Contents lists available at ScienceDirect

Computer-Aided Design

journal homepage: www.elsevier.com/locate/cad

Support-free hollowing for 3D printing via Voronoi diagram of ellipses[☆]

Mokwon Lee [a,1], Qing Fang [b,1], Youngsong Cho [c], Joonghyun Ryu [c], Ligang Liu [b,*], Deok-Soo Kim [a,c,**]

[a] School of Mechanical Engineering, Hanyang University, Republic of Korea
[b] School of Mathematical Science, University of Science and Technology of China, China
[c] Voronoi Diagram Research Center, Hanyang University, Republic of Korea

A R T I C L E   I N F O

Article history:
Received 16 August 2017
Accepted 16 March 2018

Keywords:
Additive manufacturing
Ellipse packing
Voronoi diagram of polygon
Voronoi diagram of disks
Printing overhang
Voronoi diagram machine

A B S T R A C T

3D printing, also called additive manufacturing, has been increasingly popular and printing efficiency has become more critical. To print artifacts faster with less material, thus leading to lighter and cheaper printed products, various types of void structures have been designed and engineered inside of shape models. In this paper, we present a novel method for generating support-free elliptic hollowing for 3D shapes which can entirely avoid additional supporting structures. To achieve this, we perform the ellipse hollowing in one of the cross sectional polygons and then extrude the hollowed ellipses to the other parallel cross sections. To efficiently pack the ellipses in the polygon, we construct the Voronoi diagram of ellipses to reason the free-space around the ellipses and other geometric features by taking advantage of the available algorithm for the efficient and robust construction of the Voronoi diagram of circles. We demonstrate the effectiveness and feasibility of our proposed method by designing and printing support-free hollow for various 3D shapes using `Poretron`, the program which computes the hollow by embedding appropriate APIs of the Voronoi Diagram Machine library that is freely available from Voronoi Diagram Research Center. It takes a 3D mesh model and produces an STL file which can be either fed into a 3D printer or postprocessed.

1. Introduction

Additive manufacturing (AM), also called 3D printing, has been advanced rapidly to make customized 3D models. Comparing to traditional manufacturing techniques, it offers enormous geometrical freedom for designers to create highly optimized components with various functionality.

Model hollowing is a typical practice for purposes of reducing printing material and time in 3D printing of light-weighted artifacts and various methods on generating optimized interior have been developed during the last few years [1–3]. The mainstream of 3D printing technologies, such as Fused Deposition Modeling (FDM) and Stereolithography (SLA), requires additional supporting structures to avoid the falling of relatively large overhanging parts during the printing process [4–6]. Generally the extra supporting structures have to be removed either manually or by dissolving dissolvable material from the printed objects.

However, there is no way at all to remove the supporting material inside interior voids of a printed object. A naïve method is to first decompose the model into a few subparts, print them and remove the supporting material individually, and then glue them together. Obviously it may largely affect the computed physical properties. Recently, there are quite a few attempts to create support-free interior voids or structures by constraining boundary slopes [7–9].

A noticeable work adopts rhombic cell structures, where the slope angles of all rhombic cells are smaller than a prescribed maximum overhang-angle, as an infill pattern to generate support-free interior voids inside the objects [8]. However, the rhombic cells have only C^0 boundaries, which suffers serious problem of stress concentration [10]. The stress around a discontinuity, e.g., a C^0 corner, will be excessively high when compared to the stresses at the other smooth areas as shown in Fig. 1. For example, hatches and doorways in airplanes are oval to stay away from being broken easily [11]. Rounded corners are structurally more beneficial than sharp corners and also reduce the probability of crack development unlike sharp corners.

[☆] This paper has been recommended for acceptance by T. Dey.
[*] Corresponding author.
[**] Corresponding author at: School of Mechanical Engineering, Hanyang University, Republic of Korea.
E-mail addresses: lgliu@ustc.edu.cn (L. Liu), dskim@hanyang.ac.kr (D.-S. Kim).
[1] Mokwon Lee and Qing Fang equally contributed.

https://doi.org/10.1016/j.cad.2018.03.007

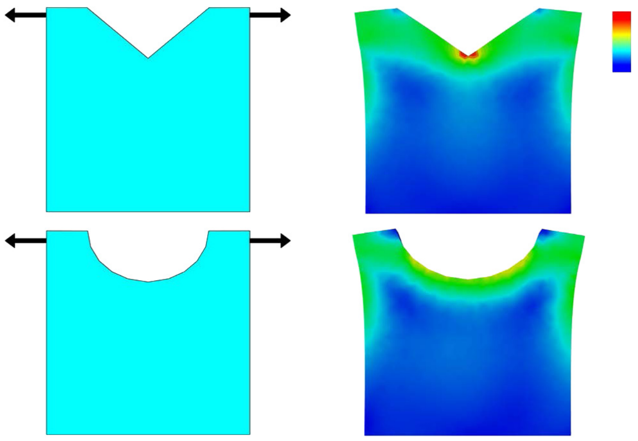


Fig. 1. Stress analysis on two thin plates by conducting two external loads (represented by two arrows respectively) on them. There is a C^0 discontinuity in the concave region of the upper shape while the concave region of the lower shape has a smooth boundary. From the stress maps, it is seen that the stress around the discontinuity is excessively high. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

Our work. We present a novel method for hollowing 3D shapes with support-free smooth elliptic interior voids. To make it simple, we first derive a class of support-free ellipses in 2D case, based on the observation on sticky property of printing material. Then we develop a novel approach for packing these support-free ellipses in the interior of 2D shapes, which is a very challenging problem. To achieve this goal, we develop a greedy but efficient algorithm on adding these ellipses successively in 2D shapes via the Voronoi diagram of ellipses in polygons using that of circular disks. Then the hollowed ellipses are extruded into 3D volume and thus a support-free hollowed 3D shape is generated. Various experimental results have demonstrated the feasibility and applicability of our proposed method. The algorithms discussed in the paper were implemented in C++ and are freely available as APIs of the Voronoi Diagram Machine (VDM) library from Voronoi Diagram Research Center (VDRC, http://voronoi.hanyang.ac.kr). Poretron, the software which implemented the proposed support-free hollowing algorithm using elliptic pores, is also freely available from VDRC: It takes 3D mesh model and produces an STL file which can be either fed into a 3D printer or postprocessed.

Contributions. Our contributions are summarized as follows.

- We develop efficient algorithms for computing the Voronoi diagram of polygon and that of ellipses within a polygon;
- We develop an efficient method for packing 2D ellipses with derivation of support-free constraint;
- We propose a method for generating support-free elliptic voids for 3D shapes.

To the best of our knowledge, our work is the first to offer a framework to generate a smooth, elliptic support-free interior voids for printing 3D shapes. This provides a practically feasible operator for support-free hollowing of 3D shapes and exemplifies research along this direction.

2. Related work

3D printing technology has drawn a lot of attention in geometric and physical modeling and optimization in computer graphics community. We discuss about the work closely related to our study and give specific discussion about their strength and limitations.

Supporting structures for 3D printing. Extra supporting structures are required to make 3D shapes printable for shapes with large overhanging parts, which leads to material waste and longer printing time. Some methods, which adopt various supporting structures such as scaffold-like structures [5] and tree-like ones [6], have been developed to generate economic usage of supporting structure for 3D models. Vanek et al. [6] search an optimal printing direction by reducing the total area of facing down regions where require additional supporting structures while Zhang et al. [12] develop a training-and-learning model to determine the optimal printing direction considering multiple factors such as contact area, viewpoint preference, and visual saliency. The work of [13] optimizes the shape of an input model to reduce the area of facing down regions for less supporting structures. However, adding supporting structure for interior voids will suffer the serious problem that it is impossible to remove these extra structures without breaking the object into pieces. Instead, we generate interior support-free voids, which completely avoid the usage of extra supports.

Interior hollowing. Significant work has been done in generating interior structure of a model to meet various geometric and/or physical properties. Stava et al. [1] hollow a 3D-printed object while maintaining its structural strength by adding some internal struts. The interior is optimized by a reduced-order parameterization of offset surfaces in [14]. Various internal structures, such as the skin-frame structure [2], the honeycomb-like structure [3], and the medial axis tree structure [12], were developed for cost-effective purposes while preserving the structural strength of printed objects. Both static balance and dynamic balance have been studied by designing the interior infills as well as changing the model shapes [15–17]. Instead of designing hollowing structure explicitly, a lot of efforts have been put on topology optimization to obtain distributions of material according to certain performance criteria during the last three decades [18,19]. However, these works have not handled the problem of avoiding large overhangs. We study this problem by developing a carving operator via support-free elliptic voids.

Support-free structure. Hu et al. [20] propose a method to decompose a 3D object into support-free pyramidal subparts. Reiner and Lefebvre [9] proposes an interactive sculpting system for designing support-free models. Recent attempts have been put on creating support-free interior structure for 3D printing. Wu et al. [8] develop a method to generate support-free infill structures on adaptive rhombic cells. A concurrent work of Langelaar [7] considers an overhang angle threshold in topology optimization and generates a support-free material distribution. However, these works only involve overhang angle and generate C^0 boundaries inevitably, resulting in large stress concentration in discontinuities. In this paper, we develop a special class of support-free ellipses and adopt them as an interior carving operator to create infill structures.

Ellipse packing. Our method of ellipse carving is quite related to the problems of ellipse and ellipsoid packing which are NP-hard. In science, the problem was prevalently approached from the packing ratio, particle size distribution, or jamming point of view to understand material properties by enforcing contacts between ellipses and using Monte Carlo method [21,22], Molecular Dynamics [23,24], local and greedy algorithms [25,26] with sampling points on the ellipse boundary [27] in either regular containers or arbitrary domains [28]. However, these methods do not balance the computation cost and packing density outcome well because research interests were primarily on the discovery of new phenomena. This is quite different from the circle packing problem which have been extensively studied from the view points of both solution quality (i.e. packing ratio) and computation time, from early study using an event-driven algorithm with a bucket acceleration [29–31] to recent ones using a systematic reasoning of empty space [32,33]. However, the infilled ellipses in our work have special constraints which make current methods infeasible to use. In this work, we develop a new method for packing ellipses in arbitrary shapes with the mathematical tool of Voronoi diagram (VD). Specifically, we perform an ellipse packing in the polygons on parallel section planes of a 3D model and extrude the ellipses of one plane to its neighboring planes in the shape volume to generate the hollowed results. To better pack the ellipses in a polygon, we construct the Voronoi diagram of ellipses to efficiently reason the free-space around the ellipses and other geometric features by taking advantage of the available algorithm for the efficient and robust construction of the Voronoi diagram of circular disks which approximate the ellipses [34]. The algorithm inherits the disk packing algorithm using the VD of disks [33].

3. Notations and overview

For the sake of simplicity, we first elaborate on our method in a 2D setting. The extension to 3D is realized by extrusion in Section 5.

3.1. Support-free ellipses

Fabrication and material parameters. Denote σ_0 as the printing precision, i.e., the thickness of each fabrication layer, which is 0.1–0.4 mm for general FDM printers. Due to the sticky property of printing material, a short length δ_0 of horizontal hangover can be successfully printed without extra supporting structure. Denote θ_0 as the maximally allowed overhang-angle. The material-dependent parameters θ_0 and δ_0 can be measured by experiments, for example, $\theta_0 = 60^\circ$ and $\delta_0 = 5$ mm for plastic PLA material. The minimum wall thickness is set as $\delta = 5\delta_0$. Note that these parameters are device and material dependent parameters, which can take different values depending on the used 3D printers and material.

