reusability of the shell elements in the structure $\mathbf{M}'$ by optimizing the augmented angles $\{\theta_{ij}\}$ while fixing the base mesh **T**. The search space of each augmented angle is the set of discrete values $\{\bar{\alpha}_i - \bar{\alpha}_k/2\}$ where $1 \le i,\ k \le K_D$; see again Section 4. We initialize each augmented angle $\theta_{ij} = \bar{\alpha}_k/2$ by choosing the partitioning plane as the plane bisecting the dihedral angle $\bar{\alpha}_k$, in order to satisfy Equations 1 and 2. Note that here we assume $\alpha_{ij} \approx \bar{\alpha}_k$ since all the dihedral angles $\{\alpha_{ij}\}$ have been well clustered. For the side face of an element that does not contact any other element in the structure, the initial partitioning plane is chosen as the plane perpendicular to the element's base polygon. Next, we cluster these initial shell elements using the approach in Section 4 and perform element replacement. Although the initial result is likely to satisfy all the conditions in Equation 3, it may form too many clusters which lead to poor reusability of the elements; see Figure 11(a&b). Hence, we attempt to merge the clusters of elements to reduce the number (K_S) of template elements.

Our cluster merge operation is performed as follows. We choose a cluster C_{md} to modify and a target cluster C_{tg}, whose template element is denoted as T_{md} and T_{tg}, respectively. The two clusters, C_{md} and C_{tg}, have to share the same parent node (i.e., base polygons from the same cluster) in the hierarchy tree in Figure 8. We modify the geometry (i.e., some augmented angles) of the template T_{md} such that the modified geometry is exactly the same as that of T_{tg}. By this, we can merge the two clusters C_{md} and C_{tg}, reducing the number (K_S) of template elements by one; see Figure 11(d&e).

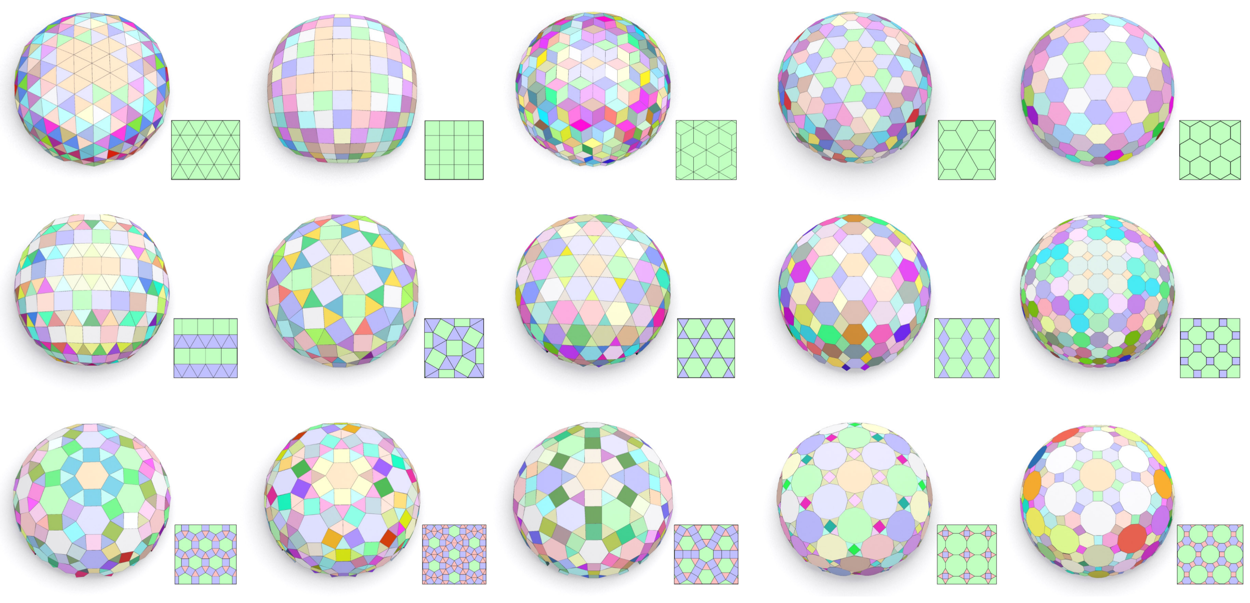

Fig. 12. *Our approach allows generating results with a variety of patterns of a* Dome *surface, where the corresponding 2D tessellation is shown in the boxed image. The 2D tessellations in the top, middle, and bottom rows are monohedral, dihedral, trihedral tilings, respectively. For each result, shell elements falling in the same discrete equivalence class are rendered with the same color.*

This modification operation is valid only when it satisfies three requirements. First, each modified augmented angle is still within the range defined by Equation 2. Second, modifying a template element T_{md} decreases the total number of template elements. This is because modifying augmented angles of a template element T_{md} will immediately change the geometry of the template's instances. To satisfy the dihedral angle constraint in Equation 1, we also have to modify the geometry of the instances' neighboring elements in the structure $\mathbf{M}'$, making it possible to introduce new clusters of elements. Third, we require that the shell structure $\mathbf{M}'$ with modified augmented angles should still be seamless and buildable by satisfying the conditions in Equation 3. We undo the cluster merge operation if the operation is invalid.

