Printing 3D Objects with Interlocking Parts

Peng Song[a,*], Zhongqi Fu[b], Ligang Liu[b], Chi-Wing Fu[c]

[a]*School of Computer Science and Technology, University of Science and Technology of China*
[b]*School of Mathematical Sciences, University of Science and Technology of China*
[c]*School of Computer Engineering, Nanyang Technological University, Singapore*

Abstract

Recent advances in 3D printing technologies bring wide range of applications from fast prototyping to product manufacturing. However, one intrinsic limitation of 3D printing is that we cannot fabricate a single object that is larger than the working volume of a 3D printer. To address this issue, we may partition the given object into 3D parts of manageable sizes for printing, and then assemble the object from the printed 3D parts. Rather than using connectors, glue, or skew, we propose to connect the printed 3D parts by 3D interlocking such that the assembled object can be not only repeatedly disassembled and reassembled, but also strongly connected by the parts' own geometry. To achieve these, we develop a voxelization-based approach to partition a given 3D model into 3D interlocking parts. To guarantee the generated 3D parts to be structurally sound and well-connected by 3D interlocking, we deform the local geometry of the 3D model to avoid voxel fragmentation, employ internal voxels to create initial interlocking parts, and analyze the local shape within voxels to guide the final parts construction. We demonstrate the effectiveness of our approach on 3D models with a variety of shapes, and realize some of them by 3D printing.

Keywords: Interlocking, Shape Partitioning, Voxelization, 3D Printing

1. Introduction

3D printing is an additive manufacturing technology, facilitating convenient and rapid fabrication of physical objects of almost any shape. It has a wide range of practical applications, from fast product prototyping, product development, 3D visualization, to distributed manufacturing of larger-sized objects such as machine parts.

However, 3D printing has an intrinsic limitation: a 3D printer cannot directly print an object whose size is greater than the printer's working volume. This practical limitation has been pointed out recently by Luo et al. [1], who proposed a solution to partition a given 3D object into parts for 3D printing and then assemble the printed parts together to reconstruct the object. Besides addressing the intrinsic limitation, this approach has several other advantages. First, it facilitates cost-effective maintenance since we only need to print a replacement part for a corresponding broken part rather than reprinting the entire object. Second, this approach is good for storage and transport since we can disassemble a large 3D object to save space and avoid breaking it during the transportation. Lastly, similar to Lego bricks, we could reassemble an existing 3D object, change some of its parts, and even reconfigure it for alternative designs and appearance.

To connect 3D printed parts, there are some common approaches, e.g., by male and female connectors [1, 2] and by glue [3]. For the case of male and female connectors, though they are generally practical, they may not provide sufficient structural strength to sustain the parts connections. Moreover, for these tiny printed features, they may be broken easily during the object assembly or the transportation. For the case of glue, though it could tightly join the printed 3D parts together, it is a permanent connection, discouraging object reassembly, cost-effective maintenance, and reconfiguration. In sharp contrast, we take a 3D interlocking

[*]Corresponding author. Email address: songpeng@ustc.edu.cn

approach [4, 5] to construct and connect printed 3D parts to form an object assembly. By this, we can overcome the above mentioned issues.

Connecting parts by 3D interlocking has several advantages: i) first, parts connections are achieved by the parts' own geometry without requiring the creation of extra tiny connectors such as male and female connectors; ii) the assembled object can be repeatedly disassembled and reassembled, facilitating cost-effective maintenance, storage, and transportation; iii) 3D interlocking is known to be strong, evidenced by its usage in long-standing architectural wooden structures; hence, this connection method allows us to achieve stronger 3D parts connections, which are in fact supported by inter-parts blockage with their geometry; and iv) lastly, it enables us to produce 3D parts with clean and smooth surface without hole drilling and protrusion.

In this paper, our goal is to partition a given 3D object into interlocking parts for 3D printing and object assembly. This is a highly challenging and unexplored problem, since we have to enforce not only the complicated 3D interlocking requirement [4, 5], but also the geometric and dimensional requirements on the printed 3D parts, as well as maintaining the 3D object appearance after its assembly. The requirements to be considered when we develop the computational method are listed as below:

- **3D Interlocking:** the printed 3D parts should interlock one another, yet can be assembled and disassembled;
- **Printable Parts:** while being not too small, each printed 3D part should fit into the working volume of the target 3D printer;
- **Structural Soundness:** we should avoid thin and/or weak features on the 3D parts since such features could be easily broken during the object printing, assembly, or transportation;
- **Strong Connection:** we should achieve strong 3D parts connections by ensuring strong blockage among the 3D parts in the 3D interlocking assembly; and
- **Aesthetics:** lastly, we should avoid having cutting seams that pass through salient regions on the assembled object surface since this affects the object appearance.

Note that state-of-the-art methods [4, 5] for creating 3D interlocking structures are insufficient to handle the problem since they mainly focus on the 3D interlocking requirement without considering the 3D printing issues such as structural soundness and object appearance.

To achieve the above requirements, we develop a novel voxelization-based approach to construct interlocking 3D parts from a given 3D model. Our technical contributions are three-fold. First, we develop a new framework that can create interlocking 3D parts from 3D models of general shape, where novel ideas include voxelizing the 3D model with local shape analysis, employing internal voxels to create initial interlocking parts, and attaching boundary voxels to initial parts while retaining the 3D interlocking. Second, we propose to deform the input model surface subject to the voxelization, which helps to avoid fragmented and disconnected shape features on the generated 3D parts. Lastly, we propose the shape and saliency connection graphs encoding local shape information to guide the parts geometric construction for achieving the parts structural soundness and aesthetics requirements. We have demonstrated the effectiveness of our approach on 3D models with a variety of shapes, and fabricated some of them by 3D printing to validate the parts connection capability, as well as their structural soundness.

2. Related Work

3D Fabrication. A number of research works in computer graphics have studied different aspects of 3D fabrication, e.g., deformation behavior [6], mechanical characters [7, 8], articulated models [9, 10], spinnable fabrication [11], 3D shape balancing [12], and printing material reduction [13, 14].

Specifically, some of them focus on structural issues in 3D printing. Telea and Jalba [15] devised several metrics to assess the printability of 3D shapes. Stava et al. [16] proposed an automatic method to detect and correct structural issues in 3D models before printing them. Zhou et al. [17] determined the worst-case loads in printing a 3D object. Umetani and Schmidt [18] detected structural weakness of 3D objects for 3D printing optimization.

Fabrication by 3D Parts. Partitioning an object into parts for 3D fabrication can be used for creating 3D furniture models with assemblable parts [19], for approximating objects with planar boundary pieces [20] or planar slices [21, 22, 23], and for building architectural structures [24]. Recently, Zhou et al. [25] proposed a method to produce 3D printing objects that can be folded into a box.

In particular, some research works focus on partitioning a solid 3D shape into disjoint parts for 3D fabrication. Medellín et al. [26] subdivided a 3D shape into parts based on a regular grid. Hao et al. [27] decomposed a large complex model into simpler 3D parts by using curvature-based partitioning. Luo et al. [1] partitioned a large 3D object into smaller parts by using planer cuts, where male and female connectors are generated on parts for making the connections. Hildebrand et al. [28] addressed the directional bias issue in 3D printing by segmenting a 3D model into a few parts and assigning an optimal printing orientation for each part. Vanek et al. [29] improved the 3D printing efficiency by converting 3D objects into shells and breaking them into parts that can be glued together.