Geometric characterization of support-free ellipses. Denote a and b as the horizontal axis and vertical axis of an ellipse E, respectively. A hollowed ellipse is called *support-free* if it can be printed without any extra supporting structure. Given an ellipse shown in Fig. 2(a), denote P_1 and P_2 as the two points whose tangent lines have an overhang angle of θ_0. By many experiments we have made the following observation: If the distance between P_1 and P_2 is no larger than δ_0, then this ellipse is support-free, that is, the elliptic arc between P_1 and P_2 (shown in red) can be safely printed without any extra support. On the other hand, too small interior ellipses cannot be printed due to the limit of machine precision and thus we set a lower bound of a as $5\delta_0$. Based on the observations, we have derived the conditions for a support-free ellipse as:

$$\begin{cases} b \geq a, \text{ if } 5\sigma_0 \leq a \leq \dfrac{\delta_0}{2\cos\theta_0}, \\ b \geq a\dfrac{\sqrt{4a^2-\delta_0^2}}{\delta_0\tan\theta_0}, \text{ if } a \geq \dfrac{\delta_0}{2\cos\theta_0}. \end{cases} \tag{1}$$

It is worthwhile to mention that a uniform shrinking of a support-free ellipse is still support-free as the support-free conditions in Eq. (1) are convex.

3.2. Ellipse packing method

Problem. Given a polygon $\mathcal{P}$, our goal is to find an optimal hollowing, i.e., packing, of a set of m ellipses $\mathcal{E} = \{E_1, E_2, \ldots, E_m\}$ within $\mathcal{P}$ so as to maximize the sum of the areas of all ellipses while satisfying mechanical and physicochemical constraints. In other words, we want to minimize the amount of the material to fill $\mathcal{P}$ by maximizing the area of the elliptic voids to be left unfilled. This ellipse packing problem is, however, not easier than the disk packing problem which has been known to be NP-hard [33,35]. We thus have developed a heuristic algorithm based on Voronoi diagrams.

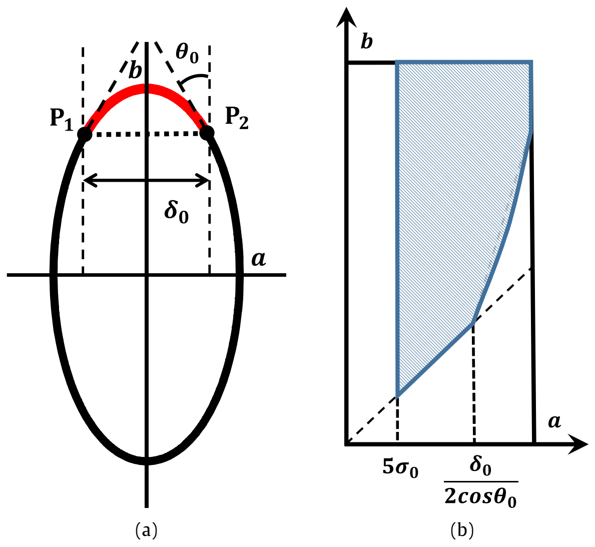


Fig. 2. (a) The condition of a support-free ellipse: $\|P_1P_2\| \leq \delta_0$ where P_1 and P_2 are the two points with tangent lines of overhang angle θ_0; (b) Ellipses with a and b within the shaded region are support-free. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2D Polygon. Given a 3D mesh $\mathcal{M}$ representing the boundary of an oriented solid, possibly with handles and interior voids, and a fabrication orientation (z-axis), we project $\mathcal{M}$ onto planes parallel to the z-axis and choose the projection direction T with the largest projected silhouette area. We then represent $\mathcal{M}$ as a sequence of parallel cross-sections, i.e., 2D polygonal shapes $\mathcal{P}^* = \{\mathcal{P}_1, \mathcal{P}_2, \ldots, \mathcal{P}_k\}$, along T. Starting from the largest cross-section, the other cross-sections are adaptively chosen according to some criteria. Specifically, the area difference between two adjacent cross-sections is kept within a threshold ξ (we set $\xi = 0.1$ in our current implementation). We choose the polygon in $\mathcal{P}^*$ with the largest area and denote it as $\mathcal{P}$ which may have internal holes (voids). Denote $\mathcal{P} = (V, E)$ where V and E are the sets of vertices and edges, thus denoted as P-vertices and P-edges where "P-" indicates their relationship with a polygon, respectively. If $\mathcal{P}$ contains holes inside, its boundary is represented as a few closed loops: One outer loop (with counter-clockwise vertices) and a few inner ones (with clockwise vertices). There can be more than one polygon on a section plane. In the following, we present our method on hollowing $\mathcal{P}$ with support-free ellipses, i.e., packing of support-free ellipses in $\mathcal{P}$.

Voronoi diagrams (VD). The Voronoi diagram (VD) of a generator set is the tessellation of space such that each cell of the tessellation consists of the locations closer to a corresponding generator than to the others and has been well-known as the most efficient and compact data structure for spatial reasoning among particles. Various types of VD can be defined by generalizing the generator type, the distance definition, and the dimension. For details, see [36].

Voronoi diagrams of disks and ellipses. In this study, we use the VDs of disks and ellipses with the Euclidean l_2-distance, particularly these VDs within a polygon $\mathcal{P}$ in 2D. See Fig. 3: (a) The VD of the interior of the polygon $\mathcal{P}$ ($\mathcal{VD}(\mathcal{P})$); (b) The VD after a disk set $\mathcal{D}$ (consisting of five disks) is inserted into $\mathcal{VD}(\mathcal{P})$ ($\mathcal{VD}(\mathcal{P}, \mathcal{D})$).

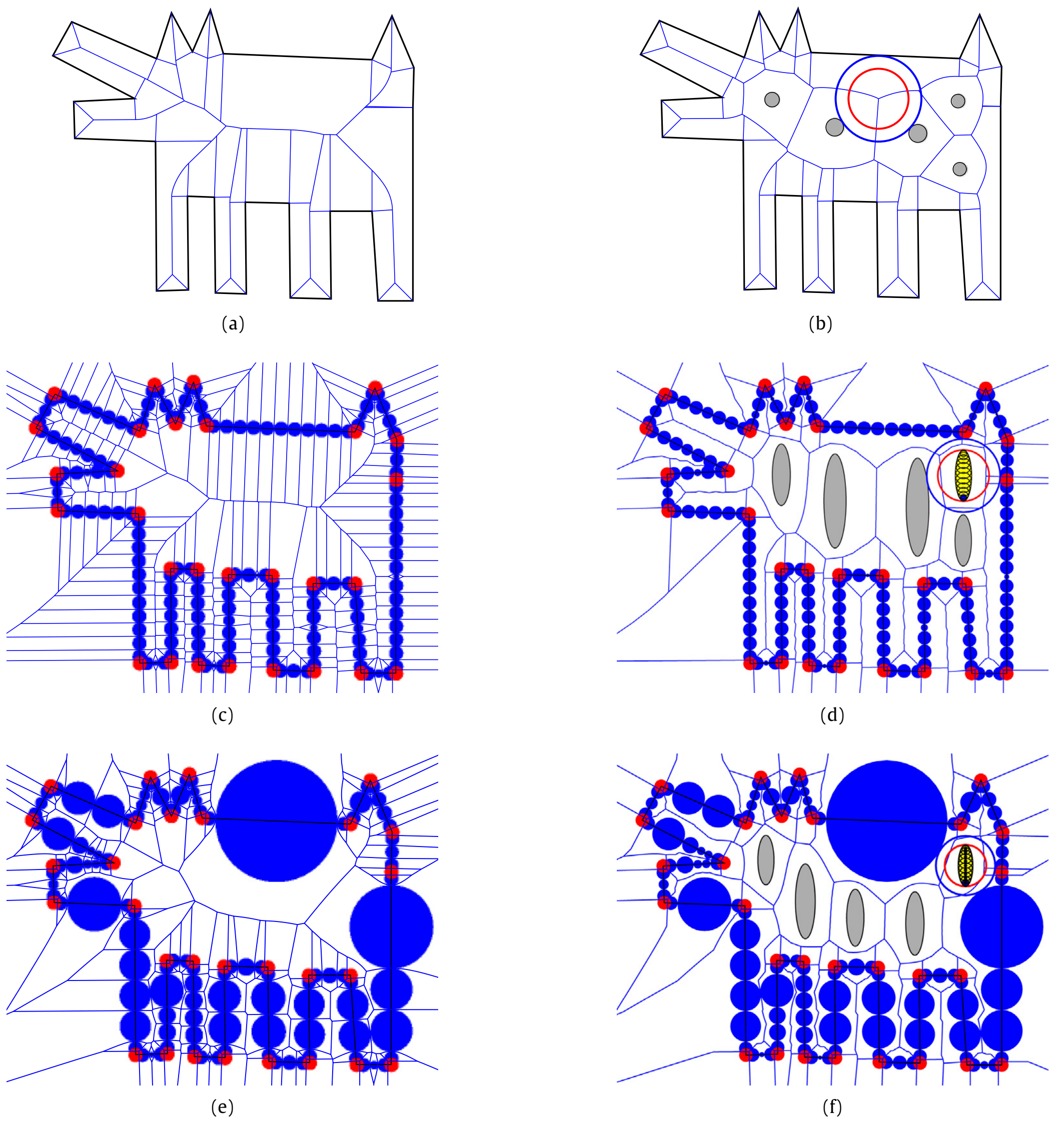


Fig. 3. Illustration of VDs in this study (a simple dog model). (a) The VD of a polygon $\mathcal{P}$ ($\mathcal{VD}(\mathcal{P})$); (b) The VD of a disk set $\mathcal{D}$ within $\mathcal{P}$ ($\mathcal{VD}(\mathcal{P},\mathcal{D})$); (c, d) The disk approximation $\tilde{\mathcal{P}}$ of $\mathcal{P}$ using "as uniform on-disks as possible (UniformOD)" method; (e, f) The disk approximation $\tilde{\mathcal{P}}$ of $\mathcal{P}$ using "as few on-disks as possible (FewerOD)" method; (c, e) The VD of the approximating disks ($\mathcal{VD}(\tilde{\mathcal{P}})$); (d, f) The VD of ellipses within $\tilde{\mathcal{P}}$ after the V-faces belonging to a P-edge are merged ($\mathcal{VD}(\tilde{\mathcal{P}},\mathcal{E})$). (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

It is known that each Voronoi vertex v is associated with a maximum empty circle, called the *clearance probe* π_v, centered at v. The radius of π_v is given by the distance from v to the boundary of its generators. The *maximum clearance probe* π_{max} is the largest clearance probe that can be defined at a Voronoi vertex of $\mathcal{VD}(\mathcal{P},\mathcal{D})$. The blue circle in Fig. 3(b) is the maximum clearance probe after the five disks are inserted into the Voronoi diagram of the polygon. The smaller red circle is 70% *shrunken probe* $\tilde{\pi}_{max}$ which is co-circular with the blue one. We emphasize that any object placed within either the maximum clearance probe or the shrunken probe is intersection-free from any other object. Note that both maximum clearance probe and shrunken probe can be found in the linear time of the number of elements of the Voronoi diagram; (c) The disk approximation $\tilde{\mathcal{P}}$ of $\mathcal{P}$ and its VD ($\mathcal{VD}(\tilde{\mathcal{P}})$); (d) The VD of ellipses $\mathcal{E}$ within the approximation $\tilde{\mathcal{P}}$ ($\mathcal{VD}(\tilde{\mathcal{P}},\mathcal{E})$).