Contact sharing graph. The major challenge of choosing a cluster merge operation is to satisfy the second requirement above since modification on a template element T_{md} is likely to propagate in the structure $\mathbf{M}'$ due to reuse of template elements and contacts between instance elements in the structure. To address the challenge, our insight is that modification on a template element T_{md} with less propagation will have a higher chance to result in a valid cluster merge operation. To this end, we propose a *contact sharing graph* to guide the cluster merge operation. In this graph, each node represents a side face of a template element, and each edge represents a contact sharing by instances of one or two template elements in the structure $\mathbf{M}'$; see Figure 11(b&c). The contact sharing graph is usually disconnected, and is composed of a number of connected subgraphs. For example, there are 14 connected subgraphs in Figure 11(c), among which 5 subgraphs consist of a single node (in grey color) and the other 9 subgraphs are visualized in different colors. The cost of modifying a side face (i.e., the augmented angle) of a template is measured using $h - 1$, where h is the number of nodes of the subgraph that the side face is located at. For example, the cost of modifying the right face of templates a_1, a_2, a_3, and a_4 in Figure 11(c) are 3, 2, 0, 1, respectively. The cost of choosing a template (i.e., a cluster) to modify is measured by summing the cost of modifying each of its side faces.

Cluster merge algorithm. To perform a cluster merge operation, we have to choose two clusters C_{md} and C_{tg}. We choose the cluster C_{md} guided by the contact sharing graph. In detail, we choose m ($m = 10$ in our experiments) candidates of cluster C_{md} with the lowest cost computed above. For each candidate of cluster C_{md}, we choose n ($n = 10$ in our experiments) candidates of the target cluster C_{tg} in a way that a small modification on the template T_{md} is necessary. To this end, we measure the difference between templates T_{md} and T_{tg} upon L different ways to correspond the base polygons' edges, where L is the number of edges in the base polygon. The difference between templates T_{md} and T_{tg} is measured by two quantities: 1) the number of different augmented angles for corresponding edges and 2) the difference in augmented angles for corresponding edges. We give the first quantity a higher priority.

For each of the m candidates of cluster C_{md}, we perform the cluster merge operation with each of the n candidates of cluster C_{tg} and check its validity. This process terminates once a cluster merge operation succeeds; see Figure 11(d&e) for an example. Our cluster merge algorithm is performed recursively. At each recursion, we update the contact sharing graph, select candidates of clusters C_{md} and C_{tg}, and perform the cluster merge operation until success. We run our algorithm until the number of clusters cannot be reduced. Please refer to the accompanying video for a running example of the cluster merge algorithm.

