Computer-Aided Design 127 (2020) 102880

Contents lists available at ScienceDirect

Computer-Aided Design

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

Continuous toolpath planning in a graphical framework for sparse infill additive manufacturing

Prashant Gupta[a], Bala Krishnamoorthy[a,*], Gregory Dreifus[b]

[a] Department of Mathematics and Statistics, Washington State University, United States of America
[b] Department of Mechanical Engineering, Massachusetts Institute of Technology, United States of America

ARTICLE INFO

Article history:
Received 23 March 2020
Received in revised form 22 April 2020
Accepted 29 April 2020

Keywords:
3D printing
Sparse infill
Graphical model
Eulerian tour
Continuous toolpath
Slicing

ABSTRACT

We develop a framework that creates a new polygonal mesh representation of the sparse infill domain of a layer-by-layer 3D printing job. We guarantee the existence of a single, *continuous* tool path covering each connected piece of the domain in every layer in this graphical model. We also present a tool path algorithm that traverses each such continuous tool path with *no crossovers*.

The key construction at the heart of our framework is a novel *Euler transformation* which converts a 2-dimensional cell complex K into a new 2-complex $\hat{K}$ such that every vertex in the 1-skeleton $\hat{G}$ of $\hat{K}$ has even degree. Hence $\hat{G}$ is Eulerian, and an Eulerian tour can be followed to print all edges in a continuous fashion without stops.

We start with a mesh K of the union of polygons obtained by projecting all layers to the plane. First we compute its Euler transformation $\hat{K}$. In the *slicing* step, we clip $\hat{K}$ at each layer using its polygon to obtain a complex that may not necessarily be Euler. We then *patch* this complex by adding edges such that any odd-degree nodes created by slicing are transformed to have even degrees again. We print extra *support* edges in place of any segments left out to ensure there are no edges without support in the next layer above. These support edges maintain the Euler nature of the complex. Finally, we describe a tree-based search algorithm that builds the continuous tool path by traversing "concentric" cycles in the Euler complex. Our algorithm produces a tool path that avoids material collisions and crossovers, and can be printed in a continuous fashion irrespective of complex geometry or topology of the domain (e.g., holes).

We implement our test our framework on several 3D objects. Apart from standard geometric shapes including a nonconvex star, we demonstrate the framework on the Stanford bunny. Several intermediate layers in the bunny have multiple components as well as complicated geometries.

1. Introduction

Additive manufacturing refers to any process that adds material to create a 3D object. 3D printing is a popular form of additive manufacturing that deposits material (plastic, metal, biomaterial, polymer, etc.) in layer by layer fashion. We focus on extrusion based 3D printing, in which material is pushed out of an extruder that follows some tool path while depositing material in beads that meld together upon contact. In this paper, we will refer to this process simply as 3D printing.

In *sparse infill* 3D printing, we first print the outer "shell" or boundary of the 3D object in each layer. We then cover the interior space by printing an *infill lattice* [1], which is typically a standard mesh. In an arbitrary infill lattice, one is not guaranteed to find a *continuous* tool path, i.e., an entire layer being printed by non-stop extrusion of material. Non-continuous tool paths typically have multiple starts and stops, which could reduce quality of the print, cause print failures (e.g., delamination), and could increase print time. To ensure existence of a continuous tool path, we need to choose the mesh modeling the infill lattice carefully. Subsequently, we need to develop algorithms that ensure a continuous tool path can be obtained for each layer with arbitrary geometry. Further, we need to identify a traversal of this tool path that avoids crossovers.

1.1. Our contributions

We propose a method that transforms a given 2-dimensional cell complex K into a new 2-complex $\hat{K}$ in which every vertex is part of a uniform even number of edges. Hence every vertex in the graph $\hat{G}$ that is the 1-skeleton of $\hat{K}$ has an even degree, which makes $\hat{G}$ Eulerian, i.e., it is guaranteed to contain an Eulerian

* Corresponding author.
E-mail addresses: prashant.gupta@wsu.edu (P. Gupta), kbala@wsu.edu (B. Krishnamoorthy), gdreifus@mit.edu (G. Dreifus).

https://doi.org/10.1016/j.cad.2020.102880
0010-4485/© 2020 Elsevier Ltd. All rights reserved.

