DOI: 10.1111/cgf.13267

Pacific Graphics 2017
J. Barbic, W.-C. Lin, and O. Sorkine-Hornung
(Guest Editors)

Volume 36 (2017), Number 7

Modeling, Evaluation and Optimization of Interlocking Shell Pieces

Miaojun Yao[1], Zhili Chen[2], Weiwei Xu[3] and Huamin Wang[1]

[1]The Ohio State University
[2]Adobe Research
[3]Zhejiang University

(a) Installation plan (b) Printed shell pieces (c) Assembled squirrel

Figure 1: *A squirrel example. Our computational system generates interlocking shell pieces from a printable model. Without using glue or screw, the shell pieces in (b), printed by a Form1+ SLA printer, can be securely locked in the assembled configuration as (c) shows. The color pieces and arrows in (a) visualize the installation plan, including both the installation order and the installation directions of the pieces.*

Abstract

While the 3D printing technology has become increasingly popular in recent years, it suffers from two critical limitations: expensive printing material and long printing time. An effective solution is to hollow the 3D model into a shell and print the shell by parts. Unfortunately, making shell pieces tightly assembled and easy to disassemble seem to be two contradictory conditions, and there exists no easy way to satisfy them at the same time yet. In this paper, we present a computational system to design an interlocking structure of a partitioned shell model, which uses only male and female connectors to lock shell pieces in the assembled configuration. Given a mesh segmentation input, our system automatically finds an optimal installation plan specifying both the installation order and the installation directions of the pieces, and then builds the models of the shell pieces using optimized shell thickness and connector sizes. To find the optimal installation plan, we develop simulation-based and data-driven metrics, and we incorporate them into an optimal plan search algorithm with fast pruning and local optimization strategies. The whole system is automatic, except for the shape design of the key piece. The interlocking structure does not introduce new gaps on the outer surface, which would become noticeable inevitably due to limited printer precision. Our experiment shows that the assembled object is strong against separation, yet still easy to disassemble.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Physically based modeling

1. Introduction

The recent advance in the 3D printing technology offers great flexibility in the design and manufacturing of customized objects for various purposes. This technology, however, suffers from two major limitations: *expensive printing material* and *long printing time*. The material cost is directly related to the volume of a 3D model. A natural way to reduce the cost is to just print the model surface as a shell. The printing time depends on many factors, including the model height, especially for powder-based and SLA printers. When a model is tall, it consists of many printing layers and each layer needs time to get solidified. To shorten the printing time, we can further divide a shell into pieces and build them in a flattened

configuration. Doing this also indirectly reduces the material cost, since less material is needed to form the support structure, even though it is not part of the model. Figure 1 shows a squirrel example. By printing its partitioned shell pieces, we can reduce the volume from 468cm^3 to 131cm^3 and the printing time from 13.5 hours to 6 hours using a Form 1+ SLA printer.

Besides reducing the material cost and the printing time, the partitioned shell idea, first proposed by Vanek et al. [VGB*14b], has other significant advantages. It allows a small and affordable printer to build a large 3D model piece by piece, even in parallel [LBRM12]. It reduces the space when a printed object is not in use for storage and transportation purposes [YCL*15]. It even makes a printed object replaceable and reusable by parts. This advantage can be of crucial importance, if part of an object is vulnerable to damage during practical use.

While researchers have extensively studied the partitioning of a 3D model and its shell, they paid little attention to a basic yet important question: *how can we assemble the partitioned pieces together?* A common practice in previous research [VGB*14b, CZL*15] is to use the glue. To glue two pieces together, we must keep them in their right positions before the glue dries. This is rather inconvenient when the shell pieces are thin. Gluing also undermines the replaceability and the reusability of a printed object, even if the glue is washable. Alternatively, we can use simple male and female connectors [LBRM12, YCL*15]. In that case, only friction can prevent the pieces from separation, so the whole structure is fragile and it cannot withstand large impacts. Other fastening options, such as screws, are often infeasible for a partitioned shell model, either because they rely too much on the precision of a 3D printer or because they can leave too many gaps on the surface.

Inspired by recent work on interlocking structures [SFCO12, FSY*15, SFLF15], we present a computational system to design the interlocking structure of a partitioned shell model, which uses male and female connectors to lock the pieces into their assembled positions. Besides easy assembling and disassembling, we believe that a good design should also have the following qualities:

- **Flexibility.** The modeling algorithm should flexibly handle curved 3D models and they should not place too many restrictions on the mesh segmentation input.
- **Strength.** The structure should be strong enough to prevent an assembled object from separation under reasonable impacts, e.g., when falling from a height of a table.
- **Unnoticeable gaps.** It is inevitable to have gaps on the model surface, due to limited printer precision. The design should introduce fewer and smaller gaps, if possible.

To achieve these goals by our computational system, we make the following technical contributions.

- **Shape modeling.** Given an installation plan that includes both the installation order and the installation directions, we develop a geometric modeling approach to build the shape of each shell piece. The connectors enforce each piece to be separable only in its separation direction. To further secure the key piece of the structure, we propose a unique key connector design based on user guidance.
- **Plan metrics.** We present both simulation and data-driven ways to evaluate the quality of an installation plan. We show how to formulate simulation metrics by running simulation under a worst-case scenario, and we demonstrate projective dynamics is particularly suitable for this task. While simulation metrics are more accurate but expensive, the data-driven metric offers a fast estimation based on the local geometry.
- **Optimal plan search.** We use a randomized graph search algorithm to find an optimal installation plan. A naïve implementation of this algorithm converges slowly. Fortunately, the locality nature of the data-driven metric allows us to develop pruning and local optimization strategies for fast plan search. In the end, we apply simulation metrics to ensure that shell pieces cannot be separated as clusters.

Thanks to the interlocking structure, the partitioned shell models are easy to assemble and disassemble, as Figure 1 shows. The printed objects have sufficient strength against falling and squeezing impacts, as demonstrated in our experiment. The whole system has a reasonable scalability and its result is relatively insensitive to the mesh segmentation input.

2. Related Work

Printable shape design. In recent years, researchers have studied many printable shape design problems, including strength improvement [SVB*12, US13, WWY*13, ZPZ13, LSZ*14], articulated bodies [BBJP12, CCA*12], appearance [DWP*10, CLD*13], elastic shape design [CZXZ14], support structure [DHL14, VGB14a], spinnability [BWBSH14], and elasticity [BBO*10, PZM*15, SBR*15].

Our work is closely related to those trying to model a 3D shape by parts. [LOMI11] proposed to divide a furniture model into pieces with connectors. [LBRM12] worked on planar partitioning of a large model, in order to build the pieces by small printers. [HLZCO14] developed an algorithm to decompose a model into pyramidal shapes for additive 3D printing. [DPW*14] studied how to assemble a self-supporting structure from its partitioned parts. [SSL*14] proposed a data-driven method for designing 3D models with template parts and connectors. [SDW*16] built a coarse internal base structure to support a partitioned shell model. Researchers [VGB*14b, YCL*15, CZL*15, Att15] have also been interested in decomposing and packing 3D models into a printing volume or an arbitrary container. Doing this can save the printing material spent on the support structure, shorten the printing time, and/or reduce the space when an object is not in use.

Shell modeling. A straightforward way to save the printing material is to just print the outer surface of the 3D model as a shell. [PKZ04, LFL09] modeled the shell by displacing surface vertices along the gradient of the distance function, so they can maintain the correspondences between the outer surface and the inner surface. Such correspondences are often not needed in 3D printing applications. So researchers [SVB*12, VGB*14b] have also adopted level set methods and modeled the shells from isosurfaces. To make the shell satisfy certain constraints, such as standing on a table, [MAB*15] presented a reduced order framework for optimizing spatially varying shell thickness.