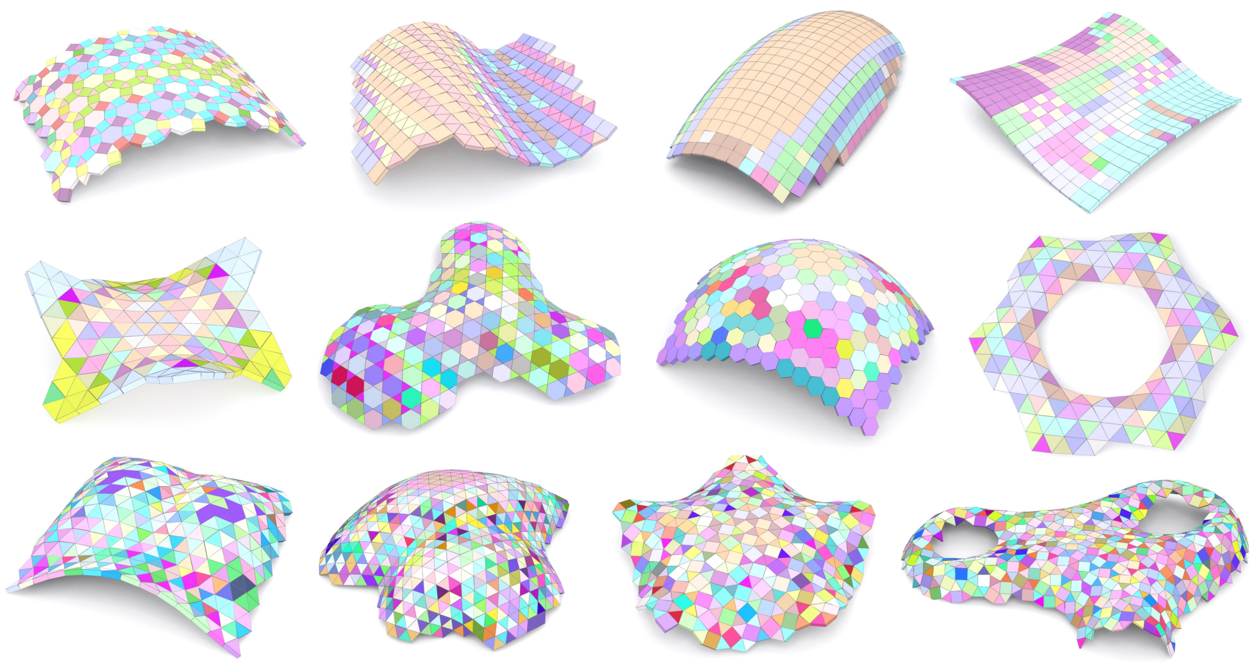

Fig. 13. *Our approach allows generating results from a variety of freeform guiding surfaces. From left to right and then top to bottom:* Arcuation, Curved Surface, Canopy, Monkey Saddle, Roof, Blob, Arch, Snow, Vouga Surface, Botanic Bubble, Pentagon, *and* Aquadom.

6 RESULTS

Implementation. We implemented our approach in C++ and libigl [Jacobson et al. 2018] on a desktop computer with a 3.7GHz CPU and 16GB memory. Our approach hierarchically clusters and optimizes shell elements to reduce the three error metrics in Section 3 for a fixed K_E (# cluster of edges) and K_D (# cluster of dihedral angles). If the optimization does not achieve the specified tolerance in Equation 3, we simply increase the K_E and/or K_D and continue the optimization. Our optimization terminates when we find K_S template elements that satisfy conditions in Equation 3.

Our approach allows modeling shell structures with discrete equivalence classes for a variety of patterns. Figure 12 shows 15 different patterns on a Dome surface, which contains triangles, quads, pentagons, hexagons, octagons, dodecagons, and a mix of them. In general, reusable shell elements are distributed symmetrically over the symmetric surface. We tested our modeling approach on a wide range of surfaces in Figure 13, e.g., Curved Surface with zero Gaussian curvature, Vouga Surface with both positive and negative Gaussian curvature, and Aquadom with non-trivial topology. Our approach automatically puts elements of the same class at locations on each surface with the same or similar curvature. Please refer to the supplementary material for the input surface and optimized base mesh of each result shown in Figure 13.

Table 1 summarizes statistics of the results presented in the paper, where each input surface model has been normalized such that the longest edge of the model's bounding box has length of 2. The optimized base mesh usually has a few clusters of edges (K_E) and dihedral angles (K_D). In our results, we find that the number of template polygons K_F is much smaller than the number of template shell elements K_S, which confirms the difficulty of reusing shell elements caused by their side faces. All our modeled shell structures appear to be seamless due to a small tolerance on the error metrics (C_{avg}, C_{max}, G_{avg}, G_{max}, O_{avg}, and O_{max}). Our hierarchical cluster-and-optimize approach is efficient. The base mesh optimization usually takes less than one second while the optimization on augmented angles can take a few minutes to an hour. For each result, we provide 3D models of the input surface, optimized base mesh, computed template elements, and modeled shell structure in the supplementary data.

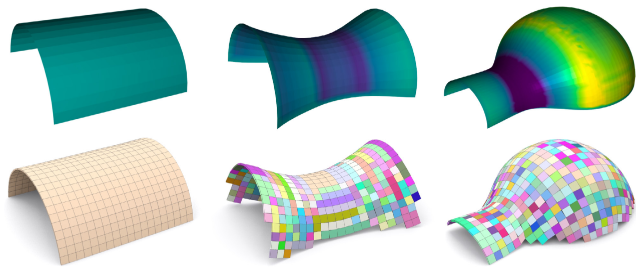

Fig. 14. *Reusability of shell elements becomes worse when the curvature of a guiding surface becomes more complex, e.g., from (left) a single-curved surface, (middle) a double-curved surface, to (right) a freeform surface.*

In our experiments, we find that reusability of shell elements (measured with N/K_S) highly depends on the guiding surface's