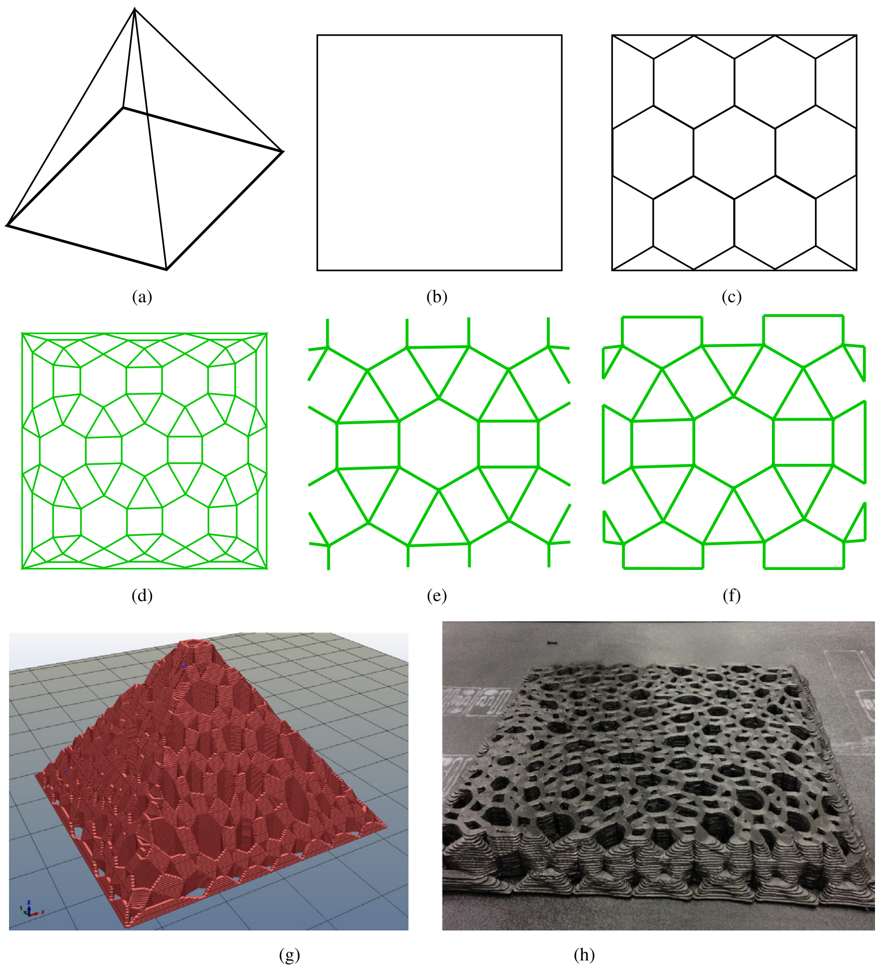

Fig. 1. Illustration of our framework (see Section 1.1 for details). SubFig. 1(g) shows the plan produced by our framework for printing the infill lattice of a 610 mm × 610 mm × 610 mm pyramid with 143 layers. SubFig. 1(h) shows details of the print after 23 layers. See Section 7 for other prints produced by our framework.

tour. We refer to this method as an *Euler transformation* of a polygonal (or cell) complex. For 2-complexes in $\mathbb{R}^2$ under mild assumptions (that no two adjacent edges of a polygon in K are boundary edges), we show that the Euler transformed 2-complex $\hat{K}$ has a geometric realization in $\mathbb{R}^2$, and that each vertex in its 1-skeleton has degree 4. We bound the numbers of vertices, edges, and polygons in $\hat{K}$ as small scalar multiples of the corresponding numbers in K.

We present a computational framework for 3D printing that identifies continuous tool paths for printing the infill lattice in each layer. We illustrate the steps in our framework in Fig. 1 (on a 3D pyramid with a square base). First we find the polygons for each layer of the input 3D domain (typically presented as an STL file, e.g., Fig. 1(a)) using a slicing software. Let $\mathcal{P}$ be the union of all of these polygons in 2D (Fig. 1(b)). We fill the space in $\mathcal{P}$ with some infill K, using any meshing algorithm (Fig. 1(c)). We then apply Euler transformation to obtain a new infill lattice $\hat{K}$ that is guaranteed to be Euler (Fig. 1(d)). In the next step, we *clip* $\hat{K}$ using P_i, a polygon in layer i (Fig. 1(e)). Depending on the shape of P_i, this step could create terminal vertices in the infill lattice for layer i, making it no longer Euler. In the last step, we *patch* the clipped infill lattice by adding new edges such that the resulting infill lattice is Euler again (Fig. 1(f)). An application of this framework is illustrated in Figs. 1(g) and 1(h). Finally, we propose a tool path algorithm (Section 6) that identifies the actual print tool path from the patched Euler infill lattice that avoids crossovers and material collisions. We address all geometric/computational challenges that arise along the way to ensure the proposed framework is complete. Since each layer can have multiple polygons in general, our framework can generate continuous tool path for each polygon in a given layer. As we might not print every boundary edge after the patching step, we also print *support* edges (see Section 5). The overall goal of our framework is to create an Euler infill lattice in each layer, and also prevent printing in free space so as to avoid print failures.

1.2. Related work

Euler Graph. The 1-skeleton of K is an undirected planar graph (G). One approach to make G Eulerian is to delete a minimal number of vertices and/or edges. But Cai and Yang [2] showed that the Euler vertex deletion and Euler edge deletion problems are NP-Complete. More importantly, removing edges and/or vertices could create gaps in the coverage of the domain, potentially affecting the mechanical properties of the final product. Another approach to make G Eulerian is to add a minimum number of edges that pair odd degree vertices in G, which can be cast as a *graph augmentation* problem. Boesch [3] presented a polynomial time algorithm for this problem, but planarity of G, which is necessary to avoid material collision, is not guaranteed after the augmentation. Another approach for augmentation is to use the Chinese postman problem to double the edges along shortest paths connecting odd degree vertices in G. But printing the resulting multigraph could be challenging due to non-uniform thickness of (multiple copies of) edges and non-uniform degrees of nodes. In fact, the degree of some of the nodes could be multiplied by up to $n^o/2$, where n^o is the number of odd degree vertices in G. Visiting a vertex a large number of times could reduce quality of the print.