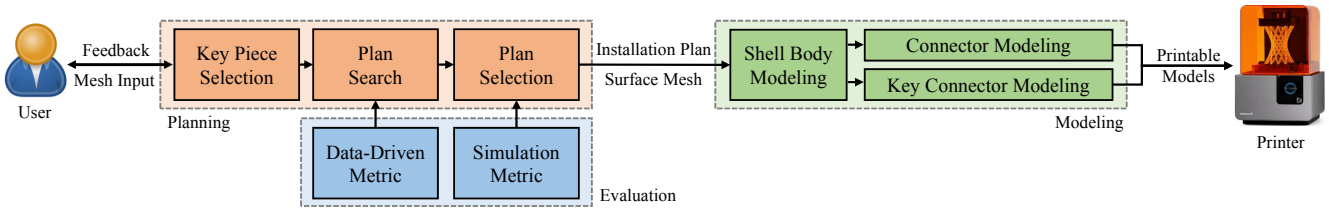
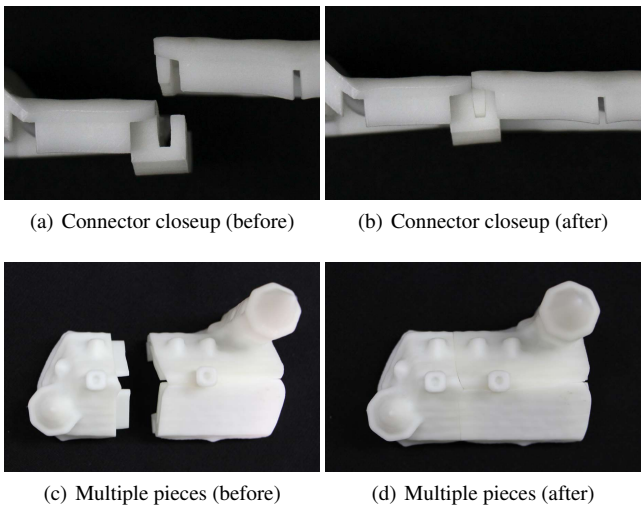


Figure 3: *The system pipeline. Our system can be divided into three major components: planning, evaluation, and modeling. Most of these components are automatic, except for the design of the key piece that still demands some user guidance at this time.*



(a) Connector closeup (before) (b) Connector closeup (after)

(c) Multiple pieces (before) (d) Multiple pieces (after)

Figure 2: *Connectors and pieces before and after installation. The connectors modeled on the boundary between two pieces ensure that they are installable/separable only in one direction. If some pieces have already been installed, the next piece being installed will determine the installation directions of the related boundary connectors, as shown in (c).*

To prevent a shell model from easy collapse, [WWY*13] proposed to strengthen the shell structure by adding internal struts. Alternatively, [SVB*12,LSZ*14] suggested to perform stress analysis and hollow low-stress regions only. In mechanical engineering, shape optimization for strength improvement is an extensively studied research topic. A recent survey on this topic can be found in [SINP05]. Different from previous works, our work is more focused on connector failures than the yield of the shell itself.

Puzzles and interlocking structures. The history of interlocking structures can be traced back to ancient Asia and Europe, when people used interlocking joints to build wood architecture and furniture. The pioneer work of Cutler [Cut78, Cut94] first employed computers to find solutions to six-piece interlocking burr puzzles. Later, [XLF*11] generated burr puzzles from a 3D model, by replicating and connecting a known knot structure. [SFCO12, SFLF15] presented a voxel-based approach to decompose a 3D model into interlocking pieces with various complexities. [FSY*15] formulated a scheme to break a furniture complex into globally interlocking assemblies. [LFL09] created shell-based 3D puzzles, using polyomino tilings over parameterized surfaces. [SZ15] developed a computational approach for constructing puzzles with twisty joints. The interlocking structure of flat panels has also been studied by graphics researchers [HBA12,CPMS14,SDW*16]. While many previous works on printable puzzles and interlocking structures relied on specific patterns or installation directions, we are interested in handling arbitrary surface mesh segmentation inputs with only a few restrictions.

3. Overview

The main idea behind this work is to model male and female connectors on the boundary between two adjacent shell pieces, so that they are separable only in a single direction as Figure 2 shows. The opposite of this *separation direction* is the *installation direction*, the only direction for separated pieces to be assembled back together. The directions of the two pieces are also opposite to each other: the installation of the male piece is identical to the separation direction of the female piece, and vice versa. Although we specify the installation direction at each piece, we found it is convenient to assign directions to its boundaries and connectors as well, depending on how the piece gets connected to its neighbors. When a piece has many neighbors and its connectors have conflicting separation directions, it gets interlocked into the assembled state. A special case is the *key piece* [FSY*15], which is installed later than all of its neighbors. In that case, the direction of the key piece determines all of its connector directions, so it can be not interlocked. In our model, there is only one key piece, the very last piece being installed.

Our system pipeline is illustrated in Figure 3. We intentionally assume that the user will provide the segmentation of the surface mesh as an input. This offers the user some flexibility in designing shell pieces. For example, the user can segment the mesh at curved locations to make gaps less noticeable [YCL*15]. In case the user is not familiar with meshes, our system provides a basic mesh segmentation component using the CGAL library as well. The output of our system is a series of shell piece meshes, which are directly produced by the modeling process described in Section 4. To make the interlocking structure effective, the modeling process must use a good installation plan specifying the installation order and the installation directions. In Section 5, we present different evaluation metrics to quantify the quality of an installation plan. Later in Section 6, we discuss our randomized algorithm with pruning and local optimization strategies for fast plan search. Our system is free of user intervention, except for the shape design of the key piece. Once the system automatically identifies the key piece, it asks the user to refine its shape and connector locations in the mesh segmentation input, so that it can be secured as well.

4. Interlocking Shell Piece Model

To begin with, we would like to present our interlocking shell piece model, given a mesh segmentation input and an installation plan. To interlock shell pieces and eliminate their relative movements after installation, our design places male and female connectors on the boundary between two adjacent shell pieces. We will first study the modeling of the shell piece body and its connectors in Subsection 4.1 and 4.2, respectively. Next, we will optimize thickness and size parameters used by this model in Subsection 4.3.

4.1. Piece Body Model

The modeling of an individual shell piece body is a relative simple problem. First, we partition the 3D space into a regular grid and compute the distance from each grid node to the segmented mesh surface. Doing this allows us to divide the whole 3D volume by multiple level set functions, each of which represents the 3D region closest to one segmented mesh patch as did in [YCL*15]. Let H be the shell thickness. We formulate the level set function of the shell piece body k as:

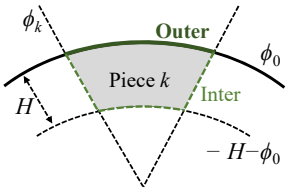


Figure 4: *The level sets that define the body of piece k.*

$$\psi_k(\mathbf{x}) = \min\left(\phi_0(\mathbf{x}), \phi_k(\mathbf{x}), -H - \phi_0(\mathbf{x})\right), \quad (1)$$

in which ϕ_0 is the level set of the outer surface mesh and ϕ_k is the level set representing the region close to piece k, as Figure 4 shows. To prevent shell piece bodies from being blocked in their separation directions, we perform intersection tests and carve their level sets, if any intersection is found. Finally, we use the marching cube method to create the inner surface mesh from modified level sets, compute the intersection between the inner surface mesh and the outer surface mesh, and merge the intersection results to form the surface mesh of a shell piece body. The reason we do not use level sets to reconstruct the whole surface mesh immediately is because it may fail to recover the details on the original outer surface.

4.2. Connector Model

Our next goal is to model the connectors between every two adjacent pieces, according to the separation directions of the pieces found by the optimal plan search in Section 6. Without loss of generality, let us consider a single edge e connecting two vertices $\mathbf{x}_{e0}$ and $\mathbf{x}_{e1}$ on the inner boundary, as Figure 5a shows. Let $\mathbf{u}_{e0}$ and $\mathbf{u}_{e1}$ be the tangent of the inner curve at the two vertices, and $\mathbf{s}_{e0} = \mathbf{s}_{e1}$ be the separation direction of the green piece which is assigned to the whole boundary, if we assume that the green piece is installed later than the blue piece. We compute two upper directions: $\mathbf{n}_{e0} = -\mathbf{u}_{e0} \times \mathbf{s}_{e0}$ and $\mathbf{n}_{e1} = -\mathbf{u}_{e1} \times \mathbf{s}_{e1}$. Using all of these normalized directions, we can expand edge e with distance D into a polyhedron, which acts as an enclosure around e. Note that $\mathbf{n}_{e0}$ and $\mathbf{n}_{e1}$ are not surface normals, and they typically do not lie in the boundary surface between the two pieces. If the male connector grows forward as the cross section view shows in Figure 5b, we model its pin by expanding the edge with shorter distances in the $-\mathbf{s}_{e0}$ direction and the $-\mathbf{n}_{e0}$ direction. To model the pin's attachment to the male piece, we take the union of the pin and the back side of the enclosure, and then calculate its intersection with the inner surface mesh. Similarly, we model the socket using the front side of the enclosure, and then attach it to the female piece by mesh operations. We perform these operations on meshes, not on level sets, because level sets must be in high resolution to provide sufficient accuracy. Backward connectors can be modeled in the same way as shown in Figure 5c, with similar mesh operations. We note that our modeling method allows the installation direction of the green piece to be in an upward direction as well, if we treat it as a female piece instead.