Our work also partitions a 3D shape into smaller parts for 3D fabrication. But in sharp contrast, we achieve 3D parts connections by 3D interlocking rather than by glue or by tiny geometry like male and female connectors. Our approach has several distinctive advantages over existing approaches as highlighted earlier in the introduction.

3D Interlocking Puzzles. Designing 3D interlocking puzzles is extremely difficult even for highly skilled puzzle designers [30]. So far, there are only a few computational methods that can construct 3D interlocking structures. Xin et al. [4] created interlocking puzzles from 3D models by replicating and connecting a specific six-piece burr configuration, but since this method requires a centralized bulky structure to achieve 3D interlocking, it is highly restrictive, and could not deal with complex shapes and general topology. Later, Song et al. [5] developed a recursive method to construct 3D interlocking structures by iteratively extracting puzzle pieces from a voxelized model. The produced puzzle pieces are polycubes with cubical appearance, and cutting seams are randomly arranged on the object surface, resulting in rather distracting appearance. If we directly apply constructive solid geometry (CSG) operations to refine their puzzle pieces with the original object surface, disconnected, fragmented, and weakly-connected components would be easily produced on the puzzle pieces, thus making [5] inadequate for 3D printing, see Section 6 for detailed comparison.

Compared with the above works, our method enforces various fabrication requirements while achieving 3D interlocking, which is already nontrivial on its own. In particular, our method not only can automatically and flexibly create interlocking 3D parts from a given object model of general shape, but also can produce 3D parts that are structural sound and avoid obvious cutting seams on salient features of the assembled object. By this, a given object is ready for being 3D printed with its parts.

3. Overview

In this section, we first discuss the challenges of partitioning a given 3D shape into interlocking 3D parts, and then give an overview of our approach.

Challenges of 3D Shape Partitioning. For a given 3D shape, there exist different ways of partitioning it into parts, for example, plane- [1], curvature- [27], and voxelization-based [26, 25] approaches. This work employs the voxelization-based partitioning approach to create interlocking parts since i) cubical voxel space facilitates the design of interlocking structures like [5]; ii) shape analysis of parts can be simplified by computing the local shape within voxels and connecting local shapes between neighboring voxels to form component parts; and iii) parts dimension can be easily controlled by the bounding box in voxel space.

However, there are two major challenges to construct interlocking parts by voxelization. First, when voxelizing a 3D model, a voxel may be full, partial, or empty according to the local shape within the voxel, see Figure 1(a). More importantly, a resulting partial voxel may have various kinds of local shapes, e.g., a tiny fragment (Figure 1(b) (case 1)), disconnected fragments within a voxel (Figure 1(b) (case 2)), or a thin structure (Figure 1(b) (case 3)). Hence, a naive voxelization could easily produce a large amount of partial voxels with undesired (e.g., tiny or disconnected) local shapes. Second, when we later connect neighboring voxels to form 3D component parts in the assembly, some problematic parts can be generated if we arbitrarily connect neighboring voxels into parts, e.g., parts with a weak fragment (see P1 in Figure 1(c)) and parts with disconnected fragments (see P2 in Figure 1(c)). For the case of P2, we may not just solve it by separately attaching the two fragments to different parts since this may violate the 3D interlocking.

To address the above issues, we propose to analyze the local shape within each partial voxel, and apply the information to guide the voxelization process as well as the 3D parts construction. By this, we can achieve structurally-sound component parts (see P3 in Figure 1(c)) in the 3D interlocking assembly.

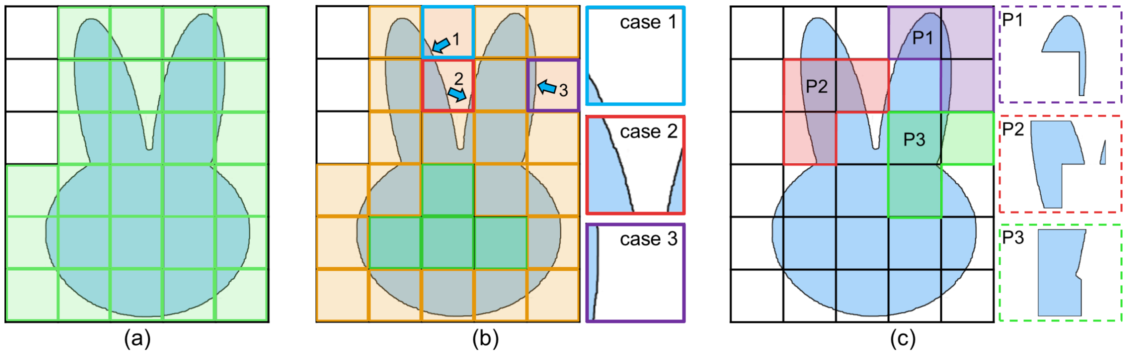

Figure 1: Voxelization-based shape partitioning. (a) A 2D shape and its voxelization; (b) full (in green) and partial (in orange/blue/red/purple) voxels: the three partial voxels in blue, red, and purple show cases of tiny fragments (case 1), disconnected fragments (case 2), and thin structures (case 3), respectively; (c) naively connecting voxels into component parts could easily produce thin and disconnected 3D parts (in purple and red) rather than structurally-sound 3D parts (in green).

Overview of Our Approach. We take a 3D watertight surface as input since only meshes with well-defined exterior boundary and closed volume can be 3D printed, see Figure 2(a). Note that an input mesh that does not satisfy this requirement (e.g., with self-intersections) can be repaired by methods such as [31].

Given an input watertight surface, we first place a voxel grid in the 3D object space and slightly adjust the voxel size to reduce the number of undesired partial voxels. Then, we voxelize the input model. For partial voxels with tiny fragment(s), we avoid the fragment(s) by slightly deforming the model geometry locally. Next, we measure the connection strength between neighboring (partial) voxels and build a shape connection graph by analyzing the local shapes (Figure 2(b)). We also identify salient appearance features on the model by [32], and represent the information by a saliency connection graph.

After the above preparation works, we construct an initial set of interlocking 3D parts from the internal voxels (Figure 2(c)). Hence, the initial 3D parts can fulfill the structural soundness and strong connection requirements since all employed voxels are large enough and thus provide sufficient blockage to enforce the 3D interlocking. After that, we attach the remaining partial voxels one by one to the constructed interlocking parts as guided by the shape connection graph (Figure 2(d)) without breaking the fulfilled requirements. Later, we enforce the aesthetics requirement by reassigning boundary voxels among neighboring parts as guided by the saliency connection graph, aiming to avoid cutting seams on salient object features.

Lastly, we construct the geometry of the final 3D parts (Figure 2(e)) by using CSG intersection between the voxelized parts (Figure 2(d)) and the input mesh (Figure 2(a)).