Jin et al. [4] showed that subpaths in this setting could be generated using a geometric approach and then joined optimally to reduce the amount of jumps. Zhao et al. [5] proposed a method that finds global continuous paths using Fermat spirals. But both of these approaches are designed for completely filling the region (with or without holes), and not for sparse infilling.

The Catmull–Clark subdivision [6] creates quadrilateral polygons from any input 2-complex. But some vertices in the resulting mesh may have odd degrees.

Our Euler transformation appears similar to the Doo–Sabin subdivision [7], which creates new vertices for each polygon after each iteration. But Doo–Sabin subdivision does not preserve the underlying space, and creates boundary vertices with odd degree. It will add $O(b)$ jumps, where b is the number of boundary vertices after subdivision. Also, new vertices are created in the Doo–Sabin subdivision based on vertices and the number of edges in the original polygon. But in our Euler transformation, new vertices are created through mitered offsets (parallel offset of neighboring edges) of polygons. The length of edges get scaled roughly by a factor of 0.25 by Doo–Sabin subdivision when the number of vertices is large. In contrast, one gets much greater control on edge lengths in our Euler transformation by choosing mitered offsets suitably, independent of the number of vertices. Further, original angles are not preserved in the Doo–Sabin subdivision. Finally, combinatorial changes in general and topological changes for concave polygons are not allowed the Doo–Sabin subdivision. But our Euler transformation allows combinatorial as well as topological changes, thus avoiding small edges after transformation (see Section 4.1).

Tool path. While the perimeter and inset can typically be printed as continuous loops, tool paths for the infill lattice commonly resemble a back and forth pattern, a spiral, or a fractal-like path for an arbitrary geometry. The tool path consists of *print paths* (material is pushed through the nozzle) and *travel paths* (extruder moves from one location to other without pushing material). Galceran and Carreras [8] described coverage path planning as assignment of finding a path that passes through all the points without obstacles. Xu [9] presented the use of graph-based algorithms in coverage problems. General requirements for graph-based coverage problems such as all vertex coverage, non-overlapping and continuous paths, etc. [10] are applicable in 3D printing as well, including the requirement that each edge should be printed. One of the major steps in path planning is the identification of the tool path trajectory [11]. This tool path generation step involves filling interior regions and joining sub paths [12]. While attempts have been made to join sub paths into a continuous path, they are all limited by increasing complexity of geometry. Fewer sub paths in the tool path trajectory implies better quality of print.

Wang et al. [13] developed 3-dimensional infill (crossfill) using space filling curves, whose layer by layer cross-section is a continuous curve. Crossfill curves are fit into the infill region by intersecting with boundary of the polygon in a given layer, but this step can create multiple components. Later these components are connected into one continuous curve through an outer perimeter. New edges added to create these components are not guaranteed to have a support below it. We can still have material collision in each individual component if their boundary is too skewed or component is too thin.

Use of graphical models in additive manufacturing was demonstrated by Dreifus et al. [14]. They mesh each layer of the print as a graph, and find an Eulerian cycle over all edges of the graph. If the infill lattice is not Eulerian, they add "phantom edges" to the odd-degree vertices of the graph. When the extruder reaches an odd degree vertex, it stops printing, lifts above the printed material, moves to its matched vertex, and resumes printing. However, these stops and starts leave *teardrops* of material in their wake, as the extruder drags excess material behind it. Such teardrops weaken the print. Also, stopping and starting repeatedly increases print times. Further, their approach to identify the Eulerian cycle created *crossovers* when pairs of sub-paths of the tour cross each other.

2. Euler Transformation

We recently introduced the Euler transformation of polyhedral complexes in a general setting, with details provided for 2D and 3D cases [15]. Our framework for continuous tool path planning depends crucially on the Euler transformation in 2D, and uses extensively the related notation and definitions. We present the main results for the 2D case here.

2.1. Definitions on polygonal complexes

Definition 2.1 (*Polygonal Complex*). A polygonal complex K is a collection of *polygons* in $\mathbb{R}^2$ such that every face of a polygon in K is also included in K, and the nonempty intersection of any two polygons in K is a face of both. Polygons in K are referred to also as its 2-*cells*. We refer to K as a 2-complex.

We will work with *finite* polygonal complexes, i.e., where the set of cells in K is finite. The cells of interest in this work are vertices, edges, and polygons. Our definition of Euler transformation (in Section 2.2) as well as geometric realization results (in Section 3) do not require polygons in K to be convex. In general, some cells in the Euler transformed complex $\hat{K}$ may not be convex. But if we assume cells in K are convex, then we can guarantee a large majority of cells in $\hat{K}$ are so as well.

Definition 2.2 (*Pure Complex*). A polygonal complex is *pure* if every vertex and every edge is a face of some polygon in K.

Thus in a pure 2-complex, there are no "isolated" edges or vertices, and all top-dimensional cells are polygons.