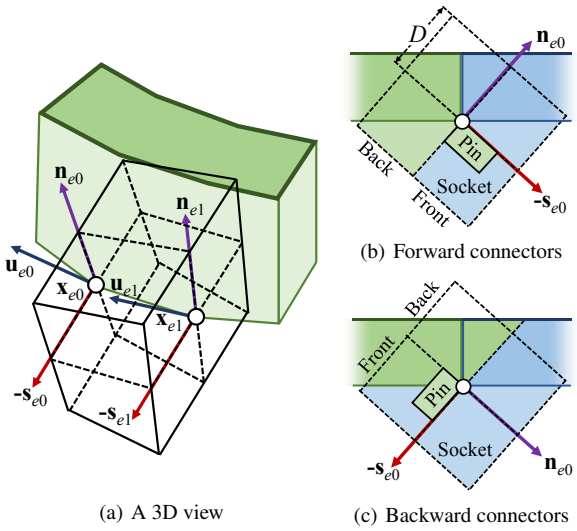
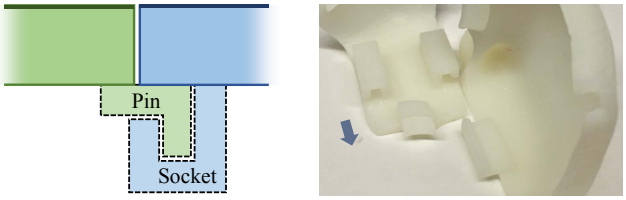


Figure 5: *The modeling of male and female connectors. We build a polyhedral enclosure around each inner boundary edge to facilitate the modeling of its connectors. The connectors prevent the two pieces from leaving each other, except in a given direction.*

Although we try to create connectors for every inner boundary edge, there are two typical exceptions. First, we do not build connectors, if an edge is too close to the ends of a boundary between two pieces. This prevents intersections among multiple connectors modeled on multiple boundaries. Second, if $\mathbf{n}_{e0}$ and $\mathbf{n}_{e1}$ of an edge are too different, the polyhedral enclosure may be in self-intersection. So we do not model its connectors as well. Figure 2a and 2c contain the connectors of printed shell pieces. When two edges are adjacent, their connectors are naturally merged.

Key connectors. The key piece, i.e., the very last piece being installed, is unique in that it cannot be interlocked by its neighbors, so we must rely on friction to make it tightly connected to the rest of the model. Our idea is to slide the key piece in by a special key connector design, as the cross section view in Figure 6a shows. These connectors are elongated in the separation direction, to offer a sufficient contact area for friction. We note that this design can cause intersections between key connectors and shell bodies in the separation direction. Fortunately, this issue does not exist, if the boundaries between the key piece and its neighbors are straight and aligned with the separation direction. Our current solution requires the user to enforce this condition on the mesh segmentation input and to specify key connector locations on the boundaries.



(a) A cross section view (b) A photograph

Figure 6: *The modeling of key connectors. Unlike other connectors, these connectors have their separation directions perpendicular to the cross section view shown in (a).*

Figure 6b shows a photograph of a key piece with its connectors. Our experiment verifies that these connectors can effectively prevent the separation of the key piece. Occasionally, they can even cause the key piece to become difficult to assemble, due to limited printer resolution. So we recommend to leave some gaps between male and female key connectors for easy disassembling.

4.3. Parameter Optimization

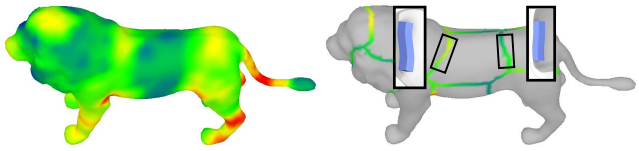
The modeling process described in Subsection 4.1 and 4.2 uses two important parameters: the shell thickness H and the connector size D. In this subsection, we will optimize their values to make the whole interlocking structure strong against impacts.

Thickness optimization. For fast modeling and optimization, our system cannot afford performing highly detailed structural analysis. Instead, we assume that the thickness is uniform over the whole shell and we ignore the existence of the segmentation and the connectors. Using a shell finite element formulation [Log11], our stress analysis tool derives the stress field over the shell under given external forces. Our goal is to find the minimum thickness H, such that the stress result satisfies unilateral constraints:

$$\max_{c\in\mathcal{C}} \left(\sigma_s^c\right) < \sigma_{\max}, \quad \text{for all } s \in \mathcal{S}, \tag{2}$$

in which $\mathcal{S}$ is the set of shell elements, $\mathcal{C}$ is the set of test cases, σ_s^c is the von Mises stress computed from the Cauchy stress tensor of element s in the c-th case, and $\sigma_{\max}$ is the yield strength of the printing material. Figure 7a shows the maximum stress result of a lion example. For the photopolymer resin material used in some of our experiment, we set $\sigma_{\max}$=42MPa. Because there is only one parameter needed for optimization, we search for its minimum value in a bisection way. Given an initial thickness interval [h_0=0mm, h_1=10mm], such that the constraints are satisfied at h_1 but not at h_0, we evaluate the constraints at $\frac{1}{2}(h_0+h_1)$, select the sub-interval accordingly, and then start another iteration. We terminate the iterative procedure once the interval is sufficiently small and select h_1 as the result.

We construct our test cases to mimic collisions with the ground floor. Specifically, we rotate the model to a random orientation in each test case, and then apply constraints and forces at its top and bottom regions respectively. The set contains 12 test cases in total. Suppose that the collision is completely inelastic and it happens within a short time Δt, we calculate the force at vertex i within impact zones as: $\mathbf{f}_i = -m_i\mathbf{v}_i/\Delta t$, where m_i and $\mathbf{v}_i$ are the vertex mass and velocity. We typically set $\Delta t = 1/300$s and $\mathbf{v}_i = 5$m/s, which is the speed after falling from a height of 1.25m.



(a) Maximum stress (b) Stress over the boundaries

Figure 7: *A stress analysis result of a lion example. Based on the analysis, we optimize the shell thickness and the connector sizes.*

Although other optimization techniques such as topology optimization [BS04, SM13] can provide more sophisticated solutions, they are computationally expensive and it is difficult to synthesize the optimization results generated by different test cases. By using a uniform thickness, however, our method can quickly process the test cases and find a thickness that works in the worst case.

Connector size optimization. Next we must optimize the connector size. The size of a connector is determined by its polyhedral enclosure, which is expanded from an inner boundary edge with distance D as Figure 5b shows. So the question becomes how to find an optimal D. Although we can define D as a variable at each vertex, we choose not to do so, to avoid the modeling complexity that follows. Instead we treat the distance as a constant for connectors on the same boundary and we estimate its value by a simple linear model. Let $\max_{c,i\in\mathcal{B}_j}(\sigma_i^c)$ be the maximum von Mises stress at any vertex i on boundary $\mathcal{B}_j$. We compute the expansion distance D_j on boundary $\mathcal{B}_j$ as:

$$D_j = D_{\min} + \min\left(\alpha \max_{c,i\in\mathcal{B}_j}\left(\sigma_i^c\right), 1\right)(D_{\max} - D_{\min}), \tag{3}$$

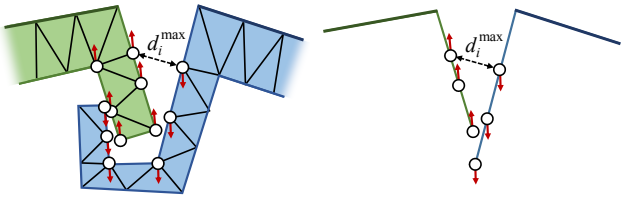
in which α is a control coefficient, and $D_{\min}$=2mm and $D_{\max}$=4mm are the minimum and maximum distances. Figure 7b illustrates the connectors of a shell piece with different sizes. When the stress on a boundary is large, Equation 3 increases the distance value to make the connectors bigger, as expected.

5. Plan Metrics

To derive a good installation plan later in Subsection 6, we must understand what is a good installation plan first. In this section, we will present three evaluation metrics with different computational costs. We will use them for different purposes during the optimal plan search later in Section 6.