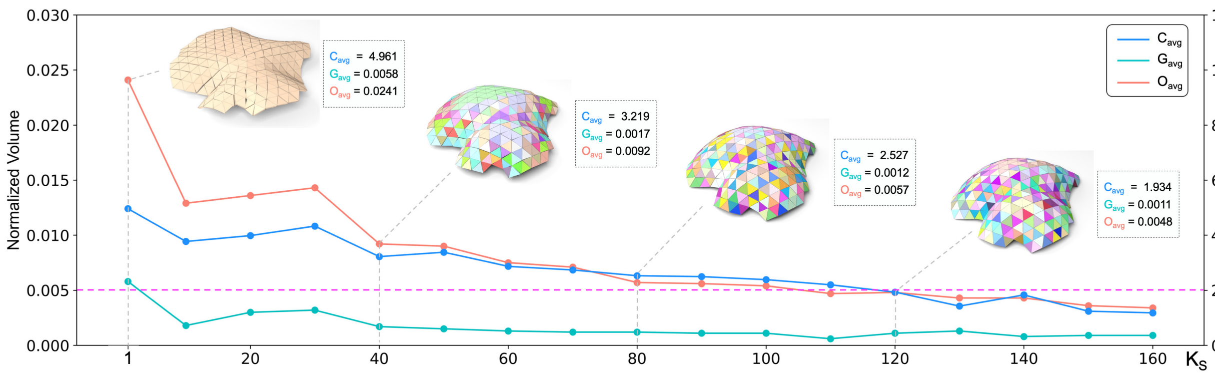

Fig. 15. *Average errors of the three metrics decrease when a larger number of template elements are allowed to model a shell structure with $N = 328$ elements. Few template elements lead to obvious gaps and overlaps in the structure; see the left two sampled results. The horizontal purple line indicates the thresholds that we set on the error metrics.*

Table 1. *Statistics and timings. We report the input surface, shell thickness (τ), number of shell elements (N), number of clusters of edges (K_E), dihedral angles (K_D), base polygons (K_F), and shell elements (K_S), reusability (N/K_S), average and maximum errors to evaluate the resulting shell structure $\mathbf{M}'$, as well as timings to optimize the base mesh and augmented angles respectively.*

Fig	Surface	τ	N	K_E	K_D	K_F	K_S	N/K_S	C_{avg}	C_{max}	G_{avg}	G_{max}	O_{avg}	O_{max}	Optim. Mesh (min)	Optim. Angle (min)
1	Flower	0.05	181	8	16	32	60	3.0	0.101	1.720	0.0001	0.0009	0.0001	0.0016	0.003	0.003
	Arcuation	0.03	353	1	3	3	27	13.1	1.095	6.304	0.0002	0.0106	0.0014	0.0092	0.009	0.136
	Curved Surface	0.03	383	1	3	2	12	31.9	1.037	8.575	0.0003	0.0139	0.0018	0.0110	0.006	0.004
	Canopy	0.02	311	2	3	3	15	20.7	1.718	7.645	0.0008	0.0302	0.0031	0.0199	0.008	0.006
	Monkey Saddle	0.03	323	1	3	1	18	17.9	0.861	3.129	0.0001	0.0110	0.0014	0.0070	0.002	0.005
	Roof	0.04	198	2	6	4	26	7.6	1.531	5.875	0.0004	0.0188	0.0033	0.0185	0.001	0.033
	Blob	0.03	385	2	4	14	114	3.4	1.256	5.906	0.0003	0.0147	0.0016	0.0146	0.009	0.611
13	Arch	0.04	259	2	3	10	60	4.3	1.120	6.780	0.0015	0.0215	0.0023	0.0347	0.012	0.246
	Snow	0.04	204	4	8	11	18	11.3	0.740	2.333	0.0004	0.0108	0.0020	0.0122	0.009	0.097
	Vouga Surface	0.04	444	2	12	4	141	3.1	1.412	8.574	0.0003	0.0206	0.0036	0.0280	0.004	7.680
	Botanic Bubble	0.02	949	3	9	11	276	3.4	1.414	9.939	0.0006	0.0236	0.0035	0.0298	0.018	56.427
	Pentagon	0.03	456	2	9	9	181	2.5	0.956	5.924	0.0001	0.0133	0.0016	0.0123	0.008	5.688
	Aquadom	0.01	822	3	12	28	500	1.6	1.451	9.642	0.0004	0.0227	0.0019	0.0263	0.023	22.458
	Cylinder	0.02	475	1	2	1	1	475.0	0.110	0.677	0.0001	0.0025	0.0003	0.0020	0.015	0.001
14	Hyperbolic	0.02	425	4	6	11	114	3.7	1.465	6.420	0.0006	0.0363	0.0032	0.0391	0.013	2.694
	Igloo	0.02	458	8	20	81	278	1.6	1.177	7.328	0.0004	0.0262	0.0028	0.0298	0.015	1.066
	Blob	0.02	251	1	6	1	105	2.4	1.336	5.269	0.0016	0.0283	0.0025	0.0247	0.012	13.145
15	Blob	0.01	750	1	6	1	286	2.6	0.753	4.236	0.0011	0.0181	0.0016	0.0244	0.027	25.486
	Blob	0.008	2286	1	6	1	357	6.4	0.698	4.819	0.0015	0.0228	0.0021	0.0262	0.263	60.329
	Blob	0.003	5631	1	6	1	422	13.3	0.540	3.971	0.0019	0.0245	0.0023	0.0231	5.080	94.663
	Dome	0.18	45	4	6	9	12	3.8	0.050	0.225	0.0001	0.0016	0.0002	0.0024	0.001	0.001
17	Arch	0.08	81	2	6	12	16	5.1	0.962	4.498	0.0066	0.0172	0.0000	0.0010	0.003	0.001
	Hyperbolic	0.08	103	3	7	7	27	3.8	1.620	9.441	0.0019	0.0342	0.0036	0.0495	0.001	0.006
	Roof	0.05	133	4	10	15	37	3.6	0.670	8.245	0.0001	0.0055	0.0008	0.0063	0.005	0.007

shape. A guiding surface with simple curvature usually leads to high reusability of shell elements, such as developable surfaces Arcuation and Curved Surface in Figure 13. A guiding surface with symmetric shape also improves reusability of shell elements, such as Canopy, Roof, and Snow in Figure 13. To validate this observation, we conduct an experiment to model shell structures for single-curved, double-curved, and freeform surfaces, respectively,