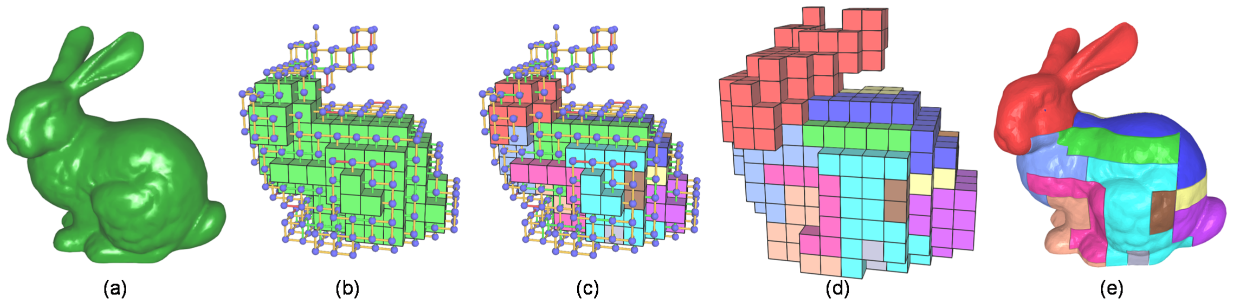

Figure 2: Overview. (a) A 3D watertight model as input; (b) voxelization and shape analysis; (c) creating initial interlocking parts; (d) attaching boundary voxels to the parts; and (e) refining the parts' geometry by CSG intersection with the input mesh model.

4. Voxelization and Shape Analysis

This section presents the various preparation works we have before constructing the geometry of interlocking 3D parts from an input 3D model. This includes voxelization, local shape deformation, and the generation of shape connection graph and saliency connection graph.

Voxelization Method. Given the input watertight 3D model, we first voxelize it by [33], which casts parallel rays through the model and uses parity count to classify voxels as interior or exterior. We improve this method by casting additional parallel rays to not only classify voxels but also estimate the local shape within each voxel. For a bounded volume of $W\times H\times D$ voxels, we first build a $(K\times W+1)\times(K\times H+1)\times(K\times D+1)$ uniform 3D point grid within it, where we have $K+1$ sample points along the edge of each voxel. Then we cast $(K\times W+1)\times(K\times H+1)$ rays through the model to classify each sample point as interior or exterior, where each ray passes through $(K\times D+1)$ sample points. Note that we can do ray casting along any of the three major axis of the grid. Figure 3 illustrates our voxelization method with a 2D example having $K=6$.

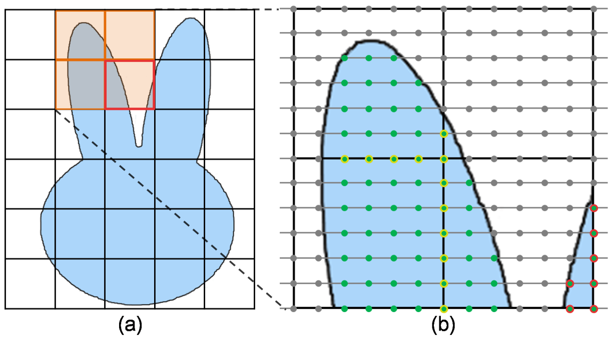

Figure 3: (a) We take a 2D shape to illustrate our voxelization method; (b) classified interior and exterior sample points are in green and gray, respectively; the bottom-right voxel is detected as a disconnected partial voxel by breadth first search from an interior sample point (see red circles).

Next, we calculate properties of the local shape within each voxel based on the classified sample points:

- *Local shape volume* is estimated by counting the number of interior sample points and then computing their coverage percentage within the voxel;
- *Type of the voxel* is classified as full, partial, or empty based on the value of local shape volume, i.e., 1, (0,1), and 0, respectively;
- *Contacting face area* for a partial voxel is estimated by computing the percentage of interior sample points on the voxel's contacting face with its neighbor, see the yellowish-green sample points in Figure 3(b);
- *Local shape connectivity* in a partial voxel is checked by i) breadth first search from an interior sample point, and ii) checking whether it reaches all interior sample points in the voxel.

To facilitate shape partitioning, partial voxels with tiny or disconnected local shape should be avoided, see again Figure 1. Hence, for voxel grid with a given resolution, we first adjust the voxel size slightly to reduce the number of undesired partial voxels. This is achieved by randomly sampling the voxel size within a small range and selecting the voxel size corresponding to the least number of undesired voxels. Note that we do not adjust the 3D model pose (orientation) to avoid undesired partial voxels since this could result in non-axis-aligned cuts on the model, affecting both object aesthetics and parts assembly.

Local Deformation to Avoid Tiny Fragments. For undesired partial voxels in the voxelization, we propose to resolve them by locally deforming the 3D model geometry. To constrain the local deformation within an undesired voxel and its neighbors, we remesh the 3D model [34, 35] such that the mesh triangles are regular and much smaller than the voxel size. Note that we can detect tiny fragments by looking at partial voxels whose local shape volume is smaller than a threshold, which is set as 0.01 in our experiments.

For a partial voxel with a tiny fragment (Figure 4(a)), we compute three variables for the fragment by examining (averaging) the mesh triangles related to the tiny fragment in the voxel: its centroid V_c, mean surface normal $\vec{N}$, and radius of a bounding sphere r. For a partial voxel with disconnected tiny fragments (Figure 4(c)), the same three variables are computed for each tiny fragment. To avoid a tiny fragment, we could locally deform the 3D model geometry by iteratively moving the related mesh vertices by using

$$V_i{}^{k+1} \;=\; V_i{}^k - \alpha \; (2r - dist(V_i{}^k, V_c)) \; \vec{N} \;, \tag{1}$$

where V_i is a mesh vertex within a bounding sphere centered at V_c with radius $2r$, and α is a parameter to control the deformation, where we put α as 0.01 in our implementation. Note that we have to recompute the three variables (i.e., V_c, $\vec{N}$, and r) for the deformed tiny fragment in each iteration, and the iterative process terminates when the tiny fragment vanishes in the related voxel, see Figure 4(b&d).

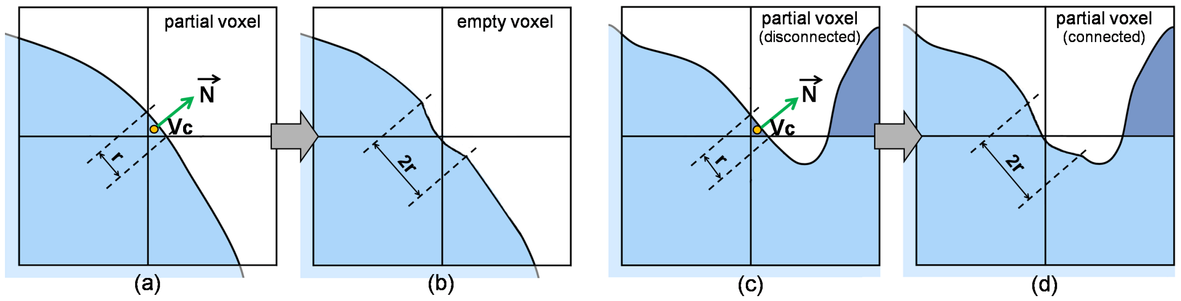

Figure 4: Deform the model's local geometry (a&b) to remove a partial voxel with a tiny fragment, and (c&d) to resolve a partial voxel with disconnected local shape.