We assume the input mesh K is a finite, connected, pure 2-complex in $\mathbb{R}^2$. Along with K, we assume we are given a collection $\mathcal{C}^H$ of polygons that capture 2-dimensional "holes", and a singleton set $\mathcal{C}^O$ that consists of a polygon capturing the "outside". Note that vertices and edges in the intersection of a polygon in K and a polygon in $\mathcal{C}^H$ or $\mathcal{C}^O$ are precisely the boundary cells of K. We make the following assumptions about intersections of polygons in K, $\mathcal{C}^H$, and $\mathcal{C}^O$. We denote the *underlying spaces* of these objects as $|K|$, $|\mathcal{C}^H|$, and $|\mathcal{C}^O|$, respectively. To be precise, $|\mathcal{C}^H| = \cup_{c_i \in \mathcal{C}^H} |c_i|$.

Assumption 2.3. The following conditions hold for the input complex K, the collection of holes $\mathcal{C}^H$, and the outside polygon $\mathcal{C}^O$.

1. $|K| \cup |\mathcal{C}^H| \cup |\mathcal{C}^O| = \mathbb{R}^2$.
2. Polygons in $\mathcal{C}^H$ are pairwise disjoint, and are also disjoint from the polygon that is $\mathcal{C}^O$.
3. Any polygon in K and a polygon in $\mathcal{C}^H$ intersect in at most *one* facet (edge) of both.
4. No two edges that are *adjacent facets* of a polygon in K intersect the polygon that is $\mathcal{C}^O$.

Hence polygons in K, $\mathcal{C}^H$, $\mathcal{C}^O$ cover $\mathbb{R}^2$, and each polygon in $\mathcal{C}^H$ captures a separate hole that is also separate from the outside.

We point out that *articulation* (or cut) vertices are allowed in K, i.e., vertices whose removal disconnects the complex (we

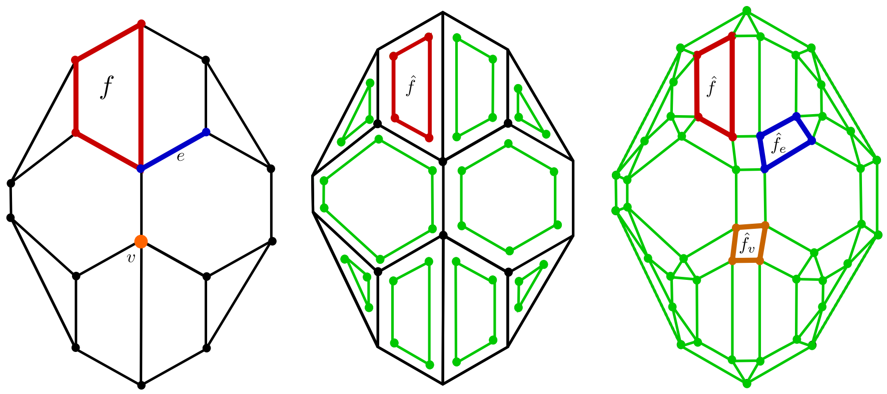

Fig. 2. A polygon f, edge e, and a vertex v highlighted in an input complex K (left), an intermediate complex showing only the copies of original polygons in K that are included in $\hat{K}$, i.e., of Class 1 (middle), and the final Euler transformation $\hat{K}$ (right).

assume K is connected to start with). Conditions specified in Assumption 2.3 ensure such vertices are boundary vertices of K. For instance, K could consist of two copies of the complex in Fig. 1(c) that meet at one of the four corner points.

2.2. Definition of Euler transformation

We define the Euler transformation $\hat{K}$ of the input 2-complex K by listing the polygons included in $\hat{K}$. We denote vertices as v (or u, v_i), edges as e (or e_i), and polygons as f (or f_i). The corresponding cells in $\hat{K}$ are denoted $\hat{v}, \hat{e}, \hat{f}$, and so on. We first define cells in $\hat{K}$ *abstractly*, and discuss aspects of geometric realization in Section 3.

We start by *duplicating* every polygon in $K \cup \mathcal{C}^H \cup \mathcal{C}^O$. Since we do not want to alter the domain in $\mathbb{R}^2$ captured by K, we set $\hat{\mathcal{C}}^H = \mathcal{C}^H$ and $\hat{\mathcal{C}}^O = \mathcal{C}^O$. But we "shrink" each polygon in K when duplicating (Section 3). By Assumption 2.3, this duplication represents each edge $e \in K$ by two copies in $\hat{K}$.

The polygons in $\hat{K}$ belong to three classes, and correspond to the polygons, edges, and vertices in K (see Fig. 2).