Note that both $\mathcal{VD}(\mathcal{P})$ and $\mathcal{VD}(\mathcal{P},\mathcal{D})$ can be correctly, efficiently, and robustly computed. However, we compute $\mathcal{VD}(\tilde{\mathcal{P}},\mathcal{E})$ instead of $\mathcal{VD}(\mathcal{P},\mathcal{E})$ because of the challenges involved in the computation of the vertices and edges of Voronoi diagram $\mathcal{VD}(\mathcal{P},\mathcal{E})$, abbreviated as V-vertices and V-edges, which will be explained in detail in Section 4. Hence, we construct the approximation $\mathcal{VD}(\tilde{\mathcal{P}})$ instead of $\mathcal{VD}(\mathcal{P})$ where each P-vertex is associated with a disk called at-disk (the red filled-circles in Fig. 3(c) and (d)) and each P-edge is associated with more than two disks called on-disks (the blue filled-circles). In Fig. 3(d), we place an ellipse which inscribes the shrunken probe $\tilde{\pi}_{max}$ (instead of the maximum clearance probe π_{max}) to be conservative. The shrinking ratio is given by users according to their intention to control the overall distribution of ellipse heights. Be aware that $\tilde{\mathcal{P}}$ is associated with a disk set $\mathcal{D}^{\mathcal{P}} = \{\mathcal{D}_{at},\mathcal{D}_{on}\}$ where $\mathcal{D}_{at}$ and $\mathcal{D}_{on}$ are the sets of at-disks and on-disks, respectively.

Idea of the packing algorithm. Let $\mathcal{VD}(\tilde{\mathcal{P}})$ be the VD of the interior of $\mathcal{P}$ (Fig. 4(b)) obtained by trimming the exterior part of the entire VD in Fig. 4(a). Let $\mathcal{VD}(\tilde{\mathcal{P}},\mathcal{E})$ be the Voronoi diagram of $\mathcal{E}$ within $\mathcal{P}$. Let π_v be the clearance probe of a V-vertex v of $\mathcal{VD}(\tilde{\mathcal{P}})$ and π_{max} be the maximum clearance probe. Let v_{max} be the V-vertex corresponding to π_{max}. Starting from $\mathcal{E} = \{\emptyset\}$, we first find the

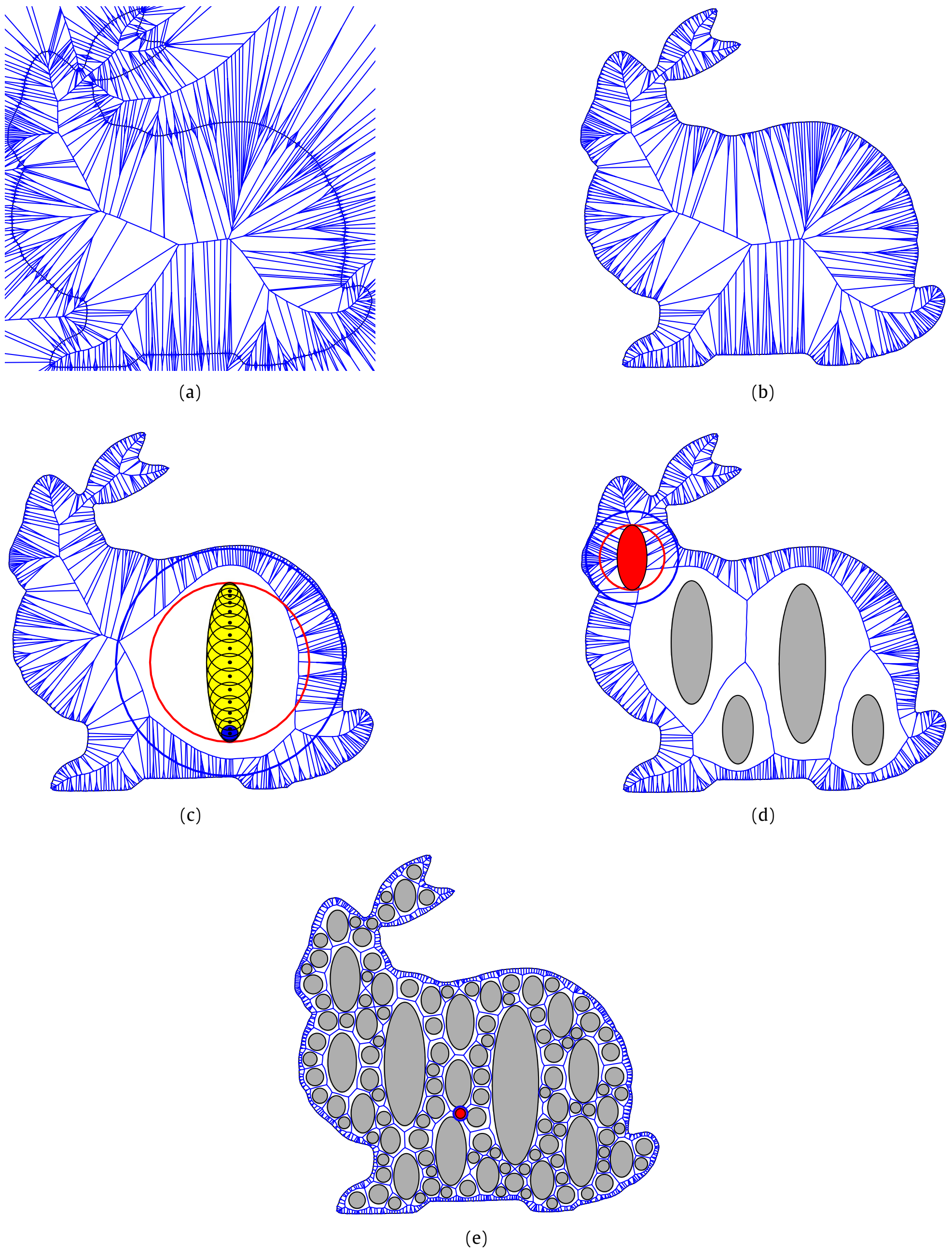


Fig. 4. The ellipse packing process for the bunny polygon using the VD of ellipses. (a) The VD of boundary disks. (b) The VD of the bunny interior. (c) The in-disks of the first ellipse within the clearance probe are incremented. (d) The fifth ellipse located within the clearance probe after the four ellipses were incremented. (e) The 100-th ellipse to be incremented. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

V-vertex v_{max} with π_{max} (the blue solid circle in Fig. 4(c)) and its shrunken probe $\tilde{\pi}_{max}$ (the red one) and place the first new ellipse E within $\tilde{\pi}_{max}$ (Fig. 4(c)). Then, we construct $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$ by inserting E into $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$. We repeat the clearance-probe-finding, the ellipse-placement, and Voronoi diagram update processes for a sufficient number of times until a termination condition is met. Fig. 4(d) shows the process after four ellipses are incremented. Fig. 4(e) shows after one hundred ellipses are incremented.

4. Ellipse hollowing via Voronoi diagram

Problem. There are two major computational phases for using VD in the ellipse hollowing: First, the construction of $\mathcal{VD}(\mathcal{P})$ and second, the construction of $\mathcal{VD}(\mathcal{P}, \mathcal{E})$.

4.1. Voronoi diagram of a polygon

Challenges. It is well-known that the algorithm for an efficient and robust construction of $\mathcal{VD}(\mathcal{P})$ is *not* trivial due to the influence of numerical error on maintaining correct topology of Voronoi diagram [36,37]. In this study we developed and implemented a new, simple, and efficient yet robust algorithm for $\mathcal{VD}(\mathcal{P})$ based on the topology-oriented approach [34,38–40]. This is because we eventually need to construct $\mathcal{VD}(\mathcal{P}, \mathcal{E})$ which is lacking in existing codes such as CGAL [41,42] and VRONI [39].

4.1.1. TOI-D algorithm

The proposed algorithm takes advantage of the Voronoi diagram of circular disks [43,44], particularly the recently reported topology-oriented incremental (TOI) algorithm for computing the Voronoi of circular disks, thus abbreviated as the TOI-D algorithm, which takes $O(n^2)$ time in the worst case but $O(n)$ time on average for n disks [34]. The idea is to approximate target geometric entities using circular disks in a sufficient resolution, construct the VD of the disks using the TOI-D algorithm, and merge some V-cells. While a similar idea was used for curved objects using the ordinary Voronoi diagram of points which were sampled from curves [39,45,46], the proposed algorithm using the TOI-D algorithm is much powerful as circles can significantly reduce problem size and complexity.

4.1.2. Approximating polygon with disks

To take advantage of the available TOI-D algorithm in the Voronoi Diagram Machine (VDM) library for disks in the plane, it is necessary to represent the problem in terms of disks. We have implemented two methods to represent a polygon using a set of disks: (i) As uniform on-disks as possible (UniformOD-method) and (ii) as few on-disks as possible (FewOD-method). In the first UniformOD-method, we represent a polygon $\mathcal{P} = (V, E)$ as follows. Let $e^* \in E$ be the shortest P-edge with its length L^*. Suppose that we cover e^* with two open disks with the diameter $L^*/2$ by placing their centers on e^* and the boundary of each disk coincides either one of the two extreme points of e^*. For each of the other P-edges with the length $L \geq L^*$, we place $\lfloor 2L/L^* \rfloor$ non-overlapping open disks in a sequel on the P-edge ($\lfloor\rfloor$ denotes a floor function). The uncovered remaining segment with the length $L - \lfloor 2L/L^* \rfloor L^*/2$ on the P-edge is then covered by one, and only one, smaller open disk. We place this smaller disk in the middle of the P-edge for algorithmic simplicity. Hence, the P-edge is entirely covered by $\lfloor 2L/L^* \rfloor + 1$ mutually exclusive open disks called an on-disk (The blue ones in Fig. 3(c)). A disk d is a child of an associated P-edge e and e is the parent of d. We also place a disk of same size, called at-disk, d_v at each P-vertex v (The red ones in Fig. 3(c)): v and d_v also have a child–parent relationship. Each disk knows its parent and each parent knows its children disks via pointers. We eventually have a set $\mathcal{D}^{\mathcal{P}}$ of children disks representing $\mathcal{P}$ where no disk contains any other while two disks may intersect, $|\mathcal{D}^{\mathcal{P}}| > m_p$ where m_p is the number of P-vertices and P-edges. The computation speed of the UniformOD-method is sensitive to the shortest P-edge and can be slow.

The computational efficiency can be improved by enforcing fewer on-disks for each P-edge using the FewerOD-method (Fig. 3(e) and (f)). We initially allocate only three on-disks on each P-edge: Two near the extreme points of the P-edge with the radii identical to the at-disks for P-vertices and the third in the middle with the diameter covering the entire rest segment of the P-edge. For example, see the biggest blue on-disk in Fig. 3(e) and its two adjacent small on-disks adjacent to the two red at-disks. If the biggest on-disk intersects any other disk, either an at-disk or on-disk, we subdivide it (to avoid computational complications in the following processes) until the intersection is resolved. Note that some P-edges of the legs in Fig. 3(e) have more than three blue on-disks due to subdivisions. We implemented this subdivision by employing a bucket system to accelerate the intersection check. Note that the number of children disks of the FewerOD-method is significantly smaller than that of the UniformOD-method. This approach of using on-disks with non-uniform sizes works well for the polygons produced from fine mesh models such as bunny. We used the FewerOD-method in our implementation.

We note here that the FewerOD-method requires a precondition. For example, see the dog model in Fig. 3: The big blue on-disks prevent the placement of ellipses and thus leave an undesirable bulk material in printed artifacts as shown in Fig. 3(f). This bulk-material symptom tends to occur for engineering models with large planar facets. In fact, the symptom may occur if $L^* \gg \delta$ for the minimum wall thickness δ. In such a case, we subdivide each P-edge into multiple shorter P-edges with 2δ.