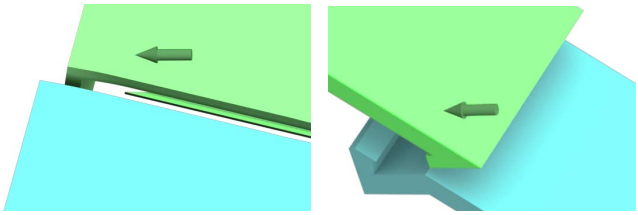
5.1. Volume-based Metric

Perhaps the most straightforward way to evaluate an installation plan is to construct the surface meshes of shell pieces, convert them into tetrahedral meshes, and then perform volumetric finite element analysis. The goal of this analysis is to find out how easily shell pieces can be separated, when they are under external forces. Since it is impractical to test all of the possible forces under different scenarios, we construct our tests in a worst-case way. Given the shell pieces and the separation directions, we apply uniform



(a) Volumetric model (b) Surface model

Figure 8: *Volumetric and surface meshes used for our simulation-based metrics. These metrics use the maximum displacement between male and female connector vertices created from the same boundary vertex to evaluate the installation plan quality.*



(a) Shallow connectors (b) Ill-shaped connectors

Figure 9: *Bad connectors. The issues associated with these connectors are caused by the separation direction, which is penalized in our plan metrics by a weight function.*

forces at connector vertices in their separation directions as shown in Figure 8a, and then run the simulation with the key piece being fixed as constraints. We perform tetrahedron intersection tests and penalty forces to handle and resolve collisions among shell pieces. We do not consider friction forces in our simulation, because unlike the key connector, the normal connectors cannot provide sufficient contact area, and the friction forces are typically insignificant compared with the strain forces caused by the high stiffness of the printing material. Once the simulation gets stabilized at an elastostatic state, we calculate the maximum displacement $d_i^{\max}$ between male and female connector vertices created from the same boundary vertex i. This displacement evaluates the response of the model to external forces, but it has not considered the quality of the connectors yet. During the modeling process described in Section 4, the separation direction can actually cause two issues. First, if $\mathbf{s}_i$ is close to the tangent direction $\pm\mathbf{u}_i$, the effective depth of its connectors decreases, as shown in Figure 9a. Second, if $\mathbf{s}_i$ is close to the binormal direction $\mathbf{v}_i$ as in Figure 10, the connectors are too backward and ill-shaped, as Figure 9b shows. We use a weight function to describe the intrinsic connector quality related to these two issues:

$$w_i = \max\left(|\mathbf{s}_i \cdot \mathbf{u}_i|, \max(\mathbf{s}_i \cdot \mathbf{v}_i, 0)\right) + \beta, \tag{4}$$

in which β is the base weight. We then formulate the overall plan quality as: $Q = -\max w_i \mathbf{d}_i^{\max}$, for any boundary vertex i. A higher plan quality score indicates that the model constructed under this plan should be more resistant against the separation problem.

To prevent the deformation from being too large, we apply a uniform separation force of 10.0N/m on the connectors and discard those plans which contain totally separable pieces. With small deformations, we can use a simple linear elastic model for the simulation, with a Young's modulus of 2.0GPa and a Poisson's ratio of 0.35. To represent the squirrel example as shown in Figure 1, our tetrahedral mesh contains 18K tetrahedral elements in total. Each shell piece contains 1.6K to 3.7K tetrahedral elements, including 1.0K to 1.4K elements for the shell body and 0.6K to 2.3K elements for the connectors. In average, our simulator needs approximately 3.6 seconds to evaluate an whole installation plan.

5.2. Surface-based Metric

While we can use the volume-based metric to generate ground truth for comparison purposes, it is too computationally expensive and we cannot use it extensively in the optimal plan search to be discussed in Section 6. Our solution is to simplify a volumetric mesh model into a triangular mesh model, as Figure 8b shows. The shell body is represented by the segmented surface mesh and the connectors become stripes of triangles attached to the shell body.

Our simulator is built upon the recent success of projective dynamics [LBOK13, BML*14]. Different from other simulation techniques [BW98, MG04] that try to solve a single Newton iteration, projective dynamics formulates implicit time integration of a dynamical system into a nonlinear optimization problem:

$$\mathbf{x}_{t+1} = \arg\min_{\mathbf{x}} \left\{ \frac{1}{2h^2} \|\mathbf{x} - \mathbf{y}_t\|_{\mathbf{M}}^2 + \sum_c \frac{k_c}{2} \|\mathbf{A}_c\mathbf{x} - \mathbf{B}_c\bar{\mathbf{x}}_c\|^2 \right\}, \tag{5}$$

in which $\mathbf{x}_{t+1}$ is the vertex position vector at time $t+1$, $\mathbf{y}_t$ is the expected vertex position vector under body forces, h is the time step, $\mathbf{M}$ is the mass matrix, c is an edge constraint, k_c is the stiffness, and $\mathbf{A}_c$ and $\mathbf{B}_c$ are the constant matrices for the constraint [BML*14]. In our simulation, c can be any triangle edge for planar elasticity, or any hinge edge for bending elasticity. The basic idea behind projective dynamics is to iteratively solve two steps: a local step that projects $\mathbf{x}$ into $\bar{\mathbf{x}}_c$ by each constraint c, and a global step that solves Equation 5 as a linear system when $\bar{\mathbf{x}}_c$ is constant. The key advantage of projective dynamics is that the matrix of the linear system involved in the global step is constant, which can be pre-factorized for fast solve. This is particularly important to us, since different plans use different external forces but the same matrix.

Similar to the volume-based metric, our surface-based metric defines boundary forces in a worst-case way. The main difference is in collision detection and handling. The surface-based metric handles a collision as a positional constraint: a vertex belonging to the male connector must be projected onto its closest triangle belonging to the female connector, but not vice versa. One challenge posed by such a constraint is that it changes the system matrix, as the closest pair varies in each iteration. Fortunately, we found it is still safe to use the original system matrix, without considering the contribution of these varying collision constraints. The reason is because projective dynamics can be fundamentally interpreted as a gradient descent method with the matrix acting as the preconditioner. Since the matrix is strictly diagonally dominant, the method is guaranteed to converge as long as the step length is sufficiently small. Our experiment shows there is even no need to use an adaptive step length, since collision constraints are much more sparse than other constraints. Once the simulator generates the maximum displacement

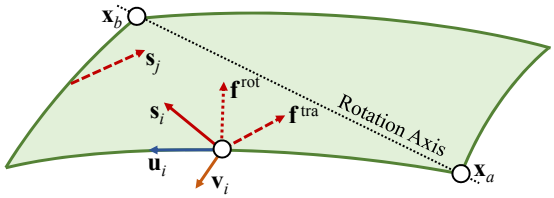


Figure 10: *Data-driven metric model. This model assumes that the resistance of a shell piece against separation can be derived the resistance of each vertex i against translation and rotation, using two types of testing directions* $\mathbf{f}^{\texttt{tra}}$ *and* $\mathbf{f}^{\texttt{rot}}$.

for every boundary vertex, we use the weight function in Equation 4 to evaluate the overall plan quality as did in Subsection 5.1.

We apply the same stiffness for the bending, stretch and positional constraint. Our experiment shows that our surface-based simulator can evaluate a whole installation plan in less than 0.2 seconds, for the squirrel example shown in Figure 1. This is nearly 20 times faster than the volume-based simulator. Thanks to multi-threading, we can further reduce the evaluation time of 100 plans to less than 10 seconds.

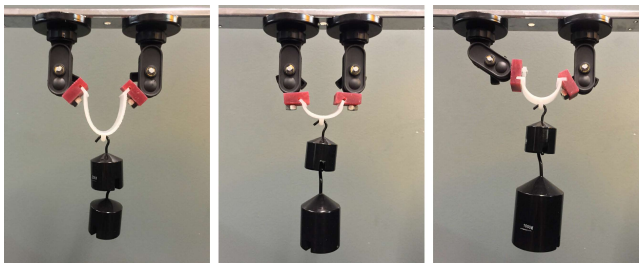
5.3. Boundary-based Data-Driven Metric

While the surface-based metric is significantly faster, our system still cannot afford using it to evaluate millions of installation plans during the optimal plan search. So we need some inexpensive yet plausible way to estimate the plan quality and use that estimation to avoid unnecessary simulations.

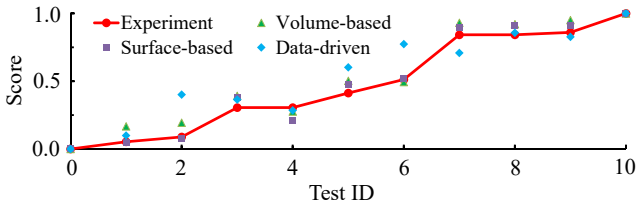
We propose to build a novel metric in a local and data-driven fashion. Suppose that the surface and the segmentation are both sufficiently smooth. We can approximate the local neighborhood of a boundary edge as two half-planes. The connectors are two narrow stripes underneath this edge. The separation direction controls both the depth and the orientation of the connectors. Since the connector depth has already been considered in the weight function in Equation 4, we just have to consider the connector orientation with only one degree of freedom which is the freedom of rotation about the edge. To know how this local neighborhood responds to forces in a specific testing direction, we tessellate the neighborhood into triangle meshes, run our surface-based simulator in Subsection 5.2, and use the maximum displacement to form the quality metric as before. The outer ring of this neighborhood is treated as fixed constraints in simulation. We select 8 connector orientations and 26 testing directions, including 6 axial directions and 20 in-plane and out-of-plane diagonal directions. In total, our local test set contains 208 cases and it takes less than five seconds to simulate by our simulator. Given the simulated data, we can then predict the displacement of a local neighborhood by a piecewise linear function: $d(\mathbf{u},\mathbf{v},\mathbf{s},\mathbf{f})$, in which $\mathbf{u}$ and $\mathbf{v}$ specify the orientation of the neighborhood, $\mathbf{s}$ is the separation direction, and $\mathbf{f}$ is the direction of the testing force.