1. For each polygon $f \in K$, we include $\hat{f} \in \hat{K}$ as its copy.
2. Each edge $e \in K$ generates the 4-gon $\hat{f}_e$ in $\hat{K}$ specified as follows. Let $e = \{u, v\} \in f, f'$, where $f \in K$ and $f' \in K \cup \mathcal{C}^H \cup \mathcal{C}^O$. Then $\hat{f}_e$ is the polygon whose facets are the four edges $\{\hat{u}, \hat{v}\}$, $\{\hat{v}, \hat{v}'\}$, $\{\hat{u}', \hat{v}'\}$, and $\{\hat{u}, \hat{u}'\}$. Here, $\hat{v}, \hat{v}'$ are the two copies of v in $\hat{K}$. Edges $\hat{e} = \{\hat{u}, \hat{v}\}$ and $\hat{e}' = \{\hat{u}', \hat{v}'\}$ are facets of the Class 1 polygons $\hat{f}$ added to $\hat{K}$ or of polygons $\hat{f}'$ in $\hat{\mathcal{C}}^H$ or $\hat{\mathcal{C}}^O$. Edges $\{\hat{u}, \hat{u}'\}$ and $\{\hat{v}, \hat{v}'\}$ are added new.
3. Each vertex $v \in K$ that is part of p polygons in K generates a p-gon (polygon with p sides) $\hat{f}_v$ in $\hat{K}$ whose vertices and edges are specified as follows. Let $v \in f_k$ for $k = 1, \ldots, p$ in K. Then $\hat{f}_v$ has vertices $\hat{v}_k$, $k = 1, \ldots, p$, where $\hat{v}_k$ is the copy of v in $\hat{f}_k$ (in $\hat{K}$). For every pair of polygons $f_i, f_j \in \{f_k\}_1^p$ that intersect in an edge $e_{ij} \in K$, the edge $\hat{e}_{ij} = \{\hat{v}_i, \hat{v}_j\}$ is included as a facet of $\hat{f}_v$. Edges $\hat{e}_{ij}$ are ones added new as facets of Class 2 polygons (see above).

3. Geometric properties of Euler transformed complex

As the Euler transformation adds new polygons corresponding to input polygons, edges, as well as vertices, we *offset* the Class 1 polygons in $\hat{K}$ in order to generate enough space to add extra polygons. Intuitively, we "shrink" each of the polygons in K. We use standard techniques for producing offset polygons in 2D, e.g., *mitered* offset generated using the straight skeleton (SK) of the input polygon [16]. We define the polygon offset as a mitered offset of the input polygon that creates no *combinatorial* or *topological changes*—i.e., no edges are shrunk to points, and the polygon is not split into multiple polygons. Later on, we generalize the definition of Euler transformation to allow combinatorial or topological changes (Sections 4.1 and 4.2). Naturally, we do not want to alter the print domain $|K|$. Hence we include the polygons in $\mathcal{C}^H$ and $\mathcal{C}^O$ in $\hat{K}$ without any changes.

3.1. Geometric realization

Theorem 3.1. *Every vertex in the 1-skeleton of $\hat{K}$ has degree 4.*

Proof. Consider a vertex v shared by adjacent edges $e_1, e_2 \in f$, where $f \in K \cup \mathcal{C}^H \cup \mathcal{C}^O$ is a polygon. Following Assumption 2.3, the edges e_1 and e_2 are shared by exactly two polygons each from the input complex, holes, or the outside cell. Let f_1', f_2' be the other polygons containing edges e_1, e_2, respectively (with f being the first polygon).

Consider the vertex $\hat{v} \in \hat{K}$ generated as part of $\hat{f}$. The polygon $\hat{f}$ is a mitered offset of f when $f \in K$, or is identical to f when it belongs to $\mathcal{C}^H \cup \mathcal{C}^O$. Hence $\hat{f}$ is a simple polygon in both cases, and $\hat{v}$ is part of two edges $\hat{e}_1, \hat{e}_2 \in \hat{f}$. Further, $\hat{v}$ will be part of two more edges $\{\hat{v}, \hat{v}_1'\}$ and $\{\hat{v}, \hat{v}_2'\}$ added as facets of the Class 2 polygons generated by e_1, e_2. Here $\hat{v}_i' \in \hat{e}_i' \in \hat{f}_i'$ for $i = 1, 2$. Hence $\hat{v}$ has degree 4 in the 1-skeleton of $\hat{K}$. □

Remark 3.2. We show why we require the input complex to satisfy Conditions 3 and 4 in Assumption 2.3, which require that no two adjacent edges of a polygon in K can be boundary edges. Consider the input complex K consisting of a single square, whose four edges are shared with the outside cell $\mathcal{C}^O$. Then every vertex in the Euler transformation $\tilde{K}$ will have the odd degree of 3 (see Fig. 3). But if we apply the Euler transformation *once more* to $\tilde{K}$, we do get a valid complex $\hat{K}$ with each vertex having degree 4. Note that $\tilde{K}$ satisfies Condition 4, and hence becomes a valid input.