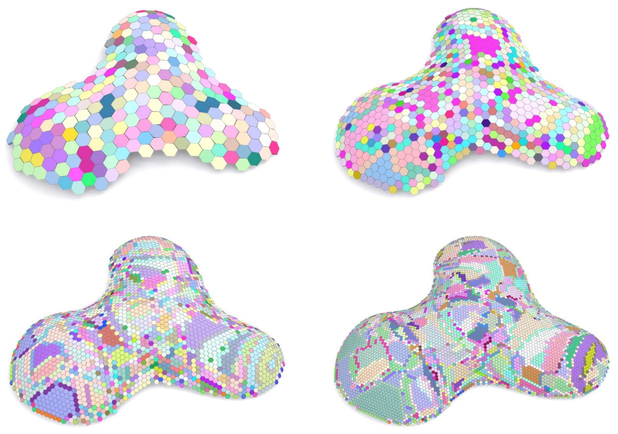

Fig. 16. *Reusability of shell elements improves when there are more shell elements used for approximating an input smooth surface; see also statistics in Table 1.*

using the same pattern; see Figure 14. A single template shell element can be used to tile the whole shape of the single-curved surface, which is a half cylinder. However, reusing shell elements becomes harder when the surface curvature gets more complex. We conduct one more experiment to study the relation between reusability of shell elements and the number of elements used to approximate a guiding surface; see Figure 16. We find that reusability of shell elements increases when we approximate a smooth surface with a larger number of elements. One possible reason to explain this is that a guiding surface can be approximated better by using more planar elements (i.e., E_{surf} in Equation 8) and thus there is a higher tolerance on the clustering errors.

We also study the relation between reusability of shell elements and tolerance on the errors in seamlessness and buildability of shell structures. In detail, we change our approach by fixing the number (K_S) of template elements and running our optimization to minimize the errors for a structure with $N = 328$ elements. We run our optimization for different K_S's and plot average errors

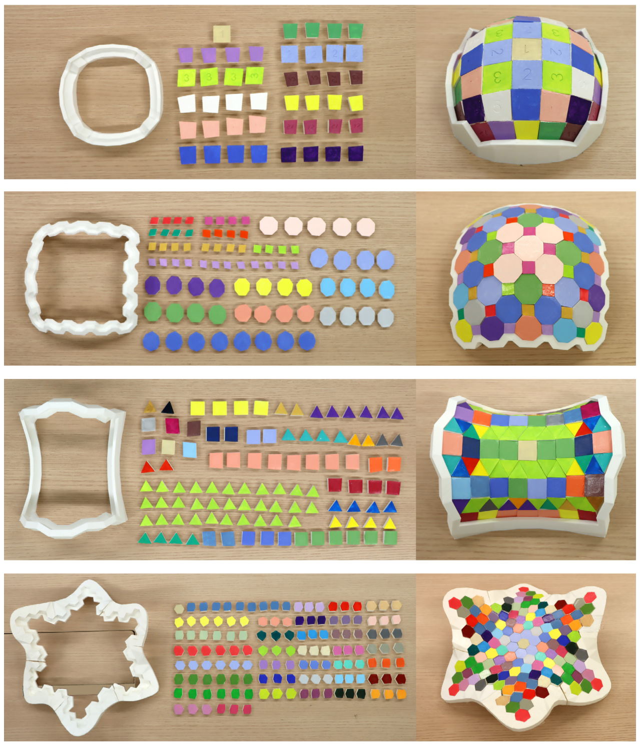

Fig. 17. *3D printed shell elements and assembled shell structures. Shell elements in the same class are painted with the same color.*

of the three metrics; see Figure 15. When only a single template is allowed, there exist many gaps and overlaps in the structure and the structure cannot closely approximate the input surface. When more templates are allowed, all the three errors decrease and the structure can approximate the surface better. Undesirable contacts, gaps, and overlaps become unnoticeable when more than $K_S = 120$ templates are used. Note that undesirable contacts, gaps, and overlaps can be completely removed (i.e., zero errors) by further increasing $K_S = N = 328$. In this case, our shell structure with discrete equivalence classes is degenerated into conventional ones with no reuse of the elements.