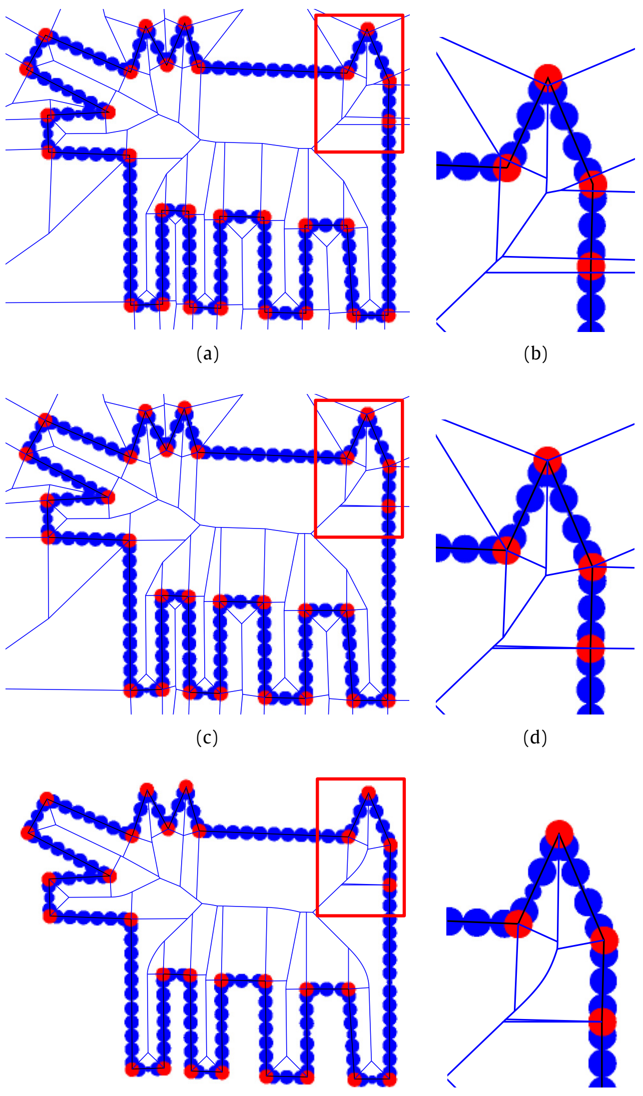


Fig. 5. Three important steps of the TOI-P algorithm for $\mathcal{VD}(\mathcal{P})$. (a, b) $\mathcal{VD}(\mathcal{D}^{\mathcal{P}})$ after the merge process. (c, d) After some V-vertices are relocated. (e, f) After outside V-edges are trimmed, the V-vertex coordinates and V-edge equations are computed. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

4.1.3. TOI-P algorithm

We construct $\mathcal{VD}(\mathcal{D}^{\mathcal{P}})$ of the disk set $\mathcal{D}^{\mathcal{P}} = \{\mathcal{D}_{at}, \mathcal{D}_{on}\}$ using the TOI-D algorithm (Fig. 3(c)) and merge the V-cells of the children on-disks of each P-edge (Fig. 5(a, b)). As shown from the figure, the resulting VD structure has a unique V-cell for both each P-edge and each P-vertex and thus its structure is close to $\mathcal{VD}(\mathcal{P})$ from both topology and geometry point of views. However, some V-vertices are off P-vertices (where V-vertices and P-vertices should coincide) (Fig. 5(b)) and these V-vertices should be moved to the related P-vertices. Fig. 5(c, d) shows the VD after relocating those V-vertices to the polygon boundary and flipping some V-edges to get the correct topology, both taking at most $O(m)$ time for m disks. Note that a V-edge flipping is required to get the correct topological structure of $\mathcal{VD}(\mathcal{P})$ (Compare Fig. 5(b) and (d)). Then, removing the exterior part of the VD and computing the V-vertex coordinates and the V-edge equations transforms the intermediate

VD structure to the correct $\mathcal{VD}(\mathcal{P})$ (Fig. 5(e, f)). Note that some previously linear V-edges are curved. Removing the at-disks and on-disks results in the VD of Fig. 3(a).

There are three cases of V-edges: (i) If a V-edge e is defined between two P-vertices, e is a line segment; (ii) Between two P-edges, e is also a line segment; (iii) Between a P-vertex and P-edge, e is a parabolic arc. As the V-edges of $\mathcal{VD}(\mathcal{P})$ are quadratic curve segments, they can be represented as a rational quadratic Bézier curve [37]. There are four cases of V-vertices: (i) Among three line segments; (ii) Among two line segments and one point; (iii) Among one line segment and two points; (iv) Among three points. Each V-vertex coordinate and V-edge equation can be correctly computed in $O(1)$ time [37].

Lemma 1. *Given $\mathcal{VD}(\mathcal{D}^P)$, TOI-P algorithm constructs $\mathcal{VD}(\mathcal{P})$ in $O(m)$ time in the worst case where m represents the number of children disks in $\mathcal{D}^P$.*

Proof. With the polygon $\mathcal{P}$ of n P-vertices and n P-edges, $n < m$, the merge process takes $O(m - n)$ time because each merge of two adjacent V-cells takes $O(1)$ time. In addition, V-vertex shift and coordinate correction for all at-disks takes $O(n)$ time. □

As m depends either on the length of the shortest P-edge or on the minimum wall thickness, the proposed TOI-P algorithm is input-sensitive. All time complexities in this paper are in the worst case sense unless otherwise stated. Note that the TOI-D algorithm for the construction of the VD of m disks takes $O(m)$ time on average and $O(m^2)$ time in the worst case [34].

4.2. Voronoi diagram of ellipses in a polygon

4.2.1. Challenges to Voronoi diagram of ellipses

Construction of $\mathcal{VD}(\mathcal{P}, \mathcal{E})$ involves the computation of V-vertices and V-edges and the topological structure among them where each of these tasks is challenging. Consider a V-vertex defined by three ellipses. It is known that there can be up to 184 complex circles that are simultaneously tangent to three conics in the plane and each corresponds to a root of a polynomial of degree 184 [47]. It is hard to expect to find the roots of a polynomial of such a high degree both exactly and efficiently. Given 10-bit precision to represent the coefficients of three random ellipses, each coefficient of the resultant necessary for the exact computation of a V-vertex v is, on average, 4603-bit integers [47]. Hence, the exact and efficient computation of the correct coordinate of v itself is hard to expect. We are not aware of any method to solve this resultant exactly and efficiently and thus an exact computation approach to construct the VD of ellipses seems impractical. The V-edge between two ellipses can be more complicated than one might expect. Even the bisector between a point and an ellipse can be very complicated [48]: It may have cusps and self-intersections and is disconnected if the point is located outside the ellipse. The bisector between two rational curves can be non-rational and even a two-dimensional object [49]. Therefore, the computation of V-vertices, V-edges, and their association through the topological structure among ellipses in a free-space is a challenge, not to mention about the VD of the ellipses within a polygon. As far as we know, no study has been reported for constructing $\mathcal{VD}(\mathcal{P}, \mathcal{E})$.

4.2.2. Idea to increment ellipses with circle approximations

We represent an ellipse E as an approximation with a set $\mathcal{D}^E$ of an odd number of disks, called in-disks which inscribes E (The yellow ones in Fig. 3(d) and Fig. 4(c)). In-disks may intersect but none is contained by another. The in-disks are generated as follows. The first in-disk d_1 is the maximal inscribing disk which is centered at the center of E. Let ϵ be an approximation error defined as the horizontal distance between ∂E and ∂d_1. Then, a point $p \in \partial d_1$ can be located for an *a priori* defined error, say ϵ_0. Hence, a second in-disk d_2 passes through p while inscribing E. We alternate this calculation up and down of d_1 to get d_2 and d_3, respectively. Repeating this calculation produces in-disks with a strictly controlled error bound ϵ_0. Given ϵ_0, a shorter ellipse has fewer in-disks than a longer one does.

Regarding $\mathcal{VD}(\tilde{\mathcal{P}})$ for $\mathcal{D}^{\mathcal{P}}$ as $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} = \{\emptyset\})$, we first find the V-vertex v_{max} with the maximum clearance probe π_{max} and place an ellipse E which inscribes the shrunken probe $\tilde{\pi}_{max}$ so that its center coincides v_{max}. We incrementally insert E at v_{max} of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ to get $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$. Instead of directly inserting E into $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$, we insert the in-disks, one by one, into $\mathcal{VD}(\mathcal{D})$ where $\mathcal{D} \equiv \mathcal{D}^{\mathcal{P}}$. As the ellipse increment process goes on, $\mathcal{D}$ contains all the in-disks of the ellipses incremented so far in addition to $\mathcal{D}^{\mathcal{P}}$.

4.2.3. TOI-EinP algorithm

The idea is very simple as follows. We insert each in-disk in $\mathcal{D}^E$ into $\mathcal{VD}(\mathcal{D})$ of existing disks and then merge the V-cells of the in-disks of $\mathcal{D}^E$. If a sufficient number of in-disks approximates each ellipse, the VD of the disks well-approximate the VD of ellipses from both topology and geometry point of views. As ellipses do not intersect, the in-disks from distinct ellipses do not intersect. In addition, the in-disks do not intersect both on-disks and at-disks on the polygon boundary, either.

The increment of an in-disk is done using the TOI-D algorithm. When we increment an in-disk, we maintain a dual representation of both $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ and $\mathcal{VD}(\mathcal{D})$. In other words, $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ and $\mathcal{VD}(\mathcal{D})$ are carefully synchronized in the following sense. We first incrementally update $\mathcal{VD}(\mathcal{D})$ until all in-disks of $\mathcal{D}^E$ (thus those of E) are exhausted to get $\mathcal{VD}(\mathcal{D} \cup \mathcal{D}^E)$. Then, we merge the V-cells of the in-disks of E to produce the topology of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$. After the merge process, each of the remaining V-vertices of $\mathcal{VD}(\mathcal{D} \cup \mathcal{D}^E)$ becomes the V-vertices of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ and a connected subset of some appropriate remaining V-edges becomes the V-edge of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$. Hence, we carefully maintain the correspondence of the V-vertices and V-edges between $\mathcal{VD}(\mathcal{D} \cup \mathcal{D}^E)$ and $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$. Note that the merge can also be done incrementally as soon as after an in-disk is incremented. Fig. 6 shows the process of incrementing ellipses (For visual convenience, we used UniformOD-method in these figures): (a) The maximum clearance probe π_{max} (the large blue circle), its shrunken probe $\tilde{\pi}_{max}$ (the red circle), the ellipse within $\tilde{\pi}_{max}$, and the biggest in-disk (blue filled circle) is incremented into the VD; (b) The second in-disk (the blue filled circle) is incremented into the VD (The previously incremented in-disk is yellow now); (c) After four in-disks are incremented; (d) After all in-disks of the first ellipse are incremented; (e) After the second ellipse is incremented; (f) After the third ellipse is incremented.

The V-vertices of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$ remaining after the V-cell merge have their coordinates inheriting from $\mathcal{VD}(\mathcal{D} \cup \mathcal{D}^E)$ which are computed from a triplet of in-disks. Thus, they are not necessarily correct for ellipses and it is necessary to compute their correct coordinates for the successful packing of next ellipse because the maximum clearance probe needs to be found from these V-vertices. The six cases of generator combination for a V-vertex in $\mathcal{VD}(\mathcal{P}, \mathcal{E} \cup \{E\})$ among ellipses, line segments (i.e. P-edges), and points (i.e. P-vertices) become a unified case of generator combination among three ellipses in $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$ because of $\mathcal{D}_{at}$ and $\mathcal{D}_{on}$. The six cases are as follows: among three ellipses, among two ellipses and one line segment, among two ellipses and one point, among one ellipse and two line segments, among one ellipse and two points, and among one ellipse, one line, and one point.