Our data-driven metric assumes that the maximum displacement of a shell piece k can be predicted from the displacement of its boundary vertex i under a number of test cases:

$$d_k = \max_{i\in\mathcal{B}_k,\mathbf{f}} w_i d(\mathbf{u}_i,\mathbf{v}_i,\mathbf{s}_i,\mathbf{f}), \tag{6}$$



(a) Experiment setup



(b) Scores compared with experimental ground truth

Figure 11: *Metric evaluation by real-world experiment. The metrics roughly agree with the ground truth which is provided by the experiment in (a). All of the scores are normalized.*

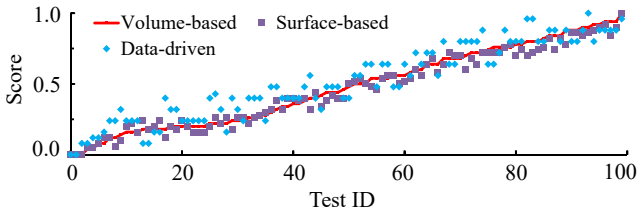


Figure 12: *Metric evaluation by synthetic experiment. The metrics roughly agree with the ground truth which is provided by the volume-based metric. All of the scores are normalized.*

in which w_i is the weight function penalizing bad connectors, as in Equation 4. Our test case includes translational ones that simply use $\mathbf{f}^{\texttt{tra}} = \mathbf{s}_j$, the separation direction of boundary j. It also includes rotational ones which treat any two boundary ends $\mathbf{x}_a$ and $\mathbf{x}_b$ as a rotation axis, as Figure 10 shows, and the testing direction can be calculated as: $\mathbf{f}^{\texttt{rot}} = \pm\frac{(\mathbf{x}_i-\mathbf{x}_a)\times(\mathbf{x}_b-\mathbf{x}_a)}{\|(\mathbf{x}_i-\mathbf{x}_a)\times(\mathbf{x}_b-\mathbf{x}_a)\|}$. Intuitively, these tests estimate the resistance of the shell piece against translation and rotation, by trying to pull or pry it out from its neighbors. Once we get the resistance of every shell piece, we combine them together to form the data-driven quality of the whole plan: $Q = -\max d_k$. Similar to simulation-based metrics, the data-driven metric uses a higher score to indicate more resistance against separation.

5.4. Metric Comparisons

To evaluate the effectiveness of our metrics on predicting the connector quality, we design a controlled experiment as Figure 11a shows. In this experiment, we use two tripod heads to secure female pieces (in red) in different orientations. We then put calibrated weights under the male piece (in white) and measure the maximum force for the pieces to get separated. This experiment result is used as the ground truth to evaluate the scores provided by the three met-

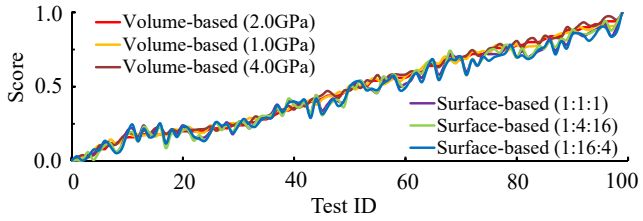


Figure 13: *Metric evaluation with different stiffness parameters. The volume-based metric agrees between different Young's modulus and the surface-based metric agrees between different ratios of bending to stretch to positional stiffness.*

rics under the same setup, as Figure 11b shows. Here we normalize all of the scores to make the comparison fair. The experiment shows that our scores are roughly aligned with the experiment data. Next we would like to evaluate our metrics on installation plans. Since it is difficult to test a whole model by real-world experiment, we treat the volume-based metric as the ground truth and use our metrics to evaluate 100 random installation plans of the squirrel example. Being significantly faster, the surface-based metric still agrees well with the volume-based metric as Figure 12 shows. The data-driven metric can also provide a plausible prediction on the plan quality, although its score is not always reliable. Finally, we test the same set of installation plans with different stiffness parameters and find that physical accuracy is not strictly required for our simulators and there is no need to tune stiffness parameters as long as the simulation result is plausible, as Figure 13 shows.

6. Installation Planning

Given the metrics, we can now discuss how to find an optimal installation plan, which contains both the installation order and the installation directions. As in Subsection 5.3, we consider only 26 installation directions, including 6 axial directions and 20 diagonal directions in the world space. We will start our discussion with the key piece in Subsection 6.1, and then describe the search algorithm with pruning and optimization strategies in Subsection 6.2.1.

6.1. Key Piece Detection

Our first job is to identify the key piece, i.e., the very last piece being assembled. Unlike other pieces, the key piece relies on friction rather than interlocking to stay with the rest of the model. Since an object typically does not experience much stress in the tangent plane, we choose to slide the key piece in by defining the separation direction in the tangent plane, as discussed in Subsection 4.2.

We use two criteria to select the key piece: 1) can it be well approximated by its tangent plane; and 2) does it have a valid installation direction? To do tangent plane approximation, we apply least squares fitting and the RANSAC method to fit a plane to its boundary vertices. The reason we use RANSAC rather than a closed form is to handle pieces whose surfaces are only partially flat. To test whether a valid direction exists, we sample eight directions in the tangent plane and check the intersection in each direction as described later in Subsection 6.2.1. In the end, we identify the key piece and its valid installation direction as the one with the highest approximation quality. Figure 1 shows that the method selects the key piece on the flat bottom of a squirrel model, as expected. We note that the shape of the selected key piece should still be refined manually to meet the shape requirement outlined in Subsection 4.2.

6.2. The Search Algorithm

Once we identify the key piece, we can formulate installation planning as a backward graph search problem on the connectivity graph over the pieces, from the last piece to the first piece. A naïve solution is to enumerate all of the orders and the directions, and find the best one among them. However, a simple calculation shows even a model with only 10 pieces can have billions of installation plans, so it is impractical to evaluate them all by our data-driven metric. Our solution is to randomly determine the order and the directions during the graph search. This randomized approach alone converges slowly, because it must be very lucky to find a better installation plan every time. To address this problem, we focus our research on developing two strategies: *plan pruning* and *local optimization*.

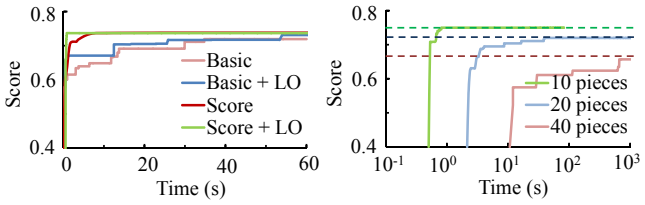
6.2.1. Plan Pruning

We use plan pruning to avoid graph search branches, if they will lead to low-score or problematic plans. Here we apply three pruning techniques on the shell piece to be added into the installation order path. These techniques are presented in the same order as they are implemented, due to their computational costs. If any pruning technique reports that the piece is not suitable for inclusion, the algorithm performs backtracking and keeps trying to obtain an acceptable plan. If the search algorithm still cannot find a plan after a number of tries, it restarts the graph search from scratch to prevent itself from becoming a brute-force search.

Connectivity pruning. We require that every shell piece, except for the first piece and the last piece, must have at least one neighbor being installed earlier and one neighbor being installed later. The first requirement prevents shell pieces from becoming disconnected during installation, and the second requirement ensures that shell pieces are all interlocked, except for the key piece. During the backward graph search, we enforce these two requirements as two conditions on the piece newly added into the installation order: it must be connected to those added earlier, and it cannot disconnect the pieces that have been not added yet. These two conditions can be easily tested upon the connectivity of the shell pieces. If a piece violates any of the two conditions, it cannot be added into the installation order right away.

Intersection pruning. We consider installing each shell piece in a single direction only. Therefore, the whole plan is invalid, if any piece intersects with the pieces installed earlier in its installation direction. Based on this observation, we implement a pruning strategy in the backward graph search by checking whether a piece intersects with those installed earlier, i.e., those not added into the installation order yet. If so, this piece cannot be added immediately.

Since we have already addressed the intersection issue among shell body interiors in Subsection 4.1, here we just need to test if external shell surfaces intersect. To do so, we develop a visibility test that checks whether a piece is partially occluded by others. Specifically, we define a virtual orthographic camera facing in each



(a) Different techniques (b) Different piece numbers

Figure 14: *The algorithm performance using different techniques in (a) and different numbers of pieces in (b). The dash lines in (b) represents the ground truth in each case.*

separation direction, and then render the outer surface of each piece into a depth map on the GPU. Using these depth maps, we perform standard occlusion tests between every two pieces and store the results into a binary matrix per direction. Now the actual intersection test during the graph search can be quickly done by table lookups.