Internal and Boundary Voxels. Partial voxels mostly cannot provide sufficient blocking strength for enforcing interlocking. Hence, one key strategy to achieve parts interlocking is by *considering full voxels to produce interlocking parts and later attaching partial voxels to the parts.* However, for most 3D models, the number of full voxels could be rather small, thus restricting the construction of interlocking parts, especially when the voxel grid has low resolution.

To include more voxels for achieving the interlocking, we also take some partial voxels into account based on the following constraints: i) large local shape volume; ii) no voxel face with tiny contacting area; and iii) has at least one full voxel as its neighbor. We call such partial voxels and the full voxels the *internal voxels*, which form the *internal volume*, and we call the remaining partial voxels outside the internal volume the *boundary voxels*, see Figure 5.

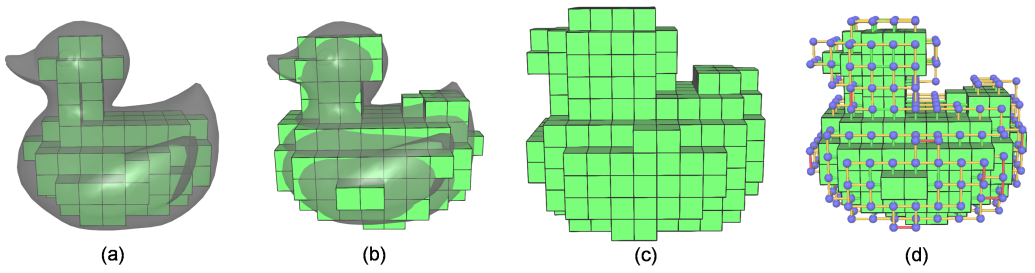

Figure 5: Voxelizing Duck. (a) Full voxels in the model; (b) internal voxels identified; (c) voxelized model; and (d) shape connection graph for the boundary voxels.

Shape Connection Graph. For the boundary voxels, we build shape connection graph to describe their structural connection strength with neighboring voxels, see Figure 5(d) and Figure 6(b) for examples. Such connection can be identified as none, weak, normal, or strong depending on the normalized contacting face area we computed earlier (in our experiments, the values are set to be 0, (0, 0.05], (0.05, 0.25], [0.25, 1.0] respectively). The shape connection graph will later help to guide the attachment of boundary voxels to initial interlocking parts, aiming to avoid the creation of disconnected or fragile 3D parts.

Saliency Connection Graph. In addition, we build saliency connection graph for boundary voxels to later avoid putting cutting seams (between interlocking 3D parts) on salient object features. In detail, we first apply [32] to compute the saliency value at every mesh vertex (see Figure 7(a)), and then estimate the saliency value of each boundary voxel by averaging the saliency values of all mesh vertices it contains. The

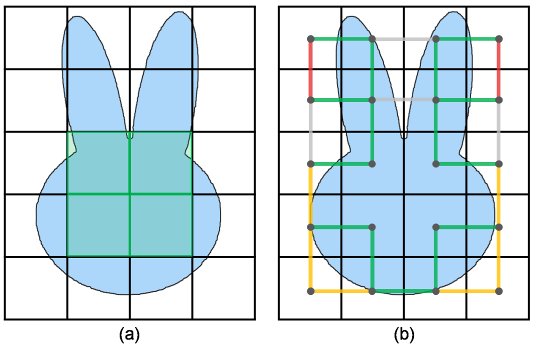
Figure 6: (a) Internal (green) and boundary voxels; (b) shape connection graph for the boundary voxels, where none, weak, normal, and strong connections are indicated in gray, red, yellow, and green, respectively.

saliency connection strength between two neighboring boundary voxels is defined as the product of the two voxels' saliency values, since we have a strong saliency connection only when the two related voxels have large saliency values, see Figure 7(b) for an example.

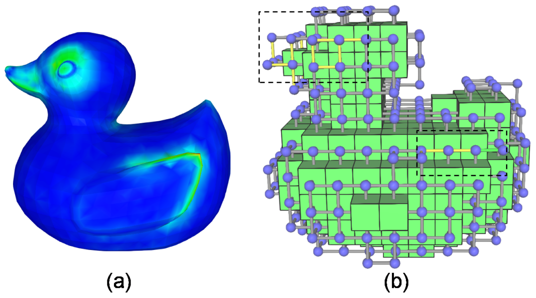
Figure 7: (a) The saliency map on Duck; (b) the saliency connection graph with the same color coding as Figure 6(b).

5. Generating 3D Interlocking Parts

Given the various inputs we prepared in Section 4, there are four major steps in generating 3D interlocking parts:

Step 1: Generating Initial 3D Interlocking Parts. First, taking the voxels in the internal volume as input, we employ [5] to construct an initial set of interlocking 3D parts, see Figure 2(c). Here we give a short description of the randomized algorithm in [5], which takes a general voxelized shape as input and iteratively extracts puzzle pieces from it to create interlocking puzzles. A formal model is proposed in [5] to ensure global interlocking of all extracted puzzle pieces by enforcing local interlocking requirement on every three consecutive intermediate puzzle pieces. To achieve this local interlocking requirement, the geometry of each puzzle piece is constructed based on a family of strategies that manipulate voxel blocking mechanics.

Here we revise [5] in two aspects:

- *General Interlocking.* We generalize the formal model in [5] to support the creation of both general interlocking and recursive interlocking structures such that we can generate interlocking parts from a given 3D model more flexibly. That is, the extracted 3D parts are general interlocking (recursive interlocking) if we enforce local interlocking requirement on every $K > 3$ ($K = 3$) consecutive intermediate parts when constructing them.
- *Parts' Size.* To meet the printable parts requirement, the size of each 3D part should not exceed the 3D printer's working volume. Hence, when constructing a 3D part, we keep updating its bounding box when it grows according to the procedure in [5], and we attach a new voxel to it only if its bounding box is still contained within the 3D printer's working volume.

By the above considerations, we can enforce the 3D interlocking and printable parts requirements for the initial 3D parts. Moreover, since the internal voxels involved here do not have tiny/thin fragments, and they have large local shape volume to support 3D parts blocking, they can help to fulfill the structural soundness and strong connection requirements for the initial 3D parts.

Step 2: Attaching Boundary Voxels to Initial 3D Parts. To assemble back the given 3D object, all boundary voxels have to be attached to the initial 3D parts while maintaining the various fabrication requirements. Note that attaching voxels to the initial 3D parts could strengthen the parts blocking, so the strong connection requirement can always be maintained.

We develop an iterative method to attach boundary voxels one by one to the initial 3D parts, see Algorithm 1 for the details. Note that for a given set of n interlocking 3D parts, we assume the disassembly order to be P_1, P_2, ..., until P_n, with P_1 as the key part that locks the entire assembly structure. For a given boundary voxel, we try to find one of its neighboring parts to attach the voxel, without violating the fabrication requirements. Here we employ the shape connection graph to ensure a boundary voxel is attached to a 3D part only when they share strong structural connection (i.e., large contacting area).