Fabricated prototypes. To validate buildability, we fabricated four shell structures modeled by our approach using Ultimaker S3 printer with tough PLA material, including Dome, Arch, Hyperbolic, and Roof; see Figure 17. In detail, we 3D print the shell elements, paint elements from the same class using the same color, and assemble the elements together with a large boundary part to form the final structure. All the four structures are buildable and the appearance of each physical prototype is consistent with its virtual counterpart, showing that our specified tolerance on the gap and overlap errors are acceptable in practice. Interestingly, we find that three (i.e., Dome, Arch, Hyperbolic) of the four structures can be self-supporting without using any material like glue to bind the elements together. In addition, the Dome structure is not only self-supporting but also able to bear external load up to 160g without structural collapse. Self-supporting of the three shell structures demonstrates that the small overlap, gap, and contact errors introduced in our modeling process are possible to be acceptable in practice such that the contacts among the 3D printed elements are still able to transmit internal forces within each of the three structures to balance the external force (e.g., gravity). Please watch the accompanying video for demos.

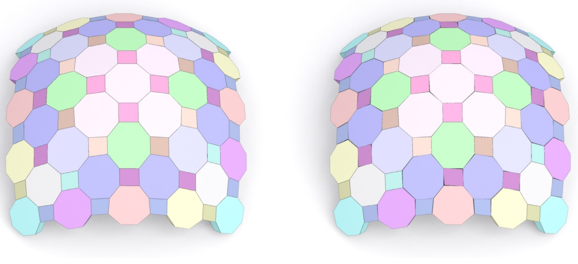

Fig. 18. *Optimized shell structures (left) before and (right) after replacing each element with a template. Note that the two structures have very similar geometry.*

Discussion. Existing static analysis methods such as the equilibrium method [Kao et al. 2022; Whiting et al. 2009; Yao et al. 2017] cannot be directly used to analyze *structural stability* of our modeled shell structures due to the contact, gap, and overlap errors even though these errors are small in our structures. Currently, we employ the equilibrium method to analyze structural stability of the shell structure before the template replacement instead. We assume the two structures (before and after the template replacement) have similar structural stability since they have very similar geometry; see Figure 18. Similarly, we analyze *buildability* of the shell structure before the template replacement by finding a collision-free disassembly plan. The shell structure is considered as buildable if we can find such a disassembly plan. We also assume the two structures (before and after the template replacement) have similar buildability. We consider developing computational methods to directly analyze structural stability and buildability of our modeled shell structures as an important yet challenging future work.

7 CONCLUSION

This paper studies a challenging problem of modeling masonry shell structures with discrete equivalence classes that closely approximate a given freeform surface. We define three error metrics that evaluate seamlessness and buildability of the shell structures and propose an approach to find a small number of template elements for modeling a shell structure whose error metrics are within user-specified tolerances. Our modeling approach is based on hierarchically clustering and optimizing geometric primitives, from edges, dihedral angles, base polygons, to polyhedrons. Our approach allows modeling shell structures with discrete equivalence classes that can approximate a wide variety of freeform surfaces using template elements with different shapes. We demonstrate buildability of our shell structures by 3D printing four prototypes, three of which can be self-supporting.

Limitations and future work. Our work has several limitations that open up interesting directions for future research. First, we cannot guarantee that our modeled shell structures are self-supporting due to the challenge of analyzing their structural stability. Second, we assume that the shape of each shell element is a convex polyhedron that contacts one another with planar faces. One interesting future work is to generalize our modeling approach to support shell elements with concave shapes and curved contacts, which has potential to improve stability of the whole structure [Wang et al. 2021a]. Third, our modeling approach optimizes the geometry of the base mesh as well as the augmented angles while keeping the surface tessellation fixed. Extending the approach to also optimize the tessellation as well as the mapping of the tessellation onto the input surface is an interesting research challenge. Lastly, we are interested in studying shell structures with discrete equivalence classes where the elements interlock with one another to form a stable structure [Song 2022].

ACKNOWLEDGEMENTS

We thank the reviewers for their valuable comments, and Zebin Chen for helping on preparing some 3D surface models and renderings. This work was supported by the SUTD grant RS-INSUR-00027, the Ministry of Education, Singapore, under its Academic Research Fund Grants (RG20/20 & MOE-T2EP20220-0014), and the Royal Society (IES\R3\193208).

REFERENCES

Sigrid Adriaenssens, Philippe Block, Diederik Veenendaal, and Chris Williams (Eds.). 2014. *Shell Structures for Architecture: Form Finding and Optimization.* Routledge.

Ergun Akleman, Vinayak R. Krishnamurthy, Chia-An Fu, Sai Ganesh Subramanian, Matthew Ebert, Matthew Eng, Courtney Starrett, and Haard Panchal. 2020. Generalized Abeille Tiles: Topologically Interlocked Space-filling Shapes Generated Based on Fabric Symmetries. *Comp. & Graph. (SMI)* 89 (2020), 156–166.

K. S. Arun, T. S. Huang, and S. D. Blostein. 1987. Least-Squares Fitting of Two 3-D Point Sets. *IEEE Trans. Pat. Ana. & Mach. Int.* PAMI-9, 5 (1987), 698–700.