Score-based pruning. The nature of the data-driven metric described in Subsection 5.3 implies that the overall quality of an installation plan is determined by the worst piece, i.e., the piece with the lowest score. Given the best installation plan we have discovered so far, we can evaluate the scores of the pieces on the fly, and discard low-score installation plans before they are formed. One advantage of doing the backward graph search is that when a piece is about to be added into the installation order, the separation directions of its boundaries are all known, because the separation direction of a piece installed later than its neighbor determines the separation direction of the boundary between them. So similar to the other strategies, the score-based pruning strategy can be implemented by placing another condition on the piece to be added into the installation order. Since we consider only a limited number of separation directions, many steps involved in Equation 6 can be precomputed to save the running cost.

6.2.2. Local Optimization

The pruning strategies avoid wasting computational time on useless installation plans, but they still cannot fully address the low convergence issue. To further boost the performance of our randomized search algorithm, we propose to improve the quality of the formed installation plan in a local and greedy fashion. Specifically, given the piece with the lowest score, we enumerate its direction and its neighboring directions, reevaluate their scores, and then update the plan if the score can be higher. Connectivity and intersection conditions must be rechecked to avoid an invalid plan, after every modification step. The whole optimization process terminates, once we cannot improve the plan score any further. Our experiment shows local optimization typically takes five to ten iterations and it can increase the score of an installation plan by up to 40 percent.

6.2.3. Implementation and Analysis

Figure 14 illustrates the performance of our algorithm when using different techniques and different numbers of pieces. Since it is impractical to find out the ground truth from a brute-force search, we let the algorithm run for six hours and select its result as the ground truth, drawn as a dash line in Figure 14. Figure 14a shows that both

Name	# of Pieces	h (mm)	Volume Ratio	Time Ratio	Comp. Cost (s)
Squirrel	11	2.2	28.0%	44.4%	130.7
Lion	12	2.7	47.6%	48.2%	118.0
Castle	11	2.4	39.8%	55.6%	117.7
Vase	9	2.3	45.5%	23.2%	126.7
Angel	10	2.0	28.1%	49.1%	130.3
Astronaut	11	2.7	54.0%	68.6%	116.4
Dinosaur	10	2.1	32.9%	46.4%	120.2
Laurana	10	2.0	26.1%	58.5%	125.1
Laurana	40	2.0	46.2%	67.1%	306.6
Budda	12	2.4	24.5%	47.8%	134.0
Car	11	1.9	26.6%	48.3%	131.2
Fist	13	2.0	20.4%	46.9%	145.1
Octopus	9	2.8	50.3%	61.9%	98.3

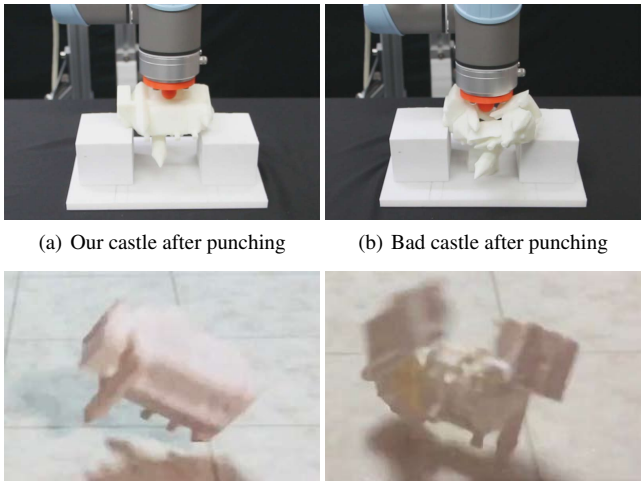
Table 1: *Statistics showing the number of pieces, the shell thickness, the ratio of the shell volume to the original volume, the ratio of the shell printing time to the original printing time, and the computational cost.*

score-based pruning and local optimization are effective in improving the result quality, as expected. Figure 14b shows that although the algorithm converges slower as the number of pieces increases, the algorithm is still affordable when there are 40 pieces. The algorithm performance can become problematic if there are even more pieces, but we believe that it is unnecessary to use too many pieces in practice. After all, our goal is to design a simple and convenient structure, not a puzzle.

Our randomized graph search algorithm depends heavily on the data-driven metric, which assumes that the overall plan quality can be determined by the qualities of the shell pieces. This assumption is not always plausible. For example, a cluster of pieces can be easily separable, while each individual one is not. Simulation-based metrics do not have this problem, as they simulate all of the pieces together in a worst-case scenario. Our solution is to maintain the top 100 installation plans found by the algorithm using the data-driven metric, and then choose the best one among them using the surface-simulation-based metric.

7. Results and Discussions

(Please watch the supplemental video for more detailed examples.) We implemented our system and tested its performance on an Intel Core i7-4790K 3.6GHz processor. The system uses CGAL to perform some of its mesh operations. Our examples, summarized in Table 1 and displayed in Figure 18, cover a wide range of 3D printable shapes. They demonstrate that partitioned shell models can be used to effectively reduce both the printing volume and the printing time as expected. The printing time is estimated using the PreForm 2.0.2 Software, which is configured for Form1+ SLA 3D printers. To evaluate the printability of our models, we use a series of 3D printers, including a Form1+ SLA 3D printer, a Stratasys Objet500 Connex3 PolyJet printer, and a Shining iSLA-650 3D printer. The printed shell pieces and the assembled objects are shown in Figure 17. We also conduct an experiment with 10 participants to test the usability of our installation plans. Given only the information of the installation order and the installation directions of the pieces, it



(a) Our castle after punching (b) Bad castle after punching

(c) Our castle after collision (d) Bad castle after collision

Figure 15: *Stress tests. Our assembled castle can withstand punching and collision impacts, as shown in (a) and (c). In contrast, if a model does not get its pieces securely locked, or if its connectors are too small, it can easily break as shown in (b) and (d).*

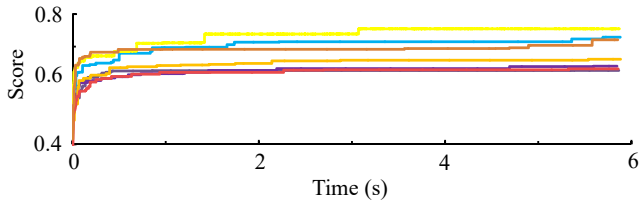
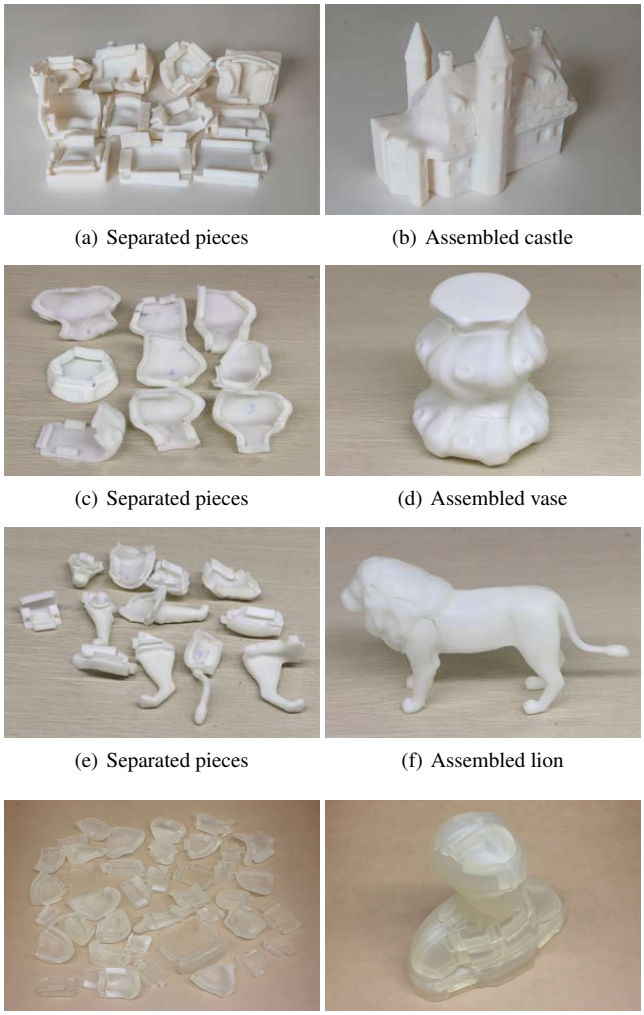


Figure 16: *Plan search results using multiple mesh segmentation inputs, each of which is represented by a curve.*

takes a user less than two minutes to assemble or disassemble each model, except the 40-piece model which takes about 10 minutes.

Stress tests. To evaluate the strength of our models against separation, we test them under two different conditions. First, we use a UR5 universal robot arm to punch the assembled models at the same contact point from the top. Figure 15a and 15b demonstrate that our castle model can easily withstand a 70N force, while the bad one whose plan quality score is 50% lower cannot. Second, we drop the assembled models from a height of 2m above a hard floor. Figure 15c and 15d show that our castle model can maintain its shape after collision with the floor, while the bad one whose connectors are 50% smaller shatters. Note that these models are slightly stronger than what we expected, probably because they are printed by a PolyJet printer, not a SLA printer.