Lemma 3.3. *Let V, E, F denote the sets of vertices, edges, and polygons (faces) in K, and let $\hat{V}, \hat{E}, \hat{F}$ denote the corresponding sets in $\hat{K}$. The following relations hold for the cardinalities of these sets: $|\hat{V}| = 2|E|$, $|\hat{E}| = 4|E|$, and $|\hat{F}| = |V| + |E| + |F|$.*

Proof. Let $\delta(v)$ denote the degree of vertex $v \in K$, and let $\hat{f}_v$ be the polygon generated by v in $\hat{K}$. This is a Class 3 polygon. Following Assumption 2.3 about $K, \mathcal{C}^H, \mathcal{C}^O$, it is clear that when v belongs to p polygons in K, we must have $\delta(v) = p$ and $\hat{f}_v$ has p vertices. Since each cell $\hat{f}$ corresponding to polygon $f \in K$ is a mitered offset, and since each vertex $\hat{v}$ is part of one such offset

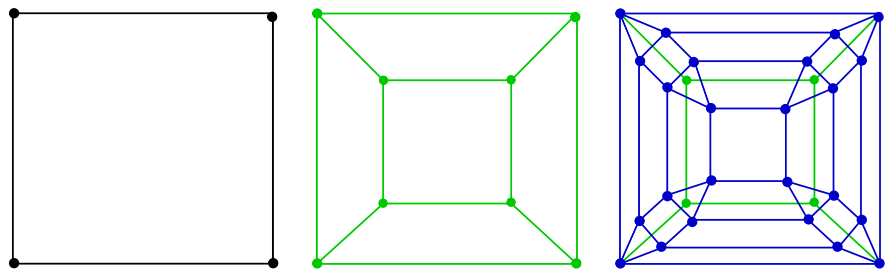

Fig. 3. Applying the Euler transformation to the square K whose more than one adjacent edge is shared with the outside (left) produces a complex $\tilde{K}$ in which every vertex has odd degree (middle). Applying the Euler transformation again to $\tilde{K}$ produces a valid complex $\hat{K}$ where every vertex has degree 4 (right).

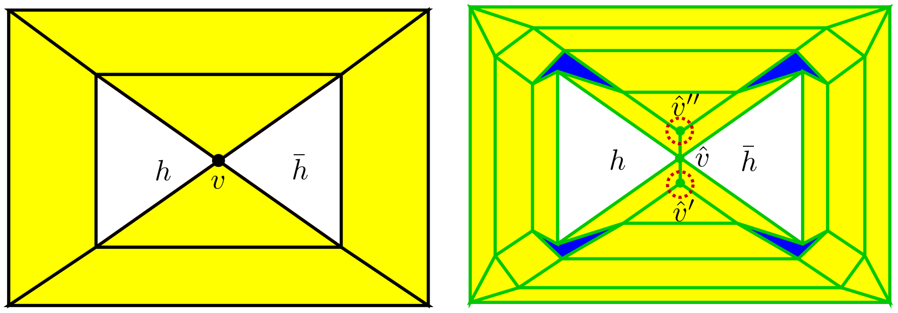

Fig. 4. Two holes touching at a vertex (left), and the result of applying Euler transformation (right). Vertices with odd degree in the result are circled. The four polygons shaded in blue are of Class 3 generated by vertices in the input complex (see Section 2). These cells could be nonconvex.

polygon, it follows that $\hat{f}_u \cap \hat{f}_v = \emptyset$ for any two vertices $u, v \in K$. Hence we get

$$|\hat{V}| = \sum_{v \in K} \delta(v) = 2|E|.$$

By Theorem 3.1, each vertex $\hat{v} \in \hat{K}$ has degree $\hat{\delta}(\hat{v}) = 4$ in $\hat{K}$. Combined with the result above on $|\hat{V}|$, we get that

$$|\hat{E}| = \frac{1}{2} \sum_{\hat{v} \in \hat{K}} \hat{\delta}(\hat{v}) = \frac{1}{2} \cdot 2|E| \cdot 4 = 4|E|.$$

Further, each polygon, edge, and vertex in K generate corresponding unique polygons in $\hat{K}$ belonging to three classes. Hence we get $|\hat{F}| = |F| + |E| + |V|$. □

Remark 3.4. While the number of edges in $\hat{K}$ is quadrupled, the total length of all edges gets roughly doubled. If we want to limit the total print length, we could start with a much sparser input complex K, and choose the mitered offsets of its polygons such that $|K|$ is covered adequately while limiting the total length of edges in $\hat{E}$.

Lemma 3.5. *The graph $\hat{G}$, the 1-skeleton of $\hat{K}$, is planar.*

Proof. By the definition of Euler transformation, each polygon $\hat{f} \in \hat{K}$ that is a mitered offset of polygon $f \in K$ is a simple closed polygon. Any two polygons $\hat{f}, \hat{f}' \in \hat{K}$ of Class 1 generated by polygons $f, f' \in K$ satisfy $\hat{f} \cap \hat{f}' = \emptyset$.

Class 2 polygons $\hat{f}_e, \hat{f}_{e'} \in \hat{K}$ generated by edges $e, e' \in K$ intersect at a vertex $\hat{v}$ if and only if e and e' are adjacent edges of a polygon $f \in K$ meeting at the vertex v. Since each $\hat{f} \in \hat{K}$ is a mitered offset of some polygon $f \in K$, at least one of the two copies $\hat{e}, \hat{e}'$ of edges in $\hat{K}$ corresponding to the edge $e \in K$ is shorter in length than e. If e is not a boundary edge then both $\hat{e}$ and $\hat{e}'$ are shorter than e. If e is a boundary edge then one edge out of $\hat{e}, \hat{e}'$ has the same length as e while the other is shorter. Hence each polygon $\hat{f}_e$ of Class 2 is a (convex) trapezium.