4.2.4. Geometry of the Voronoi diagram

V-vertex coordinate among three ellipses. Consider a V-vertex v defined by three ellipse generators. We iteratively find the correct location of v in $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ starting with its initial coordinate provided by $\mathcal{VD}(\mathcal{D})$. In other words, v is initially equidistant from three

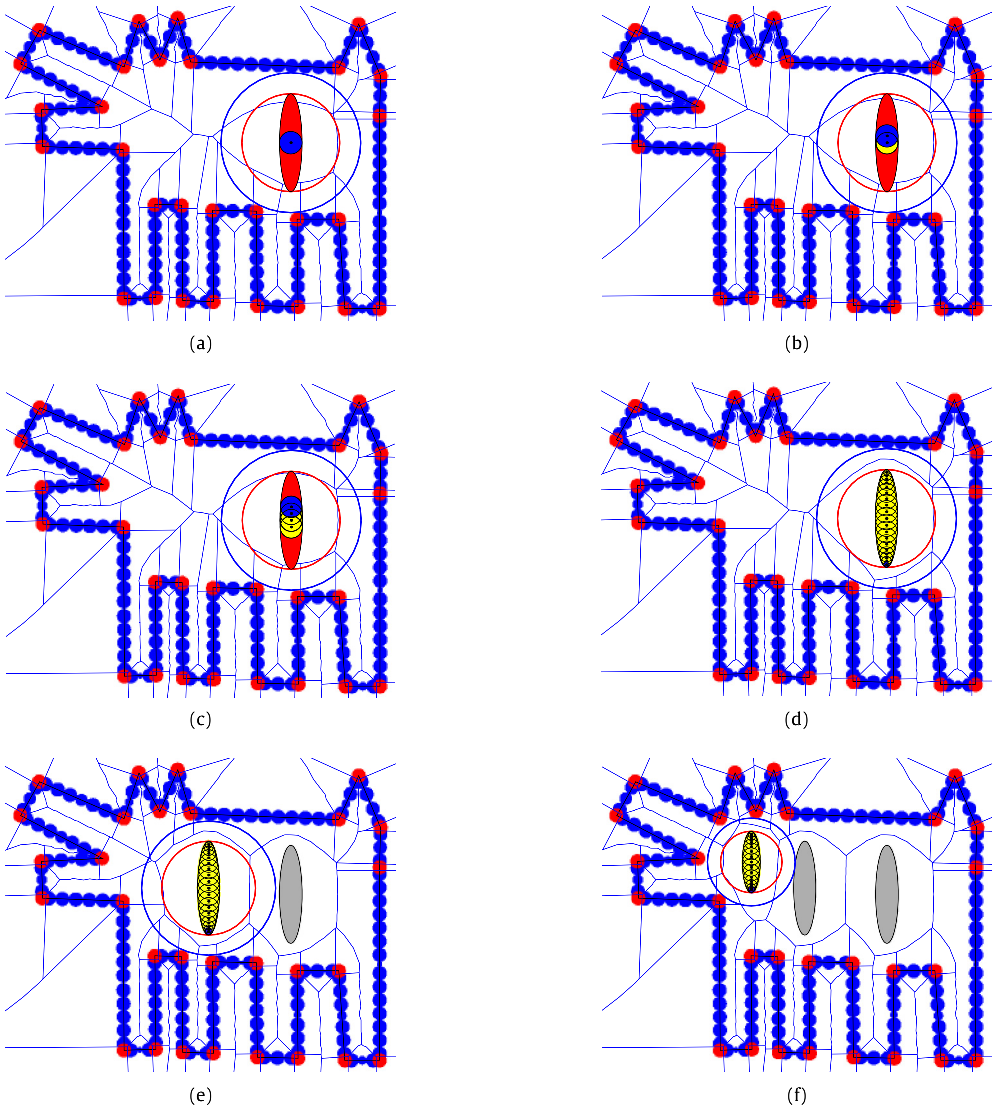


Fig. 6. The ellipse increment process of the TOI-EinP algorithm through the increments of in-disks into the VD structure. (a) Identification of the clearance probes, the ellipse, and the increment of the biggest in-disk. (b) The increment of the second in-disk. (c) The increment of the fourth in-disk. (d) After the increment of all in-disks of the first ellipse. (e) After the increment of the second ellipse. (f) The increment of the third ellipse. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

in-disks where each is a child of each of three ellipses. We project v to each of the three distinct ellipses to find its footprint (which is the closest location on an ellipse from v). Then, we compute the circumcircle, say ξ, which passes through the three footprints and use the center of ξ as the new coordinate of v. Provided that each ellipse is approximated by a sufficient number of in-disks, the iteration of this footprint-projection and circumcircle-finding process quickly converges to the correct coordinate of v due to the convexity of ellipse. Experiment shows that the initial coordinate of v is already very close to the converged coordinate. The situation that a generator(s) of v is at-disk or on-disk can be handled as an easier special case

V-edge between two ellipses. Due to the current theoretical limitations, it is practically inevitable to approximate a V-edge with a sequence of passing points computed through the envelopes of families of point/curve bisectors [48,50] or a sequence of curve segments [51]. There exist other approaches to trace bisectors for VD and medial axis transformations [52–54].

We emphasize that the topological structure of $\mathcal{VD}(\mathcal{P},\mathcal{E})$ is already known. In other words, for each V-edge e, its starting and ending V-vertices are known with correct coordinates along with its two elliptic generators. We approximate each V-edge as a sequence of points by tracing the V-edge in a way conceptually similar to the tracing algorithm of the intersection curve between two free-form surfaces [55]. Tracing V-edges in this study is, however, much simpler than tracing general intersection curve in that (i) the coordinates of two V-vertices of each V-edge are known, (ii) V-edges are planar, and (iii) each V-edge is C^1-continuous between two V-vertices. The case that e is defined between one ellipse and one at-disk (or on-disk) is an easier special case.

Finding footprints. Finding footprints is a key building block. Suppose that p is a point outside an ellipse E and L is a line passing through p. It is known that there are four, three, or two locations on E that L perpendicularly intersects E depending on whether p lies inside the evolute, lies on the evolute but not at a cusp, or lies on a cusp or outside the evolute, respectively [47]. Finding the perpendicular intersection between L and E can be formulated as a root-finding problem of a quartic polynomial thus taking $O(1)$ time. The footprint of p is obviously one of these locations which determines the minimum distance.

Lemma 2. *Suppose that the ellipses in $\mathcal{E}$ are placed inside a polygon $\mathcal{P}$ which has n P-vertices and P-edges. Suppose that there are M in-disks for $\mathcal{E}$ and the disk representation $\mathcal{D}^{\mathcal{P}}$ of $\mathcal{P}$ has N children disks. Given the synchronized $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ and $\mathcal{VD}(\mathcal{D})$, where $\mathcal{D}$ is the union of $\mathcal{D}^{\mathcal{P}}$ and the M in-disks of $\mathcal{E}$, the increment of a new ellipse E which is approximated by C in-disks takes $O(C(N+M+1)+n+|\mathcal{E}|)$ time where $|\mathcal{E}|$ represents the size of $\mathcal{E}$.*

Proof. Given a new ellipse E, with C in-disks, the increment of the C in-disks into $\mathcal{VD}(\mathcal{D})$ takes $O(C(N+M))$ time. This completes the computation of the topological structure of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$. Then, it is followed by the merges of the V-cells among the in-disks of E taking $O(C)$ time. Then, the computation of the geometry of each of $O(n+|\mathcal{E}|)$ V-vertices and V-edges takes $O(1)$ time. □

Corollary 3. *The increment of $|\mathcal{E}|$ ellipses takes $O(CN|\mathcal{E}| + C^2|\mathcal{E}|^2)$ time if $N \gg n$ and $M \gg |\mathcal{E}|$.*

Proof. The increment of one ellipse takes $O(C(N+M)+n+|\mathcal{E}|) \equiv O(C(N+M))$ time if $N \gg n$ and $M \gg |\mathcal{E}|$. □

Table 1 summarizes the three algorithms discussed above where their pseudocodes are shown in Algorithm 1, 3 and 4.

Algorithm 1: TOI-D
```
   Input: A disk set $\mathcal{D} = \{d_1, d_2, \dots, d_n\}$
   Output: The Voronoi diagram $\mathcal{VD}(\mathcal{D})$ of $\mathcal{D}$
1  Construct initial Voronoi diagram $\mathcal{VD}$.
2  for $i \leftarrow 1$ to $n$ do
3      Algorithm 2. INSERT-ONE-DISK($\mathcal{VD}, d_i$) [Insert $d_i$ into the
       current $\mathcal{VD}$]
4  return $\mathcal{VD}$
```

Algorithm 2: INSERT-ONE-DISK
```
   Input: $\mathcal{VD}(\mathcal{D})$, and new disk $d$
   Result: $\mathcal{VD}(\mathcal{D} \cup \{d\})$
1  Find the V-vertices $VV_{trim}$ and V-edges $VE_{trim}$ of $\mathcal{VD}(\mathcal{D})$ to be
   trimmed by V-cell of $d$.
2  Make new V-vertices and new V-edges which bound the
   V-cell of $d$.
3  Trim $VV_{trim}$ and $VE_{trim}$.
```

Algorithm 3: TOI-P
```
   Input: $\mathcal{VD}(\mathcal{D}^{\mathcal{P}})$ [$\mathcal{D}^{\mathcal{P}} = \{\mathcal{D}_{at}, \mathcal{D}_{on}\}$]
   Result: $\mathcal{VD}(\mathcal{P})$
1  $n \leftarrow$ the number of P-edges (or P-vertices)
2  Let $pe_i$ be the $i$th P-edge.
3  for $i \leftarrow 1$ to $n$ do
4      Collect $\mathcal{D}^i_{on} \in \mathcal{D}_{on}$ [$\mathcal{D}^i_{on}$ is on-disks of $pe_i$.].
5      Merge the V-cells of $\mathcal{D}^i_{on}$.
6  Let $pv_j$ be the $j$th P-vertex.
7  for $j \leftarrow 1$ to $n$ do
8      Relocate V-vertex $v$ which is corresponding to $\mathcal{D}^j_{at}$ [$\mathcal{D}^j_{at}$ is
       at-disk of $pv_j$.].
9      Correct topology by flipping some V-edges.
10 Compute V-vertex coordinates.
11 Compute V-edge equation.
```

4.3. Implementation issues

Criteria of adding ellipses. During the ellipse increment, the distance between two adjacent interior ellipses should be no less than the minimum wall thickness $\delta = 5\delta_0$. An intuitive scheme is to maximally pack the polygon with ellipses and shrink each ellipse by $\delta/2$. As an ellipse is not offset-invariant, the shrunk ellipse needs to be approximated but can be effectively computed.

Algorithm 4: TOI-EinP
```
   Input: $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$, and new ellipse $E$
   Result: $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$
1  $\mathcal{D}^E_{in} \leftarrow E$ [$\mathcal{D}^E_{in}$ is in-disk set of $E$]
2  $C \leftarrow$ the number of $\mathcal{D}^E_{in}$
3  for $i \leftarrow 1$ to $C$ do
4      Algorithm 2. INSERT-ONE-DISK with $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ and $d^E_i$ [$d^E_i$
       is $i$th in-disk of $E$]
5  Merge the V-cells of $\mathcal{D}^E_{in}$
6  Adjust the topological structure of $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$ Collect $v_E$
   [$v_E$ is V-vertices which bound the V-cell of $E$]
7  Compute coordinates of $v_E$
8  Collect $e_E$ [$e_E$ is V-edges which are incident to $v_E$]
9  Compute geometry of $e_E$
```

We instead use a computationally easier yet equally effective scheme as follows. We have two parameters to control the height of each ellipse: the minimum wall thickness δ and the shrink ratio $\rho \in (0, 1)$ of the maximal clearance probe π_{max}. Given π_{max}, we multiply ρ to π_{max} to get a shrunken probe $\tilde{\pi}_{max}$ with the shrunken radius γ. If $\gamma > \delta/2$, we use $\tilde{\pi}_{max}$ to produce an inscribing ellipse and increment in the VD. Otherwise, we reduce the radius of π_{max} by $\delta/2$ and produce the inscribing ellipse. The purpose of ρ is to provide users a convenient handle to control the overall distribution of ellipse heights because, in addition to the NP-hardness of the optimal ellipse packing, we never know which way is best even if we only consider geometry. Moreover, experienced users may want to have a control of overall shape distribution by tuning ρ.