Algorithm 1: Attaching boundary voxels to initial 3D parts.

```
Input:   Initial interlocking 3D parts;
         All the boundary voxels.
Output:  Flag to indicate if the attachment succeeds;
         Modified parts with boundary voxels attached.

Put all boundary voxels into a set $S$ ;
while $S$ is non-empty do
    Initialize number of attached boundary voxel $m = 0$ ;
    for each boundary voxel $V_i$ in $S$ do
        Find all the neighboring parts of $V_i$ ;
        for each neighboring part $P_k$ do
            if there is no or weak shape connection between $V_i$ and $P_k$ (using the shape
            connection graph) then
                continue ;
            end
            if attaching $V_i$ to $P_k$ violates the dimension constraint of $P_k$ then
                continue ;
            end
            if attaching $V_i$ to $P_k$ blocks the movement of any piece from $P_1$ to $P_k$ then
                continue ;
            end
            Push $P_k$ into a candidate part vector $S_P$;
        end
        if $S_P$ is non-empty then
            Select a part $P_s$ in $S_P$ which has the largest contacting area with $V_i$ ;
            Attach $V_i$ to $P_s$ ;
            Remove $V_i$ from $S$ ;
            $m$++;
        end
    end
    if $m$ is equal to 0 then
        Return false ;
    end
end
Return true ;
```

In Algorithm 1, we need to perform the voxel attachment (i.e., the 1st *for loop*) for several iterations (i.e., the *while loop*) such that the boundary voxels that are failed to be attached in current iteration could be attached in the next iteration. This is because a boundary voxel can have more choices of parts to be attached to after its neighboring boundary voxels have been assigned to certain parts. Due to the many constraints on the voxel attachment, our algorithm does not guarantee all boundary voxels can be attached, especially for 3D models with complex shapes such as the Budda in Figure 9. In such case, we could regenerate initial 3D parts and then repeat Algorithm 1.

Step 3: Cutting Seam Refinement. We further enforce the aesthetics requirement by reassigning voxels among neighboring parts to avoid putting cutting seams across salient object features. We first identify pairs of neighboring boundary voxels that have strong saliency connection (using the saliency connection graph) but belong to different 3D parts. Then, we try to put each boundary voxel pair into the same 3D part by searching for possible ways to swap voxels among different 3D parts within the neighborhood. During such voxel reassignment, we have to maintain the interlocking requirement for the modified 3D parts, as well as avoid structural issue. Figure 8(b&c) shows an example of refining cutting seams by reassigning voxels as guided by the mesh saliency (see Figure 8(a&d)).

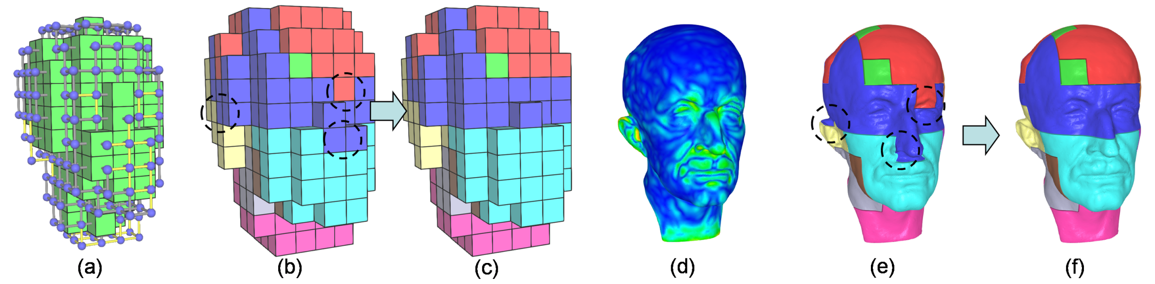

Figure 8: (a) Saliency connection graph for MAXPLANCK; (b&c) voxelized 3D parts before and after cutting seam refinement; (d) saliency map on MAXPLANCK; (e&f) CSG-refined 3D parts without and with cutting seam refinement. Note that the reassigned voxels and undesirable cutting seams are marked by black circles in (b) and (e).

Step 4: Parts Surface Refinement. To account for the machine fabrication tolerance, there should be sufficient empty space in-between 3D parts. Hence, we create the empty spacing by thinning the voxels on each part according to the voxels' local connectivity with their neighbors. The final parts' geometry can then be obtained by CSG intersection between each voxelized part and the original 3D model, see Figure 8. Comparing Figure 8(e&f), we can see that our voxel-reassigning strategy can help avoid undesirable cutting seams on the assembled object surface.

6. Results

Our method enables us to create 3D interlocking parts from object models of various shapes, see Figure 9. All the generated 3D parts are well-connected by interlocking upon the parts assembly, see Figure 10 for three more examples.

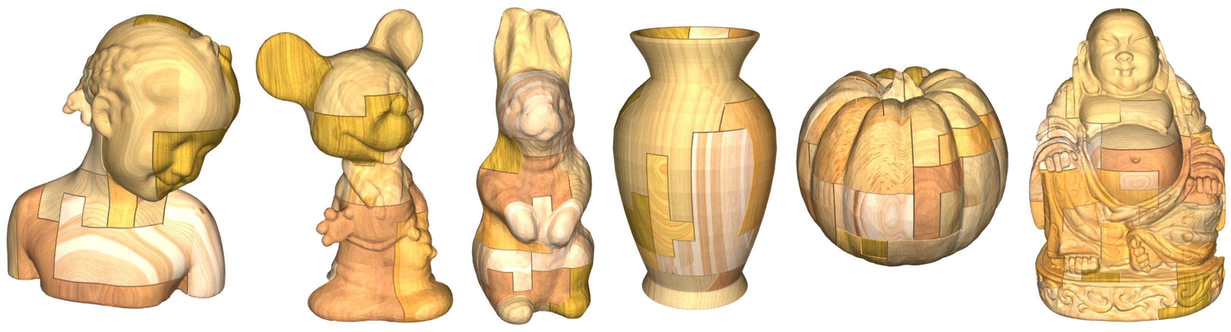

Figure 9: Various 3D interlocking models created by our approach: 8-parts BIMBA, 10-parts MICKEY, 16-parts RABBIT, 10-parts VASE, 16-parts PUMPKIN, and 20-parts BUDDHA (from left to right).

Implementation and Performance. We implemented our method in C++ and ran our experiments on a desktop computer with a 3.4GHz CPU and 8GB memory. For a given 3D model, users can specify the number of parts to be generated. Although our method allows the creation of a large number of 3D parts from a given model, we suggest partitioning the model into around 10 parts since a larger number of parts require more time for 3D printing and physical assembly. In some situations, we disable certain fabrication requirements to speed up the procedure of generating parts: i) we can disable the printable parts requirement for small objects, e.g., the ring in Figure 13; and ii) we can disable the aesthetics requirement for object without salient features, e.g., the sphere in Figure 10.