Performance. Table 1 provides the total computational cost spent on each example, including the plan search cost, the simulation cost, and the modeling cost. The plan search cost can be controlled by the user. In our experiment, we typically spend 60 seconds on the plan search, to ensure the result quality. The rest of the computational cost is mainly caused by the geometric modeling step. Both volumetric and mesh operations need considerable computational time. Fortunately, it is straightforward to model shell pieces



(a) Separated pieces (b) Assembled castle

(c) Separated pieces (d) Assembled vase

(e) Separated pieces (f) Assembled lion

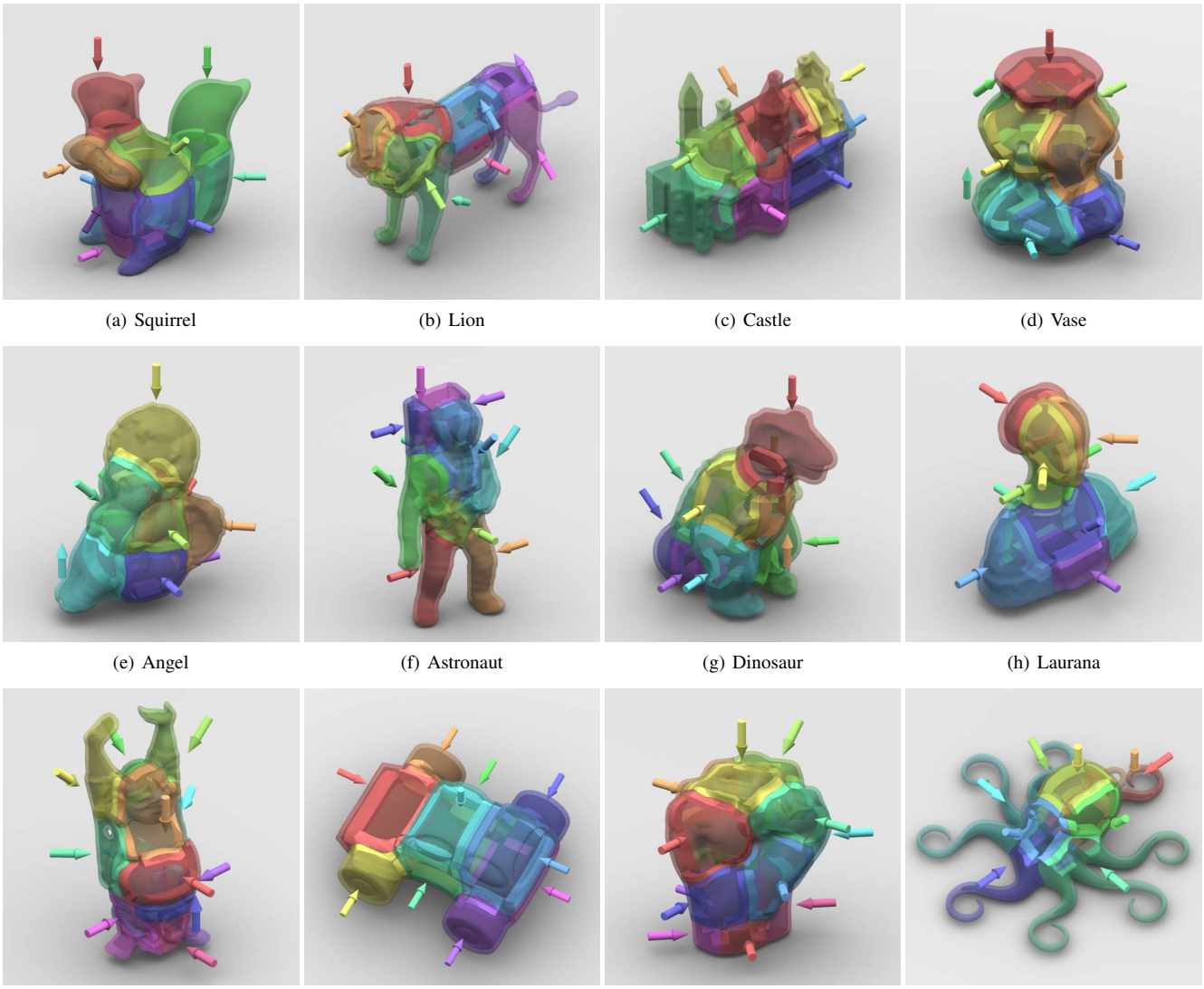
(g) Separated pieces (h) Assembled laurana

Figure 17: *Printed pieces and assembled models.*

in parallel. The simulation step is less expensive than we thought, thanks to parallelization as well.

Sensitivity to mesh segmentation. While we allow the user to provide the mesh segmentation input, it is interesting to know how different mesh segmentation inputs can affect the result qualities. So we create eight mesh segmentation inputs for the same fist example, two of which are generated manually and the rest of which are generated by CGAL using methods like shape diameter, random cuts, fitting primitives and k-means. The curves in Figure 16 illustrate how the results vary, according to different inputs. In general, the result quality depends on the connectivity of the pieces and the smoothness of the boundary curves. Although it is uncommon, the result can be totally unacceptable due to the intrinsic fault of the input. In that case, our system warns the user this issue and where the weakness is, so he/she can adjust it accordingly.

Limitations. Although our system can flexibly handle different mesh segmentations, its results can vary. In particular, it has dif-



(a) Squirrel (b) Lion (c) Castle (d) Vase

(e) Angel (f) Astronaut (g) Dinosaur (h) Laurana

(i) Budda (j) Car (k) Fist (l) Octopus

Figure 18: *Our examples and results. We use the same color scheme as in Figure 1 to visualize the installation order of the shell pieces. The colored arrow next to each piece specifies its installation direction.*

ficulty in building connectors, if pieces are too small or boundary curves are too curvy. It also requires each piece to have a sufficient number of neighbors. In the worst case when a piece has only one neighbor, it cannot be interlocked and its connectors must be designed as a key piece. Currently, our system relies on the user to segment the key piece by following a certain pattern. Without doing this, the key piece may not be secure enough against separation. The plan search algorithm considers only a limited number of separation directions and it assumes that each piece is separable in a single direction during disassembling. Our data-driven metric used by the plan search algorithm is fast and local, but it fails to consider global deformation as in simulation-based metrics. Finally, our system is less suitable for printers that have limited precision or use too much supporting materials, such as FDM printers.

8. Conclusions and Future Work

In this paper, we present a novel computational system to design the interlocking structure of partitioned shell pieces. Our research shows that this system is highly automatic and flexible, except for the design of the key piece. The interlocking structure is useable and effective against separation, as expected.

Looking into the future, we would like to find better and automatic ways to handle the modeling of the key piece. We also would like to see if struts can be added to improve the strength of the shell structure. We have not found an easy way to assemble these struts yet. To make our system more flexible, we are interested in more generic connector designs for a wider range of mesh segmentation inputs. Finally, we plan to reduce the computational cost of our system through code optimization and hardware acceleration.

9. Acknowledgments

This work was supported by NSF grant IIS-1524992. We also thank Adobe and the State Key Lab of CAD&CG at Zhejiang University for additional support through funding and equipment.

References

[Att15] ATTENE M.: Shapes in a box: Disassembling 3d objects for efficient packing and fabrication. *Computer Graphics Forum 34*, 8 (2015), 64–76. 2

[BBJP12] BÄCHER M., BICKEL B., JAMES D. L., PFISTER H.: Fabricating articulated characters from skinned meshes. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (July 2012), 47:1–47:9. 2

[BBO*10] BICKEL B., BÄCHER M., OTADUY M. A., LEE H. R., PFISTER H., GROSS M., MATUSIK W.: Design and fabrication of materials with desired deformation behavior. *ACM Trans. Graph. (SIGGRAPH) 29*, 4 (July 2010), 63:1–63:10. 2

[BML*14] BOUAZIZ S., MARTIN S., LIU T., KAVAN L., PAULY M.: Projective dynamics: Fusing constraint projections for fast simulation. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (July 2014), 154:1–154:11. 6

[BS04] BENDSOE M. P., SIGMUND O.: *Topology Optimization: Theory, Methods and Applications*. Springer, Feb. 2004. 5

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques* (New York, NY, USA, 1998), SIGGRAPH '98, ACM, pp. 43–54. 6

[BWBSH14] BACHER M., WHITING E., BICKEL B., SORKINE-HORNUNG O.: Spin-It: Optimizing moment of inertia for spinnable objects. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (August 2014), 96:1–96:10. 2

[CCA*12] CALÌ J., CALIAN D. A., AMATI C., KLEINBERGER R., STEED A., KAUTZ J., WEYRICH T.: 3D-printing of non-assembly, articulated models. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (Nov. 2012), 130:1–130:8. 2