Terminating condition. An ellipse should not be too small to be printed. We regard $a < \delta = 5\delta_0$ (see Eq. (1)) as the terminating condition of inserting new ellipses. In other words, whenever π_{max} is produced and shrunken, we check its horizontal axis a and terminate if it is less than δ.

5. Extrusion to 3D

5.1. 3D Hollowing

Extrusion of ellipses. The hollowed polygon $\mathcal{P}^*$ is lifted to 3D by extruding the ellipses orthogonal to the 2D plane in both directions. Denote i as the index of $\mathcal{P}^*$, i.e., $\mathcal{P}_i = \mathcal{P}^*$. For each ellipse E in $\mathcal{P}_i$, we project it onto $\mathcal{P}_{i+1}$. If E totally lies in $\mathcal{P}_{i+1}$ within a distance of minimal wall thickness σ, we keep it in $\mathcal{P}_{i+1}$. Otherwise, we shrink it with a factor so that the shrunk ellipse lies in $\mathcal{P}_{i+1}$ within a distance of σ. We can also enlarge the ellipse if there is much space around it. Note that the enlargement of the ellipse should meet the support-free condition (Eq. (1)). This operation is successively applied for the other cross sectional polygons.

Hollowing other polygons. After we complete the extrusion of all ellipses for all cross sectional polygons, we check each polygon and choose the one with the largest available region which can insert more ellipses. Then we set it as input and add more support-free ellipses in it using our method, and then extrude the newly-added ellipses to its neighborhood polygons. The above process is iteratively performed until no more ellipse can be added into the polygons.

Hollowed volume. After we obtain the hollowed polygons, we connect the corresponding ellipses on successive polygons and thus generate a hollowed volume of $\mathcal{M}$. As each polygon is support-free, the obtained hollowed 3D volume is also support-free.

Table 1
Summary table of algorithms.

	TOI-D	TOI-P	TOI-EinP
Function	Construct VD of disks	Construct VD of polygon interior	Construct VD of ellipses in polygon
Input	Disk set $\mathcal{D}$	$\mathcal{VD}(\mathcal{D}^{\mathcal{P}})$	$\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$ new ellipse E
Output	$\mathcal{VD}(\mathcal{D})$	$\mathcal{VD}(\mathcal{P})$	$\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E} \cup \{E\})$
Time complexity	Worst C.: $O(n^2)$ Avg. C.: $O(n)$	Worst C.: $O(m)$	Worst C.: $O(C(N + M + 1))$
Explanatory note	$\|\mathcal{D}\| = n$ Worst C.: Worst Case Avg. C.: Average Case	$\mathcal{P}$: Polygon $\mathcal{D}^{\mathcal{P}}$: children disks of $\mathcal{P}$ $\|\mathcal{D}^{\mathcal{P}}\| = m$	$\tilde{\mathcal{P}}$: disk representation of $\mathcal{P}$ $\mathcal{E}$: inserted ellipses N: the number of children disks of $\mathcal{P}$ M: the number of all children disks of $\mathcal{E}$ C: the number of in-disks of E

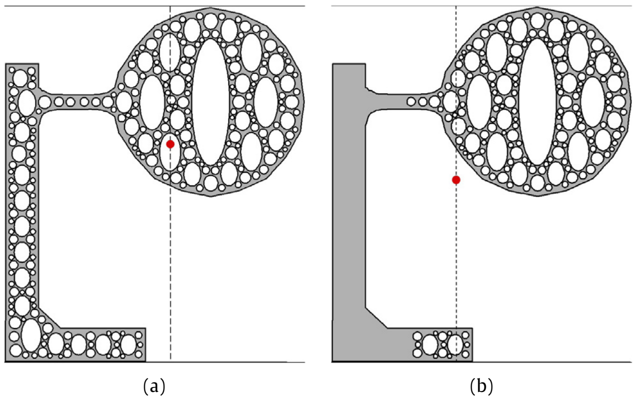


Fig. 7. Optimization of static balance (2D case). (a) The hollowed object cannot stand by itself (left). (b) It is optimized to a self-balanced object using our optimizer (right). The red dots denote the gravity center. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

5.2. Functional constraints

The printed objects are generally required to meet some functional constraints such as static balance and mechanical stiffness [2,8]. However, integration of these constraints into the VD computation is computationally expensive. Thus we handle them as a postprocess after we generate the support-free hollowed volume.

Design variables. We define a design variable $\gamma_E \in [0, 1]$, as a shrinking factor, for each ellipse E inside $\mathcal{M}$. A value $\gamma_E = 0$ means that E is totally filled with solid. The basic idea is that shrinking of ellipses can shift the center of gravity of the model and can improve its mechanical stiffness as more material is filled. Thus we can easily formulate the optimization according to a specific objective function. In particular, we discuss about the optimization with respect to static balance as an example. The optimization for other constraints can be similarly achieved.

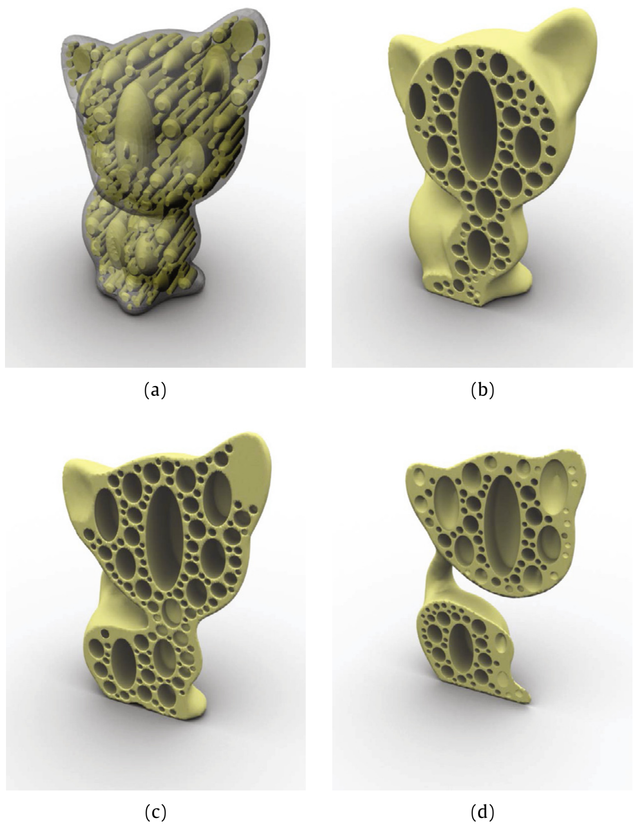


Fig. 9. A hollowed kitten model. (a) The hollowed model; (b–d) different cross-sections.

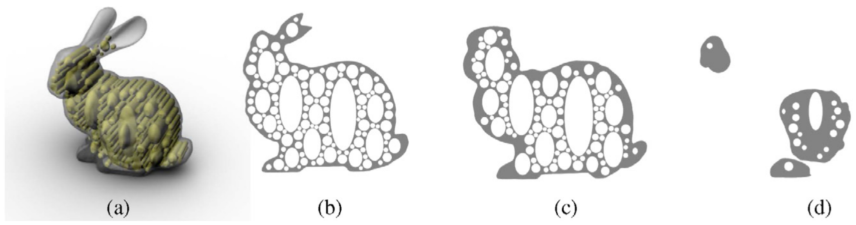


Fig. 8. A hollowed bunny model. (a) The hollowed model; (b–d) different cross-sections.

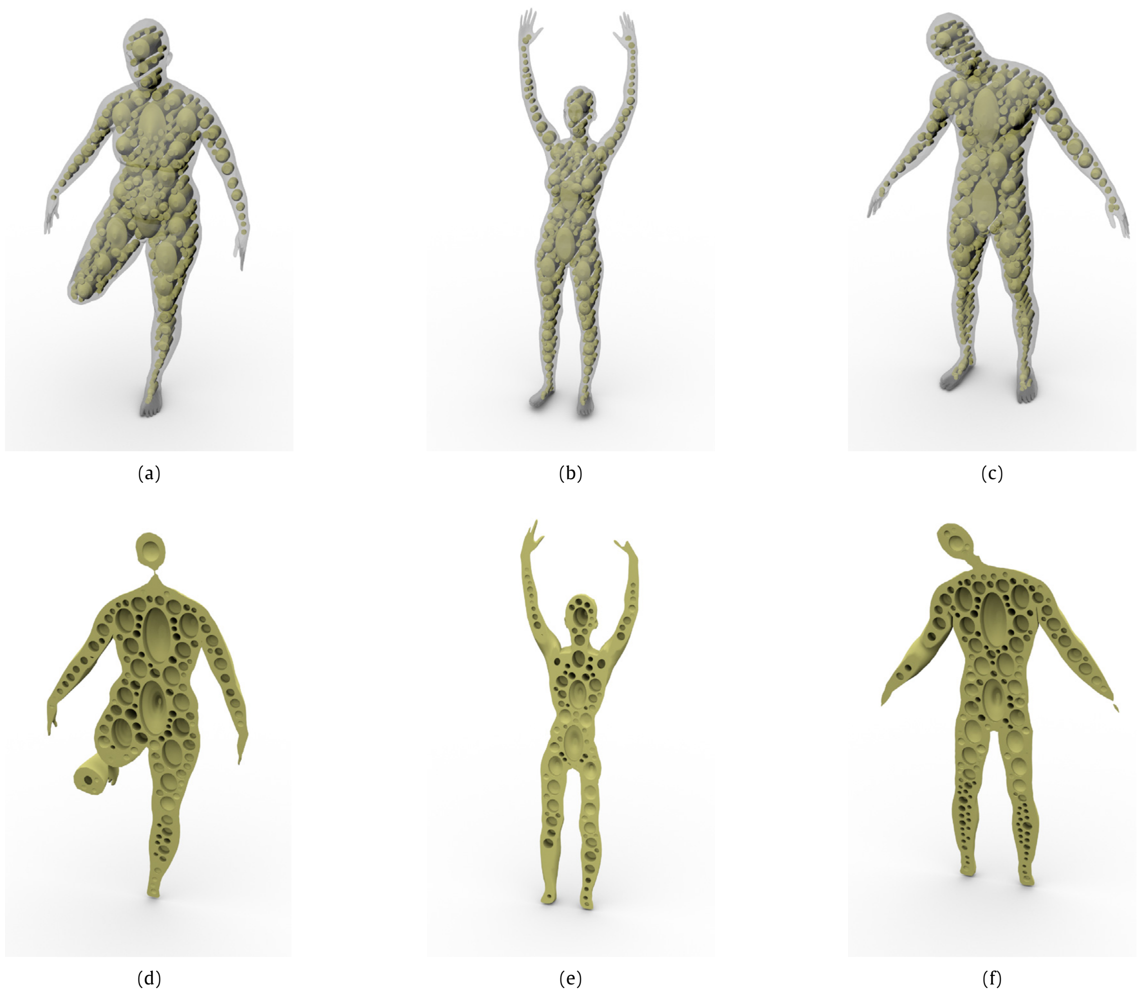


Fig. 10. Hollowed human models with different poses. The upper row shows the hollowed model and the lower row one of the cross sections.