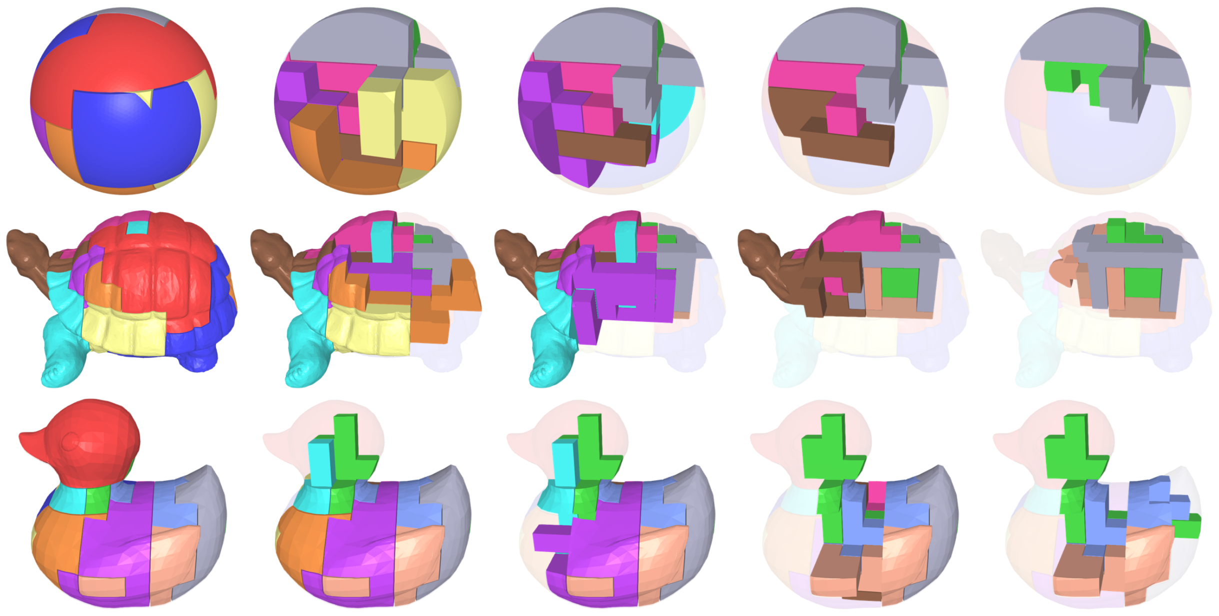
Figure 10: Assembly of a 10-parts BALL (top), a 12-parts TURTLE (middle), and a 12-parts DUCK (bottom).

Table 1 presents the performance of our method. From the timing statistics shown on the 5th column from the left, we can see that our method can generate interlocking parts fairly efficiently. Note that such time performance depends on several factors: the number of 3D parts, the complexity of the model's shape, the resolution of the input mesh, and the resolution of the voxelization. Columns 6 to 8 in the table estimate how much printing volume could be reduced after partitioning an input model into six to twenty parts. Such performance depends not only on the number of 3D parts but also on the shape complexity, since complex shapes impose stronger constraints on the parts structure and aesthetics, which may conflict with the parts dimension constraint. For example, considering the 20-parts BUDDHA (see Figure 9 and Table 1), the printing volume reduction is not that obvious since there are many salient details on its surface.

	Timing Statistics				Saving Printing Volume		
	# Model Vertices	Volume Resolution	# Parts	Timing (sec.)	Max Model Length	Max Part Length	Length Ratio
Duck	3290	9x10x11	6	25	2.00	1.33	66.5%
Bimba	15653	10x11x7	8	42	2.00	1.27	63.5%
Maxplanck	49132	7x11x8	10	62	2.00	1.23	61.5%
Bunny	15002	12x12x9	12	43	2.00	1.51	75.5%
Rabbit	10139	7x16x8	16	32	2.00	0.89	44.5%
Buddha	44973	11x16x11	20	150	2.00	1.16	58.0%

Table 1: Timing statistics and printing volume reduction.

Table 2 shows the performance of our voxelization method in resolving partial voxels with tiny or disconnected fragments, as well as in considering partial voxels for creating the interlocking. Columns 3 and 4 from the left show that in the initial voxelization, there are a few voxels with disconnected fragments (i.e., disconnected voxels) and several voxels with tiny fragments (i.e., tiny voxels). By applying local deformation on the 3D model, disconnected voxels can be avoided in the voxelization (i.e, either be removed or become a connected voxel) while tiny voxels can also be successfully removed. In addition, columns 6 and 7 in the table show that our voxelization method can effectively include a large amount of partial voxels to construct a larger internal volume for creating initial 3D interlocking parts.

		Before Deformation			After Deformation			
	Volume Resolution	# Disconne-cted Voxels	# Tiny Voxels	# Total Voxels	# Full Voxels	# Internal Voxels	# Boundary Voxels	# Total Voxels
Duck	9x10x11	2	16	547	146	270	260	530
Bimba	10x11x7	3	15	425	75	206	202	408
Maxplanck	7x11x8	1	18	364	80	170	176	346
Bunny	12x12x9	4	32	536	116	224	276	500
Rabbit	7x16x8	0	17	533	123	260	256	516
Buddha	11x16x11	24	25	1096	381	665	402	1067

Table 2: Voxel statistics of our voxelization method.

Comparison with [5]. To apply [5] for 3D printing, we need to refine the interlocking polycubes generated from [5] by performing CSG intersection with the model surface. However, this straightforward solution has several problems. First, [5] does not consider the structural constraint when connecting voxels to form 3D parts (see again Figure 1), so their generated 3D parts could have tiny and/or thin shape features after the CSG surface refinement. Second, some of these 3D parts may even be disconnected, see the red piece in Figure 11(c). Third, since [5] does not consider the aesthetics constraint, it could generate a lot of distracting cutting seams across salient features on the object assembly, see Figure 11(b). Through the local shape analysis procedure, our method can avoid putting cutting seams on salient object features, see Figure 11(e), as well as avoiding the generation of tiny, thin, and disconnected shape features on the 3D parts, see Figure 11(f).

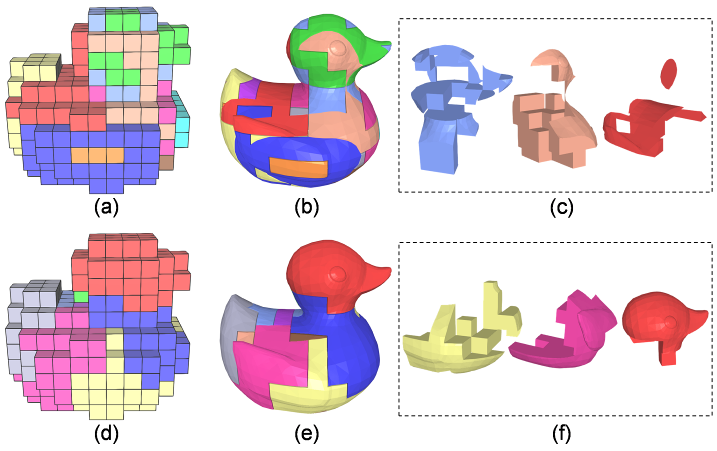

Figure 11: Duck (12 parts) created by [5] (top) and our method (bottom). (a&d) Voxelized 3D parts assembly; (b&e) CSG-refined parts assembly; and (c&f) three example 3D parts.

Printed Examples. We use 3D printers of three different types of printing technology to fabricate our interlocking 3D parts: Fused Deposition Modeling (FDM) with PLA plastic material (MakerBot Replicator 2 [36]), Selective Laser Sintering (SLS) with nylon material (Sinterstation HiQ SLS [37]), and Stereolithography (SLA) with VisiJet Proplast (ProJet 3500 HDMax [37]).