Guy Austern, Isaac Guedi Capeluto, and Yasha Jacob Grobman. 2018. Rationalization Methods in Computer Aided Fabrication: A Critical Review. *Automation in Construction* 90 (2018), 281–293.

Sofien Bouaziz, Mario Deuss, Yuliy Schwartzburg, Thibaut Weise, and Mark Pauly. 2012. Shape-Up: Shaping Discrete Geometry with Projections. *Comp. Graph. Forum (SGP)* 31, 5 (2012), 1657–1667.

Sofien Bouaziz, Sebastian Martin, Tiantian Liu, Ladislav Kavan, and Mark Pauly. 2014. Projective Dynamics: Fusing Constraint Projections for Fast Simulation. *ACM Trans. on Graph. (SIGGRAPH)* 33, 4 (2014), 154:1–154:11.

Jan Brütting, Gennaro Senatore, and Corentin Fivet. 2021. Design and Fabrication of A Reusable Kit of Parts for Diverse Structures. *Automation in Construction* 125 (2021), 103614:1–103614:15.

Xuelin Chen, Honghua Li, Chi-Wing Fu, Hao Zhang, Daniel Cohen-Or, and Baoquan Chen. 2018. 3D Fabrication with Universal Building Blocks and Pyramidal Shells. *ACM Trans. on Graph. (SIGGRAPH Asia)* 37, 6 (2018), 189:1–189:15.

Fernando de Goes, Pierre Alliez, Houman Owhadi, and Mathieu Desbrun. 2013. On the Equilibrium of Simplicial Masonry Structures. *ACM Trans. on Graph. (SIGGRAPH)* 32, 4 (2013), 93:1–93:10.

Michael Eigensatz, Martin Kilian, Alexander Schiftner, Niloy J. Mitra, Helmut Pottmann, and Mark Pauly. 2010. Paneling Architectural Freeform Surfaces. *ACM Trans. on Graph. (SIGGRAPH)* 29, 4 (2010), 45:1–45:10.

Chi-Wing Fu, Chi-Fu Lai, Ying He, and Daniel Cohen-Or. 2010. K-set Tilable Surfaces. *ACM Trans. on Graph. (SIGGRAPH)* 29, 4 (2010), 44:1–44:6.

Mathieu Huard, Michael Eigensatz, and Philippe Bompas. 2015. Planar Panelization with Extreme Repetition. In *Proc. Advances in Architectural Geometry 2014*. 259–279.

Alec Jacobson, Daniele Panozzo, et al. 2018. libigl: A simple C++ geometry processing library. https://libigl.github.io/.

Gene Ting-Chun Kao, Antonino Iannuzzo, Bernhard Thomaszewski, Stelian Coros, Tom Van Mele, and Philippe Block. 2022. Coupled Rigid-Block Analysis: Stability-Aware Design of Complex Discrete-Element Assemblies. *Computer-Aided Design* 146 (2022), 103216:1–103216:20.

Vinayak R. Krishnamurthy, Ergun Akleman, Sai Ganesh Subramanian, Katherine Boyd, Chia-An Fu, Matthew Ebert, Courtney Starrett, and Neeraj Yadav. 2020. Bi-Axial Woven Tiles: Interlocking Space-Filling Shapes Based on Symmetries of Bi-Axial Weaving Patterns. In *Proc. Graphics Interface.* 286–298.

Ligang Liu, Lei Zhang, Yin Xu, Craig Gotsman, and Steven J. Gortler. 2008. A Local/Global Approach to Mesh Parameterization. *Comp. Graph. Forum (SGP)* 27, 5 (2008), 1495–1504.

Yang Liu, Hao Pan, John Snyder, Wenping Wang, and Baining Guo. 2013. Computing Self-Supporting Surfaces by Regular Triangulation. *ACM Trans. on Graph. (SIGGRAPH)* 32, 4 (2013), 92:1–92:10.

Zhong-Yuan Liu, Zhan Zhang, Di Zhang, Chunyang Ye, Ligang Liu, and Xiao-Ming Fu. 2021. Modeling and Fabrication with Specified Discrete Equivalence Classes. *ACM Trans. on Graph. (SIGGRAPH)* 40, 4 (2021), 41:1–41:12.

Sheng-Jie Luo, Yonghao Yue, Chun-Kai Huang, Yu-Huan Chung, Sei Imai, Tomoyuki Nishita, and Bing-Yu Chen. 2015. Legolization: Optimizing LEGO Designs. *ACM Trans. on Graph. (SIGGRAPH Asia)* 34, 6 (2015), 222:1–222:12.

Long Ma, Ying He, Qian Sun, Yuanfeng Zhou, Caiming Zhang, and Wenping Wang. 2019. Constructing 3D Self-Supporting Surfaces with Isotropic Stress Using 4D Minimal Hypersurfaces of Revolution. *ACM Trans. on Graph.* 38, 5 (2019), 144:1–144:13.