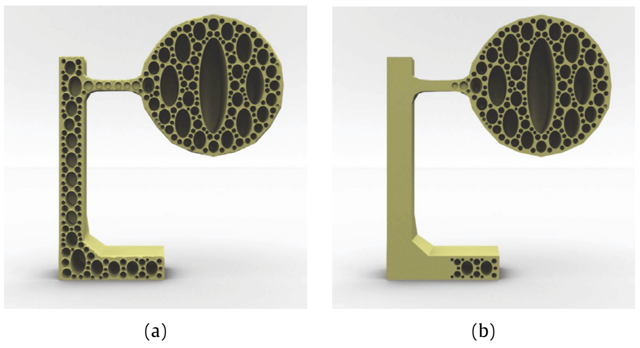


Fig. 11. (a) The hollowed hanging ball may fall down without balance optimization (left) (b) while the optimized one is standing stable.

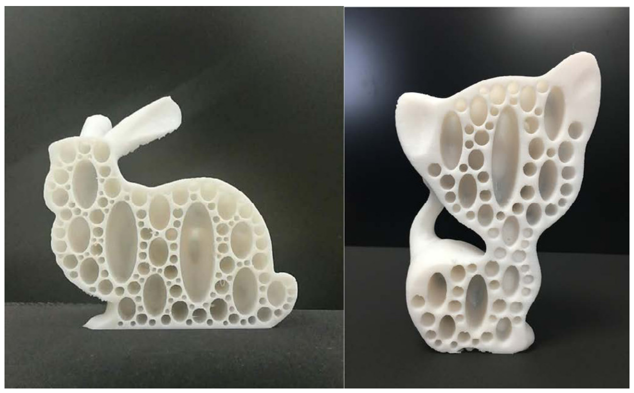
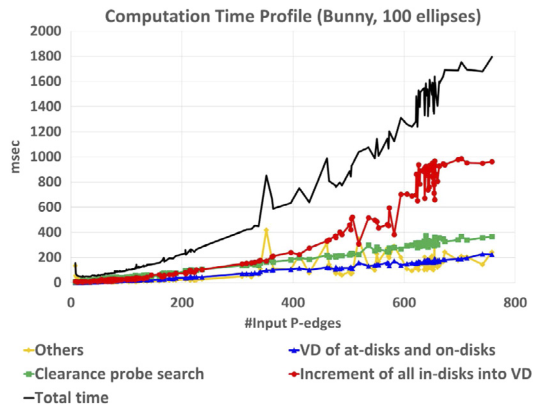


Fig. 12. Photos of the fabricated bunny model and kitten model hollowed by our method.

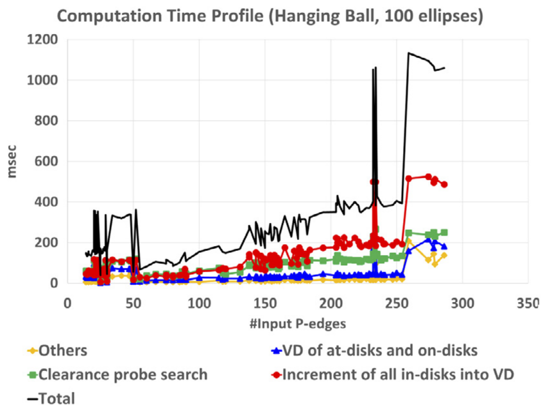
Static balance. An object is self-balanced when the vertical projection of its gravity center lies in the convex hull of its contact points with the ground. As shown in Fig. 7, it is intuitive that shrinking ellipses on the left hand side of the gravity center will shift it leftwards, i.e., closer to the convex hull of its contact points. It is easy to formulate an optimization of minimizing the horizontal distance between the gravity center and the boundary of the convex hull. Our optimizer thus tries to reach a balance by shrinking some ellipses and thus shifting the gravity center into the convex hull.

6. Experimental results

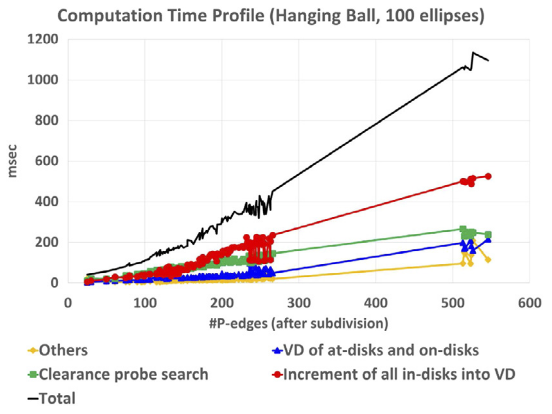
Computational platform. We have implemented our algorithm in C++ on a standard desktop PC with Intel(R) Core(Tm) i7-4790K CPU@4.0 GHz and 16 GB of RAM. Thanks to the efficient implementation of TOI-EinP, the VD generation of polygons and ellipses is fast and the ellipse hollowing takes less than 30 s for all examples.



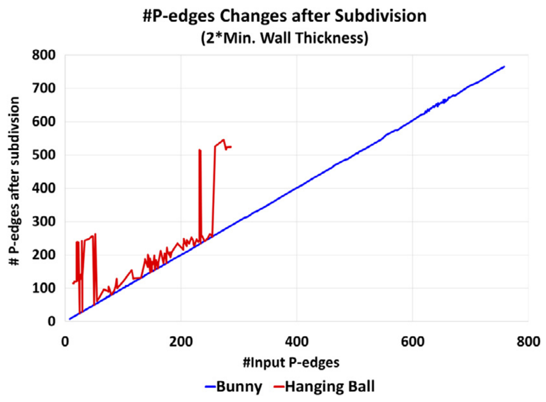
(a)



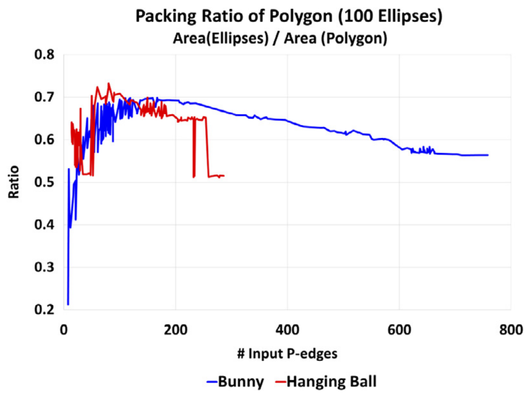
(b)



(c)



(d)



(e)

Fig. 13. Computation time profile of the TOI-EinP algorithm. (a) Bunny model (Time vs. # input P-edges). (b) Hanging ball model (Time vs. # input P-edges). (c) Hanging ball model (Time vs. # subdivided P-edges). (d) # P-edges changes of both Bunny and Hanging ball models after subdivision. (e) Packing ratio of polygons in both Bunny and Hanging ball models. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

Printer configuration and parameters. We fabricate the objects using a commercial FDM 3D printer: The Ultimaker 2+ with tray size of 223 mm × 223 mm × 205 mm. The printable layer thickness of the printer (printing precision) ranges from 0.1 mm to 0.4 mm and we use a value of $\sigma_0 = 0.2$ mm. We test the plastic PLA material used in the 3D printing and set the maximally allowed overhang-angle as $\theta_0 = 60^\circ$ and the maximal length of printable horizontal hangover as $\delta_0 = 5$ mm.

Manufacture setting. After we generate the hollowed models, we add supporting structures for the exterior part of the models but do not add any interior support. After the models are fabricated, we manually remove the exterior supports. All models have been successfully printed which reveals that the hollowed interior of the models is printed without any problem. We also validate this by printing and checking only half of the models which will be shown later.

Experiments. Fig. 8 shows an example of hollowed bunny model and a few cross-sections. Fig. 9 shows a 3D hollowed volume of kitten model with a few cross-sections. Fig. 10 shows human models of different poses.

Fig. 11 shows two hollowed hanging balls. The left one cannot stand by itself. After using our optimizer, it can be optimized to be well balanced in the right by filling material in the elliptic voids of the column. Fig. 12 shows photos of the fabricated bunny model and kitten model which are hollowed by our method. The models are successfully fabricated without adding any extra support in the interior voids.

Performance of the TOI-EinP algorithm for constructing VD. We conducted computational efficiency test using two models: the bunny and the hanging ball models. From the bunny model, we produced 121 planes resulting 219 polygons as some planes contain more than one polygon. The smallest and the largest polygons have 8 and 655 P-edges, respectively. For experimental purpose, we enforced to pack 100 ellipses into each polygon ignoring the mechanical and physicochemical constraints. Fig. 13(a) shows computation time vs. polygon size in terms of the input P-edges. The top-most black curve: the total time; The next red one: the time for incrementing all the in-disks of the ellipses into the VD structure; The next green one: that for finding the maximum clearance probe; The blue one: that for constructing the VD of the at-disks and on-disks. Note that the total time is weakly super-linear mainly due to the increment process of in-disks which is believed to be caused by the mapping mechanism of equivalent V-vertices between $\mathcal{VD}(\mathcal{D})$ and $\mathcal{VD}(\tilde{\mathcal{P}}, \mathcal{E})$. We used the map in the C++ template which is implemented by a binary search tree, taking $O(\log n)$ time for each query for n entities. With this model, we used the FewerOD method (as few on-disks as possible) using a bucket system for the acceleration.

From the hanging ball model, we produced 88 planes resulting 141 polygons: The smallest and the largest polygons have 25 and 232 P-edges, respectively. As this model consists of several large planar faces together with smaller ones, the polygons have several long P-edges together with short ones. This is common in many engineering products. Hence, we subdivided each P-edge into a set of P-edges of the length defined by the previously stated rule. Fig. 13(b) also shows computation time vs. polygon size in terms of the input P-edges. Note that the correlation is very weak compared to the bunny model as is expected because of the big variation of the P-edge lengths. Fig. 13(c) shows computation time vs. the number of subdivided P-edges: The curves are fairly well correlated in a slightly super-linear fashion. The big gap between the two clusters of data is due to many subdivided P-edges through very long input P-edges. For example, the left-most data in the right cluster in Fig. 13(c) corresponds to a polygon with 234 input P-edges which was subdivided into 513 shorter P-edges. Fig. 13(d) shows the number of subdivided P-edges vs. the number of input P-edges. The bunny model is expectedly a straight line whereas the hanging ball model shows bumpy curve which is similar to the curves in Fig. 13(b). The relationship between Fig. 13(b) and (c) can be explained by Fig. 13(d). Fig. 13(e) shows the packing ratio of the 100 ellipses in each polygon. The bunny model is expectedly smooth with decreasing pattern for bigger polygons as we incremented only 100 ellipses whereas the curve of the hanging ball model is bumpy.

Comparison to rhombic cell structure. The work of [8] adopts rhombic cell structure, which have C^0 discontinuity on boundaries, to generate support-free interior voids for 3D shapes. We compare our method with this method as shown in Fig. 14. We apply two methods on the same P model with similar hollowing ratios. Then we fix the bottom of the model and conduct an identical external load on it, respectively. From the stress map we can see that the result generated by [8] suffers the problem of stress concentration at the region marked in red, which generally happens in discontinuity. This does not happen in our method.