To validate the effectiveness of our method for printing a large 3D object with parts, we fabricate a 6-parts Duck model (30cm×27cm×25cm) using a DIY FDM 3D printer of working volume 25cm×25cm×45cm. Since the object is too large for the printer's working volume, we partition it into 6 interlocking 3D parts with smaller sizes (Figure 12(a)), among which the maximum length is only 20cm. Hence, we can fabricate each part by our 3D printer. To validate the connection capability by 3D interlocking, we assemble the printed parts of various models (Figure 12), and shake the parts assemblies. The parts assemblies are found to be steady and stable upon shaking, see the supplementary video.

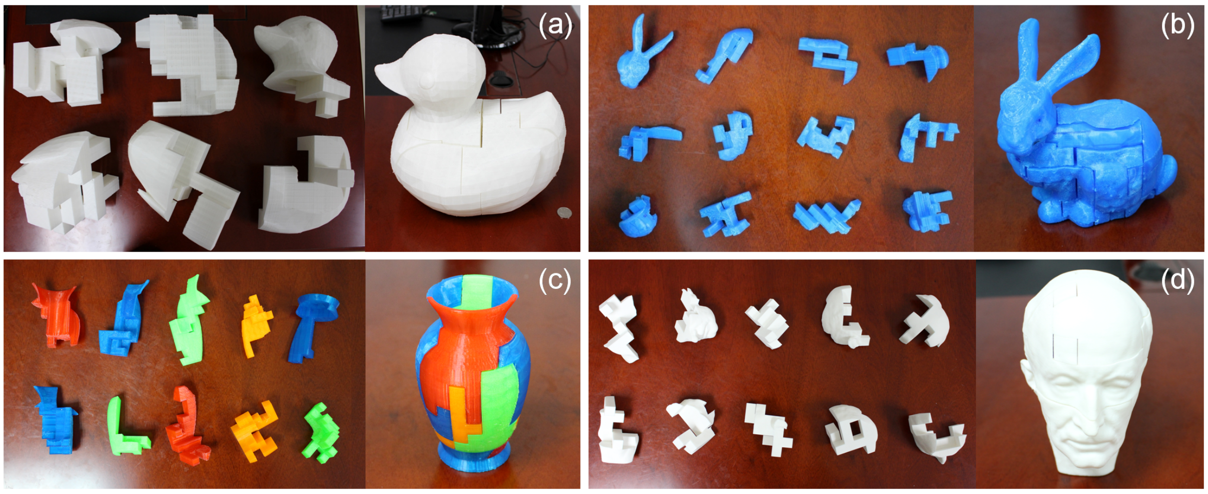

Figure 12: 3D-printed interlocking parts (left) and the corresponding assembled object (right). (a) 6-parts Duck by FDM; (b) 12-parts Bunny by FDM; (c) 10-parts Vase by FDM; and (d) 10-parts Maxplanck by SLS.

Besides the Duck, we fabricate a 12-parts Bunny (Figure 12(b)) and a 10-parts Vase with four different colors (Figure 12(c)) using the MakerBot FDM printer, from which we can see that the generated 3D parts are strong enough for printing and assembly. Note that this FDM printer can only print with a maximum resolution of 0.2mm. To fabricate models with details, we print the 10-parts Maxplanck model (Figure 12(d)) by using a 3D Systems SLS printer with 0.06mm printing resolution, so that we can achieve smoother appearance on the assembled model.

Interlocking Puzzle Ring. Lastly, we apply our method to create a 22-parts interlocking ring puzzle shown in Figure 13. It has a diameter of 3.2cm, and we fabricate its parts by a 3D Systems SLA printer with 0.025mm printing resolution. The assembled ring can be worn on a finger. We specially design its key piece: to take it out from the puzzle, we need to first lift it up for a short distance and then move it toward the ring center (see the supplementary video). Hence, when we wear this puzzle ring on our finger, our finger can block the movement of the key piece, thus preventing the ring from disassembly. Note that such interlocking puzzle could be easier to play with than those from [5] since object features on the 3D parts could give hints to the puzzle assembly.

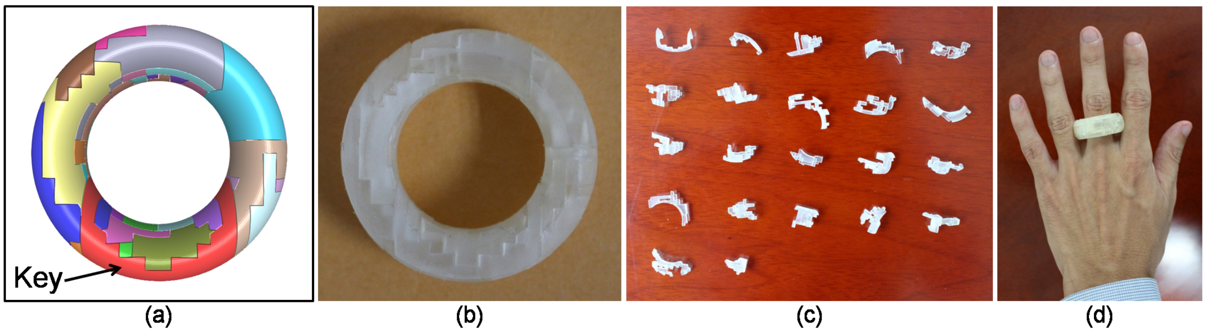

Figure 13: The 22-parts interlocking ring created by our method. (a) Virtual puzzle ring model; (b) the fabricated (assembled) ring model; (c) the twenty two 3D parts, individually printed by SLA; and (d) the ring being worn on a finger.

7. Conclusion

This paper presents a novel voxelization-based method to partition a given 3D object model of general shape into interlocking 3D parts such that the 3D object can be printed with smaller parts that are connected by 3D interlocking. We first voxelize the input 3D model and analyze the local shape within each voxel, where a local deformation strategy is developed to avoid voxel fragmentation in the voxelization. Second, we differentiate internal and boundary voxels according to the local shape contained and employ internal voxels to create initial 3D interlocking parts. Lastly, we apply the local shape information encoded by shape and saliency connection graphs to guide the construction of final parts' geometry. By these, we can generate 3D parts that are structurally sound and strongly connected by 3D interlocking, without obvious cutting seams across salient object features. We validate the performance of our method by fabricating a variety of models using different types of 3D printers, and also demonstrate its feasibility by designing and making the 22-parts interlocking puzzle ring.

Limitations. First, our method cannot generate interlocking 3D parts from a hollowed object with thin boundary since we need a solid volume to create parts blocking. Second, our method cannot control the positions of cutting seams on the input model and some of the cuts will be visible after the parts assembly. Third, the original shape of the input model could be changed slightly in the local deformation, so our current results may not be suitable for applications that require high model precision. Lastly, our shape analysis method cannot detect weak connections locally in a voxel; this issue could be circumvented by [17], or alleviated by increasing the voxel grid resolution.

Acknowledgment

The authors would like to thank Yael Friedman, a jewelry designer, for her various helpful comments on designing the 22-parts interlocking ring. The project is partially supported by the National Natural Science Foundation of China (61403357, 61222206), the Fundamental Research Funds for the Central Universities (WK0110000044), One Hundred Talent Project of the Chinese Academy of Sciences, and MOE AcRF Tier2 funding (MOE2011-T2-2-041 (ARC 5/12)), Singapore.

References

[1] L. Luo, I. Baran, S. Rusinkiewicz, W. Matusik, Chopper: Partitioning models into 3D-printable parts, ACM Tran. on Graphics (SIGGRAPH Asia) 31 (6), article 129.