[CLD*13] CHEN D., LEVIN D. I. W., DIDYK P., SITTHI-AMORN P., MATUSIK W.: Spec2Fab: A reducer-tuner model for translating specifications to 3D prints. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (2013), 135:1–135:10. 2

[CPMS14] CIGNONI P., PIETRONI N., MALOMO L., SCOPIGNO R.: Field-aligned mesh joinery. *ACM Trans. Graph. 33*, 1 (Feb. 2014), 11:1–11:12. 3

[Cut78] CUTLER W. H.: The six-piece burr. *Journal of Recreational Mathematics 10*, 4 (1978), 241–250. 3

[Cut94] CUTLER W. H.: *A computer analysis of all 6-piece burrs*. Self published, 1994. 3

[CZL*15] CHEN X., ZHANG H., LIN J., HU R., LU L., HUANG Q., BENES B., COHEN-OR D., CHEN B.: Dapper: Decompose-and-pack for 3D printing. *ACM Trans. Graph. (SIGGRAPH Asia) 34*, 6 (Oct. 2015), 213:1–213:12. 2

[CZXZ14] CHEN X., ZHENG C., XU W., ZHOU K.: An asymptotic numerical method for inverse elastic shape design. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (August 2014), 95:1–95:11. 2

[DHL14] DUMAS J., HERGEL J., LEFEBVRE S.: Bridging the gap: Automated steady scaffoldings for 3D printing. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (August 2014), 98:1–98:10. 2

[DPW*14] DEUSS M., PANOZZO D., WHITING E., LIU Y., BLOCK P., SORKINE-HORNUNG O., PAULY M.: Assembling self-supporting structures. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6 (Nov. 2014), 214:1–214:10. 2

[DWP*10] DONG Y., WANG J., PELLACINI F., TONG X., GUO B.: Fabricating spatially-varying subsurface scattering. *ACM Trans. Graph. (SIGGRAPH) 29*, 4 (2010), 62:1–62:10. 2

[FSY*15] FU C.-W., SONG P., YAN X., YANG L. W., JAYARAMAN P. K., COHEN-OR D.: Computational interlocking furniture assembly. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (2015), 091:1–091:11. 2, 3

[HBA12] HILDEBRAND K., BICKEL B., ALEXA M.: Crdbrd: Shape fabrication by sliding planar slices. *Comput. Graph. Forum 31*, 2pt3 (May 2012), 583–592. 3

[HLZCO14] HU R., LI H., ZHANG H., COHEN-OR D.: Approximate pyramidal shape decomposition. *ACM Trans. Graph. (SIGGRAPH Asia) 33*, 6 (Nov. 2014), 213:1–213:12. 2

[LBOK13] LIU T., BARGTEIL A. W., O'BRIEN J. F., KAVAN L.: Fast simulation of mass-spring systems. *ACM Trans. Graph. (SIGGRAPH Asia) 32*, 6 (Nov. 2013), 214:1–214:7. 6

[LBRM12] LUO L., BARAN I., RUSINKIEWICZ S., MATUSIK W.: Chopper: Partitioning models into 3D-printable parts. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (Nov. 2012), 129:1–129:9. 2

[LFL09] LO K.-Y., FU C.-W., LI H.: 3D polyomino puzzle. *ACM Trans. Graph. (SIGGRAPH Asia) 28*, 5 (Dec. 2009), 157:1–157:8. 2, 3

[Log11] LOGAN D. L.: *A First Course in the Finite Element Method (Fifth Edition)*. Cengage Learning, 2011. 5

[LOMI11] LAU M., OHGAWARA A., MITANI J., IGARASHI T.: Converting 3D furniture models to fabricatable parts and connectors. *ACM Trans. Graph. (SIGGRAPH) 30*, 4 (July 2011), 85:1–85:6. 2

[LSZ*14] LU L., SHARF A., ZHAO H., WEI Y., FAN Q., CHEN X., SAVOYE Y., TU C., COHEN-OR D., CHEN B.: Build-to-last: Strength to weight 3D printed objects. *ACM Trans. Graph. (SIGGRAPH) 33*, 4 (August 2014), 97:1–97:10. 2, 3

[MAB*15] MUSIALSKI P., AUZINGER T., BIRSAK M., WIMMER M., KOBBELT L.: Reduced-order shape optimization using offset surfaces. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (July 2015), 102:1–102:9. 2

[MG04] MÜLLER M., GROSS M.: Interactive virtual materials. In *Proceedings of Graphics Interface* (2004), pp. 239–246. 6

[PKZ04] PENG J., KRISTJANSSON D., ZORIN D.: Interactive modeling of topologically complex geometric detail. *ACM Trans. Graph. (SIGGRAPH) 23*, 3 (Aug. 2004), 635–643. 2

[PZM*15] PANETTA J., ZHOU Q., MALOMO L., PIETRONI N., CIGNONI P., ZORIN D.: Elastic textures for additive fabrication. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (July 2015), 135:1–135:12. 2

[SBR*15] SCHUMACHER C., BICKEL B., RYS J., MARSCHNER S., DARAIO C., GROSS M.: Microstructures to control elasticity in 3D printing. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (2015), 136:1–136:13. 2

[SDW*16] SONG P., DENG B., WANG Z., DONG Z., LI W., FU C.-W., LIU L.: Cofifab: Coarse-to-fine fabrication of large 3d objects. *ACM Trans. Graph. 35*, 4 (July 2016), 45:1–45:11. 2, 3

[SFCO12] SONG P., FU C.-W., COHEN-OR D.: Recursive interlocking puzzles. *ACM Trans. Graph. (SIGGRAPH Asia) 31*, 6 (Nov. 2012), 128:1–128:10. 2, 3

[SFLF15] SONG P., FU Z., LIU L., FU C.-W.: Printing 3D objects with interlocking parts. *Comput. Aided Geom. Des. (GMP) 35*, C (May 2015), 137–148. 2, 3

[SINP05] SAITOU K., IZUI K., NISHIWAKI S., PAPALAMBROS P.: A survey of structural optimization in mechanical product development. *J. Comput. Inf. Sci. Eng 5*, 3 (2005), 214–226. 3

[SM13] SIGMUND O., MAUTE K.: Topology optimization approaches. *Structural and Multidisciplinary Optimization 48*, 6 (2013), 1031–1055. 5

[SSL*14] SCHULZ A., SHAMIR A., LEVIN D. I. W., SITTHI-AMORN P., MATUSIK W.: Design and fabrication by example. *ACM Trans. Graph. 33*, 4 (July 2014), 62:1–62:11. 2

[SVB*12] STAVA O., VANEK J., BENES B., CARR N., MĚCH R.: Stress relief: Improving structural strength of 3D printable objects. *ACM Trans. Graph. (SIGGRAPH) 31*, 4 (July 2012), 48:1–48:11. 2, 3

[SZ15] SUN T., ZHENG C.: Computational design of twisty joints and puzzles. *ACM Trans. Graph. (SIGGRAPH) 34*, 4 (July 2015), 101:1–101:11. 3

[US13] UMETANI N., SCHMIDT R.: Cross-sectional structural analysis for 3d printing optimization. In *SIGGRAPH Asia 2013 Technical Briefs* (New York, NY, USA, 2013), SA '13, ACM, pp. 5:1–5:4. 2

[VGB14a] VANEK J., GALICIA J. A. G., BENES B.: Clever support: Efficient support structure generation for digital fabrication. *Comput. Graph. Forum 33*, 5 (Aug. 2014), 117–125. 2

[VGB*14b] VANEK J., GALICIA J. A. G., BENES B., MĚCH R., CARR N., STAVA O., MILLER G. S.: Packmerger: A 3D print volume optimizer. *Comp. Graph. Forum 33*, 6 (Sept. 2014), 322–332. 2

[WWY*13] WANG W., WANG T. Y., YANG Z., LIU L., TONG X., TONG W., DENG J., CHEN F., LIU X.: Cost-effective printing of 3D objects with skin-frame structures. *ACM Trans. Graph. (SIGGRAPH Asia) 32*, 6 (Nov. 2013), 177:1–177:10. 2, 3

[XLF*11] XIN S., LAI C.-F., FU C.-W., WONG T.-T., HE Y., COHEN-OR D.: Making burr puzzles from 3D models. *ACM Trans. Graph. (SIGGRAPH) 30*, 4 (July 2011), 97:1–97:8. 3

[YCL*15] YAO M., CHEN Z., LUO L., WANG R., WANG H.: Level-set-based partitioning and packing optimization of a printable model. *ACM Trans. Graph. (SIGGRAPH Asia) 34*, 6 (Oct. 2015), 214:1–214:11. 2, 3, 4

[ZPZ13] ZHOU Q., PANETTA J., ZORIN D.: Worst-case structural analysis. *ACM Trans. Graph. (SIGGRAPH) 32*, 4 (July 2013), 137:1–137:12. 2