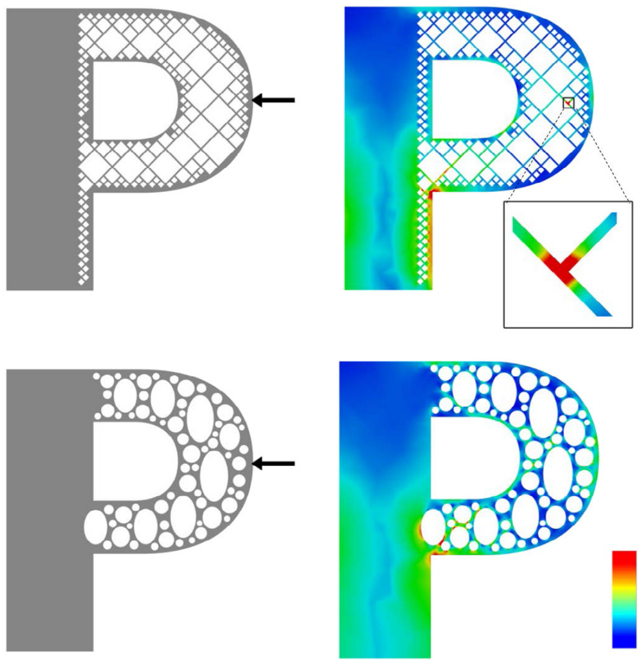


Fig. 14. Comparison with Wu et al. (2016b). Upper row: the hollowed P model using Wu et al. (2016b) with a hollowing ratio of 24.3%; Lower row: the hollowed P model using our method with a hollowing ratio of 25.7%. The bottom of the model is fixed and one identical external load is conducted on the right, respectively, as shown by the arrow. The right figures show the color maps of stress. It is seen that the region marked in red in the upper-right figure suffers the problem of stress concentration, i.e., the stress there is very high. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

7. Conclusions and future work

In this paper we propose a novel approach for generating support-free interior hollowing for general 3D shapes. The generated 3D shapes can be directly fabricated with FDM 3D printers without any usage of extra supports in interior voids. This is based on the observation of a family of support-free ellipses and is achieved by hollowing 2D shapes with these ellipses. Then the interior ellipses are extruded into volume for generating hollowed 3D shapes. We also develop a new, efficient and robust algorithm for the Voronoi diagram of polygons and the first algorithm for the Voronoi diagram of ellipses within a polygon, both based on the topology-oriented incremental approach, which are quite useful for generating the ellipse packing in 2D shapes. With the sizes of ellipses as design variables, the optimization according to a specific objective function, e.g., static stability, can be formulated. Experimental results have shown the practicability and feasibility of our proposed approach.

Limitation and future work. Our research opens many directions for future studies. First, the packing results can be further optimized by optimizing the positions and sizes of the ellipses for the purpose of increasing packing ratio. Second, it is expected to extend our approach for generating support-free ellipsoids for 3D shapes. This is feasible but needs more effort. Last but not the least, we are interested in studying general support-free shapes for additive manufacturing, which is a promising direction for geometric modeling and processing.

Acknowledgments

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIP, MSIT) (Nos. 2017R1A3B1023591, 2016K1A4A3914691).

References

[1] Stava O, Vanek J, Benes B, Carr N, Měch R. Stress relief: improving structural strength of 3D printable objects. ACM Trans Graph 2012;31(4):48:1–48:11.

[2] Wang W, Wang TY, Yang Z, Liu L, Tong X, Tong W, et al. Cost-effective printing of 3D objects with skin-frame structures. ACM Trans Graph 2013;32(6): 177:1–177:10.
[3] Lu L, Sharf A, Zhao H, Wei Y, Fan Q, Chen X, et al. Build-to-last: strength to weight 3D printed objects. ACM Trans Graph 2014;33(4):97:1–97:10.
[4] Strano G, Hao L, Everson RM, Evans KE. A new approach to the design and optimisation of support structures in additive manufacturing. Int J Adv Manuf Technol 2013;66(9):1247–54.
[5] Dumas J, Hergel J, Lefebvre S. Bridging the gap: Automated steady scaffoldings for 3D printing. ACM Trans Graph 2014;33(4):1–10.
[6] Vanek J, Galicia JAG, Benes B. Clever support: Efficient support structure generation for digital fabrication. Comput Graph Forum 2014;33(5):117–25.
[7] Langelaar M. Topology optimization of 3D self-supporting structures for additive manufacturing. Additive Manufacturing 2016;12;Part A:60–70.
[8] Wu J, Wang CCL, Zhang X, Westermann R. Self-supporting rhombic infill structures for additive manufacturing. Comput Aided Des 2016;80:32–42.
[9] Reiner T, Lefebvre S. Interactive modeling of support-free shapes for fabrication. In: EUROGRAPHICS. 2016.
[10] Pilkey WD, Pilkey DF. Peterson's stress concentration factors. third ed. Wiley; 2008. ISBN 978-0-470-04824-5.
[11] Karthikeyan K. Why hatches and doorways in ships and airplanes are oval? 2016. https://geekswipe.net/science/physics/why-hatches-and-doorways-in-ships-and-airplanes-are-oval/.
[12] Zhang X, Xia Y, Wang J, Yang Z, Tu C, Wang W. Medial axis tree-an internal supporting structure for 3D printing. Comput Aided Geom Design 2015; 35–36(5):149–62.
[13] Hu K, Jin S, Wang CC. Support slimming for single material based additive manufacturing. Comput Aided Des 2015;65:1–10.
[14] Musialski P, Auzinger T, Birsak M, Wimmer M, Kobbelt L. Reduced-Order shape optimization using offset surfaces. ACM Trans Graph 2015;34(4):102;1–9.
[15] Prévost R, Whiting E, Lefebvre S, Sorkine-Hornung O. Make it stand: balancing shapes for 3d Fabrication. ACM Trans Graph 2013;32(4):81;1–10.
[16] Christiansen AN, Schmidt R, Bærentzen JA. Automatic balancing of 3D models. Comput Aided Des 2015;58:236–41.
[17] Bächer M, Whiting E, Bickel B, Sorkine-Hornung O. Spin-it: optimizing moment of inertia for spinnable objects. ACM Trans Graph 2014;33(4): 96:1–96:10.
[18] Deaton JD, Grandhi RV. A survey of structural and multidisciplinary continuum topology optimization: post 2000. Struct Multidiscip Optim 2014;49(1):1–38.
[19] Wu J, Dick C, Westermann R. A system for high-resolution topology optimization. IEEE Trans Vis Comput Graphics 2016;22(3):1195–208.
[20] Hu R, Li H, Zhang H, Cohen-Or D. Approximate pyramidal shape decomposition. ACM Trans Graph 2014;33(6):213:1–10.
[21] Buchalter BJ, Bradley RM. Orientational order in amorphous packings of ellipses. J Phys A: Math Gen 1992;26(3):L1219–24.
[22] Schreck CF, Xu N, O'Hern CS. A comparison of jamming behavior in systems composed of dimer- and ellipse-shaped particles. Soft Matter 2010;6(13): 2960–9.
[23] Donev A, Torquato S, Stillinger FH, Connelly R. Jamming in hard sphere and disk packings. J Appl Phys 2004;95(3):989–99.
[24] Donev A, Connelly R, Stillinger FH, Torquato S. Underconstrained jammed packings of nonspherical hard particles: ellipses and ellipsoids. Phys Rev E 2007;75(5):051304.
[25] Hitti K, Bernacki M. Optimized Dropping and Rolling (ODR) method for packing of poly-disperse spheres. Appl Math Model 2013;37(8):5715–22.
[26] Birgin EG, Lobato RD, Martínez JM. Packing ellipsoids by nonlinear optimization. J Global Optim 2016;65(4):709–43.
[27] Delaney G, Weaire D, Hutzler S, Merphy S. Random packing of elliptical disks. Phil Mag Lett 2005;85(2):89–96.
[28] Lozano E, Roehl D, Celes W, Gattass M. An efficient algorithm to generate random sphere packs in arbitrary domains. Comput Math Appl 2016;71(8): 1586–601.
[29] Jodrey WS, Tory EM. Computer simulation of close random packing of equal spheres. Phys Rev A 1985;32:2347–51.
[30] Lubachevsky BD, Stillinger FH. Geometric properties of random disk packings. J Stat Phys 1990;60(5):561–83.
[31] Lubachevsky BD. How to simulate billiards and similar systems. J Comput Phys 1991;94(2):255–83.
[32] Specht E. A precise algorithm to detect voids in polydisperse circle packings. In: Proc. R. Soc. A. 2015. p. 20150421.
[33] Sugihara K, Sawai M, Sano H, Kim D-S, Kim D. Disk packing for the estimation of the size of a wire bundle. Jpn J Ind Appl Math 2004;21(3):259–78.
[34] Lee M, Sugihara K, Kim D-S. Topology-oriented incremental algorithm for the robust construction of the voronoi diagrams of disks. ACM Trans Math Software 2016;43(2):14:1–23.
[35] Zeng Z, Yu X, He K, Huang W, Fu Z. Iterated tabu search and variable neighborhood descent for packing unequal circles into a circular container. European J Oper Res 2016;250(2):615–27.
[36] Okabe A, Boots B, Sugihara K, Chiu SN. Spatial tessellations: Concepts and applications of voronoi diagrams. second ed. Chichester: John Wiley & Sons; 1999.
[37] Kim D-S, Hwang I-K, Park B-J. Representing the Voronoi diagram of a simple polygon using rational quadratic Bézier curves. Comput Aided Des 1995;27(8):605–14.
[38] Sugihara K, Iri M. Construction of the Voronoi diagram for "one million" generators in single-precision arithmetic. Proc IEEE 1992;80(9):1471–84.
[39] Held M. VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. Comput Geom 2001;18(2):95–123.
[40] Held M, Huber S. Topology-Oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments. Comput Aided Des 2009;41(5):327–38.
[41] CGAL, CGAL Library Homepage, 2016. http://www.cgal.org/.
[42] Alliez P, Delage C, Karavelas MI, Pion S, Teillaud M, Yvinec M. Delaunay tessellations and Voronoi diagrams in CGAL (Draft). In: Rien van de Weijgaert Gert Vegter JR, Icke V, editors. Tessellations in the sciences. Virtues, techniques and applications of geometric tilings. 2007.
[43] Kim D-S, Kim D, Sugihara K. Voronoi diagram of a circle set from Voronoi diagram of a point set: I. Topology. Comput Aided Geom Design 2001;18: 541–62.
[44] Kim D-S, Kim D, Sugihara K. Voronoi diagram of a circle set from Voronoi diagram of a point set: II. Geometry. Comput Aided Geom Design 2001;18:563–85.
[45] Sugihara K. Approximation of generalized voronoi diagrams by ordinary Voronoi diagrams. Graph Models Image Process 1993;55(6):522–31.
[46] Emiris IZ, Tsigaridas EP, Tzoumas GM. Exact Voronoi diagram of smooth convex pseudo-circles: General predicates, and implementation for ellipses. Comput Aided Geom Design 2013;30(8):760–77.
[47] Emiris IZ, Tzoumas GM. Algebraic study of the apollonius circle of three ellipses. In: EuroCG. 2005. p. 147–50.
[48] Farouki RT, Johnstone JK. The bisector of a point and a plane parametric curve. Comput Aided Geom Design 1994;11:117–51.
[49] Alt H, Cheong O, Vigneron A. The Voronoi diagram of curved objects. Discrete Comput Geom 2005;34:439–53.
[50] Farouki RT, Johnstone JK. Computing point/curve and curve/curve bisectors. In: IMA conference on the mathematics of surfaces. 1992.
[51] Farouki RT, Ramamurthy R. Degenerate point/curve and curve/curve bisectors arising in medial axis computations for planar domains with curved boundaries. Comput Aided Geom Design 1998;15(6):615–35.
[52] Omirou SL, Demosthenous GA. A new interpolation algorithm for tracing planar equidistant curves. Robot Comput-Integr Manuf 2006;22:306–14.
[53] Cao L, Liu J. Computation of medial axis and offset curves of curved boundaries in planar domain. Comput Aided Des 2008;40(4):465–75.
[54] Cao L, Jia Z, Liu J. Computation of medial axis and offset curves of curved boundaries in planar domains based on the cesaros approach. Comput Aided Geom Design 2009;26:444–54.
[55] Barnhill RE, Farin G, Jordan M, Piper BR. Surface/surface intersection. Comput Aided Geom Design 1987;4:3–16.