[2] K.-Y. Lo, C.-W. Fu, H. Li, 3D polyomino puzzle, ACM Tran. on Graphics (SIGGRAPH Asia) 28 (5), article 157.

[3] Shapeways, Gluing 3D printed parts, `http://www.shapeways.com/tutorials/gluing_3d_printed_parts_tutorial` (2014).

[4] S.-Q. Xin, C.-F. Lai, C.-W. Fu, T.-T. Wong, Y. He, D. Cohen-Or, Making burr puzzles from 3D models, ACM Tran. on Graphics (SIGGRAPH) 30 (4), article 97.

[5] P. Song, C.-W. Fu, D. Cohen-Or, Recursive interlocking puzzles, ACM Tran. on Graphics (SIGGRAPH Asia) 31 (6), article 128.

[6] M. Skouras, B. Thomaszewski, S. Coros, B. Bickel, M. Gross, Computational design of actuated deformable characters, ACM Tran. on Graphics (SIGGRAPH) 32 (4), article 82.

[7] S. Coros, B. Thomaszewski, G. Noris, S. Sueda, M. Forberg, R. W. Sumner, W. Matusik, B. Bickel, Computational design of mechanical characters, ACM Tran. on Graphics (SIGGRAPH) 32 (4), article 83.

[8] D. Ceylan, W. Li, N. J. Mitra, M. Agrawala, M. Pauly, Designing and fabricating mechanical automata from mocap sequences, ACM Tran. on Graphics 32 (6), article 186.

[9] M. Bächer, B. Bickel, D. L. James, H. Pfister, Fabricating articulated characters from skinned meshes, ACM Tran. on Graphics (SIGGRAPH) 31 (4), article 47.

[10] J. Calì, D. A. Calian, C. Amati, R. Kleinberger, A. Steed, J. Kautz, T. Weyrich, 3D-printing of non-assembly, articulated models, ACM Tran. on Graphics (SIGGRAPH Asia) 31 (6), article 130.

[11] M. Bächer, E. Whiting, B. Bickel, O. Sorkine-Hornung, Spin-It: Optimizing moment of inertia for spinnable objects, ACM Tran. on Graphics (SIGGRAPH) 33 (4), article 96.

[12] R. Prévost, E. Whiting, S. Lefebvre, O. Sorkine-Hornung, Make It Stand: Balancing shapes for 3D fabrication, ACM Tran. on Graphics (SIGGRAPH) 32 (4), article 81.

[13] W. Wang, T. Y. Wang, Z. Yang, L. Liu, X. Tong, W. Tong, J. Deng, F. Chen, X. Liu, Cost-effective printing of 3D objects with skin-frame structures, ACM Tran. on Graphics (SIGGRAPH Asia) 32 (6), article 177.

[14] L. Lu, A. Sharf, H. Zhao, Y. Wei, Q. Fan, X. Chen, Y. Savoye, C. Tu, D. Cohen-Or, B. Chen, Build-to-Last: Strength to weight 3D printed objects, ACM Tran. on Graphics (SIGGRAPH) 33 (4), article 97.

[15] A. Telea, A. Jalba, Voxel-based assessment of printability of 3D shapes, in: 10th International Conference on Mathematical morphology and its applications to image and signal processing, 2011, pp. 393–404.

[16] O. Stava, J. Vanek, B. Benes, N. Carr, R. Měch, Stress relief: Improving structural strength of 3D printable objects, ACM Tran. on Graphics (SIGGRAPH) 31 (4), article 48.

[17] Q. Zhou, J. Panetta, D. Zorin, Worst-case structural analysis, ACM Tran. on Graphics (SIGGRAPH) 32 (4), article 137.

[18] N. Umetani, R. Schmidt, Cross-sectional structural analysis for 3D printing optimization, in: SIGGRAPH Asia Technical Brief, 2013, article 5.

[19] M. Lau, A. Ohgawara, J. Mitani, T. Igarashi, Converting 3D furniture models to fabricatable parts and connectors, ACM Tran. on Graphics (SIGGRAPH) 30 (4), article 85.

[20] D. Chen, P. Sitthi-amorn, J. T. Lan, W. Matusik, Computing and fabricating multiplanar models, Computer Graphics Forum (EuroGraphics) 32 (2) (2013) 305–315.

[21] J. McCrae, K. Singh, N. J. Mitra, Slices: A shape-proxy based on planar sections, ACM Tran. on Graphics (SIGGRAPH Asia) 30 (6), article 168.

[22] K. Hildebrand, B. Bickel, M. Alexa, crdbrd: Shape fabrication by sliding planar slices, Computer Graphics Forum (Eurographics) 31 (2) (2012) 583–592.

[23] Y. Schwartzburg, M. Pauly, Fabrication-aware design with intersecting planar pieces, Computer Graphics Forum (Eurographics) 32 (2) (2013) 317–326.

[24] H. Pottmann, Y. Liu, J. Wallner, A. Bobenko, W. Wang, Geometry of multi-layer freeform structures for architecture, ACM Tran. on Graphics (SIGGRAPH) 26 (3), article 65.

[25] Y. Zhou, S. Sueda, W. Matusik, A. Shamir, Boxelization: Folding 3D objects into boxes, ACM Tran. on Graphics (SIGGRAPH) 33 (4), article 71.

[26] H. Medellín, T. Lim, J. Corney, J. M. Ritchie, J. B. C. Davies, Automatic subdivision and refinement of large components for rapid prototyping production, Journal of Computing and Information Science in Engineering 7 (3) (2007) 249–258.

[27] J. Hao, L. Fang, R. E. Williams, An efficient curvature-based partitioning of large-scale stl models, Rapid Prototyping Journal 17 (2) (2011) 116–127.

[28] K. Hildebrand, B. Bickel, M. Alexa, Orthogonal slicing for additive manufacturing, Computers & Graphics (SMI) 37 (6) (2013) 669–675.

[29] J. Vanek, J. A. G. Galicia, B. Benes, R. Měch, N. Carr, O. Stava, G. S. Miller, PackMerger: A 3D print volume optimizer, Computer Graphics Forum 33 (6) (2014) 322–332.

[30] S. T. Coffin, The Puzzling World of Polyhedral Dissections, Oxford University Press, 1990.

[31] M. Attene, Direct repair of self-intersecting meshes, Graphical Models 76 (6) (2014) 658–668.

[32] C. H. Lee, A. Varshney, D. W. Jacobs, Mesh saliency, ACM Tran. on Graphics (SIGGRAPH) 24 (3) (2005) 659–666.

[33] F. S. Nooruddin, G. Turk, Simplification and repair of polygonal models using volumetric techniques, IEEE Transactions on Visualization and Computer Graphics 9 (2) (2003) 191–205.

[34] P. Alliez, G. Ucelli, C. Gotsman, M. Attene, Recent advances in remeshing of surfaces, Shape Analysis and Structuring, Mathematics and Visualization (2008) 53–82.

[35] M. Botsch, L. Kobbelt, M. Pauly, P. Alliez, B. Lévy, Polygon Mesh Processing, AK Peters/CRC Press, 2010.

[36] Makerbot, `http://www.makerbot.com/` (2014).

[37] 3D Systems, `http://www.3dsystems.com/` (2014).