Since all edges of the polygon $\hat{f}_v$ of Class 3 generated by vertex $v \in K$ are precisely the *new* edges added to define the Class 2 polygons, each $\hat{f}_v$ is a simple closed polygon. Further, by the properties of Class 2 polygons specified above, $\hat{f}_v \cap \hat{f}_{v'} = \emptyset$ for any two vertices $v, v' \in K$.

Thus every polygon in $\hat{K}$ is simple and closed. Any two such polygons intersect at most in an edge or a vertex, and any two edges in $\hat{K}$ intersect at most in a vertex. Hence $\hat{G}$, the 1-skeleton of $\hat{K}$, is a planar graph. □

Remark 3.6. We illustrate why we require holes in the domain to be disjoint (Condition 2 in Assumption 2.3). Consider the input complex K with two holes $h, \bar{h} \in \mathcal{C}^H$ that intersect at a vertex v. The corresponding vertex $\hat{v}$ in the transformed complex $\hat{K}$ will not have a degree of 4. There will also be other vertices in $\hat{K}$ that have odd degree, which are circled in Fig. 4. Let these odd-degree vertices be labeled $\hat{v}', \hat{v}''$. Technically, there are two *identical* copies of the edge $\{\hat{v}, \hat{v}'\}$ and similarly of $\{\hat{v}, \hat{v}''\}$. But such duplicate edges make the graph $\hat{G}$ (1-skeleton of $\hat{K}$) nonplanar. If we include only one copy of each pair of duplicate edges, we get odd degree vertices in $\hat{G}$.

We pointed out in the Proof of Lemma 3.5 that the polygons of Class 2 in $\hat{K}$ generated by edges are convex 4-gons. Each polygon $\hat{f} \in \hat{K}$ of Class 1 is geometrically similar to the polygon $f \in K$ generating it. Hence if f is convex, so is $\hat{f}$. But polygons of Class 3 generated by vertices are not guaranteed to be convex. In fact, when $v \in K$ is a boundary vertex where K has a notch, or an "incut corner", $\hat{f}_v \in \hat{K}$ could be nonconvex—see Fig. 4 for illustrations. We finish with the result on $\hat{K}$ remaining connected.

Proposition 3.7. *If K is connected, then so is $\hat{K}$.*

Proof. We noted in the proof of Lemma 3.5 that the mitered offset polygons in $\hat{K}$ are pairwise disjoint. But we show that when polygons $f, f' \in K$ are connected, so are the corresponding offset polygons $\hat{f}, \hat{f}' \in \hat{K}$. By Assumption 2.3 on the input complex, when polygons $f, f' \in K$ intersect, they do so either in an edge

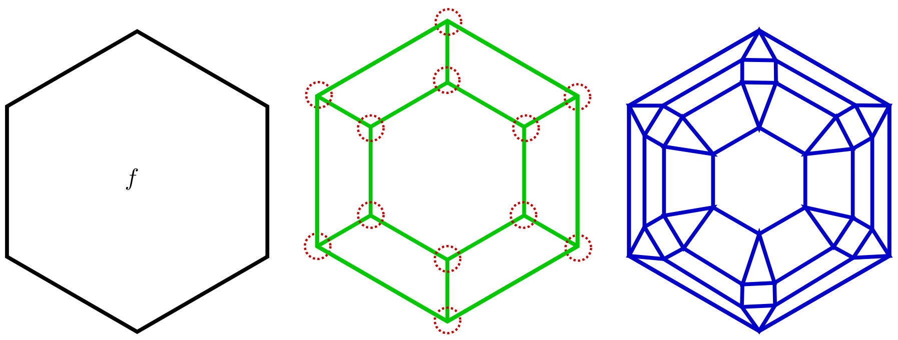

Fig. 5. $\hat{K}$ consisting of a single polygon f (left) and its Euler transformation $\hat{K}$ in green (middle). Vertices circled in red have odd degrees. The complex in blue (right) is the Euler transformation of $\hat{K}$, and its 1-skeleton is Euler.

e or in a vertex v. If $f \cap f' = e$, then by the definition of Euler transformation (Section 2), the corresponding offset polygons $\hat{f}, \hat{f}' \in \hat{K}$ are connected by the pair of new edges defining $\hat{f}_e$, the 4-gon of Class 2 generated by edge e. If $f \cap f' = v$ and v is not an articulation vertex, then the corresponding offset polygons $\hat{f}, \hat{f}' \in \hat{K}$ are similarly connected by the Class 3 polygon $\hat{f}_v$ generated by v, with the corresponding copies $\hat{v}$, $\hat{v}'$ of v in $\hat{f}, \hat{f}'$, respectively, being vertices of $\hat{f}_v$. If $f \cap f' = v$ that is an articulation vertex, then $\hat{v} = v$ is the identical copy of this vertex in $\hat{K}$. There will be two Class 3 polygons $\hat{f}_v, \hat{f}_v'$ generated by v in the two biconnected components joined at v, with $\hat{f}_v \cap \hat{f}_v' = \hat{v}$. Further, $\hat{f}_v$ is connected to $\hat{f}$ and $\hat{f}_v'$ to $\hat{f}'$, ensuring that $\hat{f}$ and $\hat{f}'$ are connected. It follows that $\hat{K}$ is connected when the input complex K is connected. □

4. Generalized Euler transformation

We now consider generalizations of the Euler transformation where we could relax parts of Assumption 2.3. The goal is to allow combinatorial and topological changes in the polygons undergoing transformation.