Masaaki Miki, Takeo Igarashi, and Philippe Block. 2015. Parametric Self-supporting Surfaces via Direct Computation of Airy Stress Functions. *ACM Trans. on Graph. (SIGGRAPH)* 34, 4 (2015), 89:1–89:12.

Daniele Panozzo, Philippe Block, and Olga Sorkine-Hornung. 2013. Designing Unreinforced Masonry Models. *ACM Trans. on Graph. (SIGGRAPH)* 32, 4 (2013), 91:1–91:11.

I-Chao Shen, Ming-Shiuan Chen, Chun-Kai Huang, and Bing-Yu Chen. 2020. ZomeFab: Cost-Effective Hybrid Fabrication with Zometools. *Comp. Graph. Forum* 39, 1 (2020), 322–332.

Shen-Guan Shih. 2016. On the Hierarchical Construction of SL Blocks. In *Advances in Architectural Geometry.* 124–136.

Mayank Singh and Scott Schaefer. 2010. Triangle Surfaces with Discrete Equivalence Classes. *ACM Trans. on Graph. (SIGGRAPH)* 29, 4 (2010), 46:1–46:7.

Peng Song. 2022. Interlocking Assemblies: Applications and Methods. *Materials Today: Proceedings (International Conference on Additive Manufacturing for a Better World)* 70 (2022), 78–82.

Peng Song, Chi-Wing Fu, Prashant Goswami, Jianmin Zheng, Niloy J. Mitra, and Daniel Cohen-Or. 2013. Reciprocal Frame Structures Made Easy. *ACM Trans. on Graph. (SIGGRAPH)* 32, 4 (2013), 94:1–94:13.

Sai Ganesh Subramanian, Mathew Eng, Vinayak R. Krishnamurthy, and Ergun Akleman. 2019. Delaunay Lofts: A Biologically Inspired Approach for Modeling Space Filling Modular Structures. *Comp. & Graph. (SMI)* 82 (2019), 73–83.

Chengcheng Tang, Xiang Sun, Alexandra Gomes, Johannes Wallner, and Helmut Pottmann. 2014. Form-finding with Polyhedral Meshes Made Simple. *ACM Trans. on Graph. (SIGGRAPH)* 33, 4 (2014), 70:1–70:9.

Romain Testuz, Yuliy Schwartzburg, and Mark Pauly. 2013. Automatic Generation of Constructable Brick Sculptures. In *Proc. Eurographics.* short paper.

Etienne Vouga, Mathias Höbinger, Johannes Wallner, and Helmut Pottmann. 2012. Design of Self-supporting Surfaces. *ACM Trans. on Graph. (SIGGRAPH)* 31, 4 (2012), 87:1–87:11.

Ziqi Wang, Peng Song, Florin Isvoranu, and Mark Pauly. 2019. Design and Structural Optimization of Topological Interlocking Assemblies. *ACM Trans. on Graph. (SIGGRAPH Asia)* 38, 6 (2019), 193:1–193:13.

Ziqi Wang, Peng Song, and Mark Pauly. 2021a. MOCCA: Modeling and Optimizing Cone-joints for Complex Assemblies. *ACM Trans. on Graph. (SIGGRAPH)* 40, 4 (2021), 181:1–181:14.

Ziqi Wang, Peng Song, and Mark Pauly. 2021b. State of the Art on Computational Design of Assemblies with Rigid Parts. *Comp. Graph. Forum (Eurographics)* 40, 2 (2021), 633–657.

Emily Whiting, John Ochsendorf, and Frédo Durand. 2009. Procedural Modeling of Structurally-Sound Masonry Buildings. *ACM Trans. on Graph. (SIGGRAPH Asia)* 28, 5 (2009), 112:1–112:9.

Emily Whiting, Hijung Shin, Robert Wang, John Ochsendorf, and Frédo Durand. 2012. Structural Optimization of 3D Masonry Buildings. *ACM Trans. on Graph. (SIGGRAPH Asia)* 31, 6 (2012), 159:1–159:11.

Jiaxian Yao, Danny M. Kaufman, Yotam Gingold, and Maneesh Agrawala. 2017. Interactive Design and Stability Analysis of Decorative Joinery for Furniture. *ACM Trans. on Graph.* 36, 2 (2017). Article No. 20.

Yinan Zhang and Devin Balkcom. 2016. Interlocking Structure Assembly with Voxels. In *Proc. IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems.* 2173–2180.

Henrik Zimmer, Marcel Campen, David Bommes, and Leif Kobbelt. 2012. Rationalization of Triangle-Based Point-Folding Structures. *Comp. Graph. Forum (Eurographics)* 31, 2 (2012), 611–620.

Henrik Zimmer and Leif Kobbelt. 2014. Zometool Rationalization of Freeform Surfaces. *IEEE Trans. Vis. & Comp. Graphics* 20, 10 (2014), 1461–1473.

Henrik Zimmer, Florent Lafarge, Pierre Alliez, and Leif Kobbelt. 2014. Zometool Shape Approximation. *Graphical Models* 76, 5 (2014), 390–401.