Consider a 2-complex K consisting of a single polygon f. K does not satisfy the input condition for Euler transformation, since adjacent edges are shared with the outside (Fig. 5). Nevertheless, we apply the transformation to K. In the resulting $\hat{K}$, all vertices will have odd degree. But this $\hat{K}$ satisfies the input condition. Hence if we apply the Euler transformation again to $\hat{K}$, i.e., we apply it *twice* on K, the resulting complex has a 1-skeleton that is Euler in the default setting. We define this process as the generalized Euler transformation.

Definition 4.1 (*Generalized Euler Transformation in $d = 2$*). Let K be a 2-dimensional cell complex in $\mathbb{R}^2$ with polygons possibly having adjacent boundary edges. Apply the Euler transformation on K to obtain $\hat{K}$, which will always satisfy Assumption 2.3. Now apply Euler transformation on $\hat{K}$.

We could use the generalized Euler transformation to improve mechanical properties of the design (by increasing the density of the mesh) in some regions while still guaranteeing that the 1-skeleton of the 2-complex is Euler. Note that the density of the mesh could increase significantly by this process.

Lemma 4.2. *After m transformations of complex K, the number of vertices $|\hat{V}^m| = 2 \cdot 4^{m-1}|E|$ and number of edges $|\hat{E}^m| = 4^m|E|$.*

Proof. After the first transformation, we get $|\hat{E}^1| = 4|E|$ (by Lemma 3.3). Extending the argument, after m transformations we get $|\hat{E}^m| = 4^m|E|$. Similarly, we get $|\hat{V}^1| = 2|E|$, then $|\hat{V}^m| = 2|\hat{E}^{m-1}| = 2 \cdot 4^{m-1}|E|$. □

We present Euler transformation with *combinatorial* and *topological* changes to a polygon resulting from mitered offset.

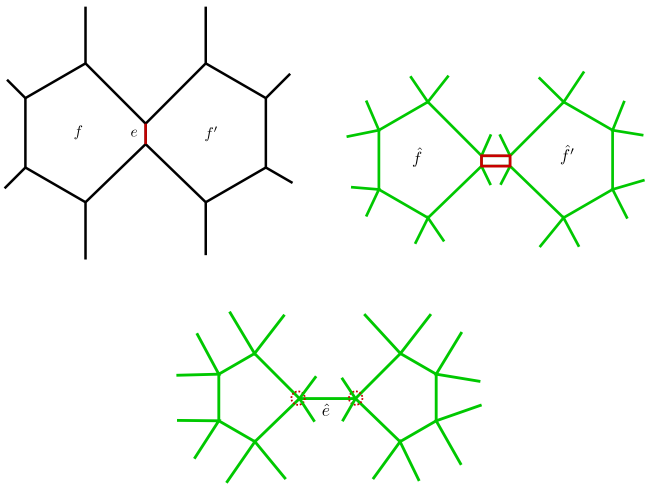

Fig. 6. Two polygons of a 2-complex K in the plane (top left). Euler transformation of K into $\hat{K}$ (top right), where $\hat{f}, \hat{f}'$ are Class 1 polygons and $\hat{f}_e$ (red) is a Class 2 polygon corresponding to edge e (red) in K. With a higher mitered offset, $\hat{f}_e$ is collapsed to edge $\hat{e}$; red circled vertices have odd degree (bottom).

4.1. Euler Transformation with combinatorial changes

Suppose $\hat{f}, \hat{f}_e, \hat{f}_v$ are 3 classes of polygons corresponding to a polygon (f), edge (e), and a vertex(v) in K. Maximum mitered offset in a polygon of the input complex K is limited by the smallest edge length, as the mitered offset in our original Euler transformation assumes no combinatorial and topological changes in $\hat{f}$. *Combinatorial change in $\hat{K}$ is a change in number of edges of some Class 1 polygon $\hat{f}$ in $\hat{K}$ from corresponding polygon in f in* K. Suppose polygon f has n edges and is now permitted to have combinatorial changes when generating its mitered offset. Then we can reduce at most $n - 3$ edges as we want $\hat{f}$ to still be a polygon, and a polygon has at least 3 edges. If $\hat{f}_e$ is sharing edges with two Class 1 polygons and if both of those edges are reduced, then $\hat{f}_e$ will collapse into an edge (see Fig. 6). Since Class 3 polygons ($\hat{f}_v$) do not share edges with any Class 1 polygons, combinatorial changes in the Euler transformation will not affect the number of edges in $\hat{f}_v$.

Lemma 4.3. *Let $\hat{v}$ is a vertex in a Class 1 polygon $\hat{f}$ of $\hat{K}$ created after collapsing π adjacent edges in $\hat{f}$, where combinatorial changes are allowed. Let $\hat{f}_e$ be a Class 2 polygon that contains one of these collapsed edges. If no $\hat{f}_e$ is collapsed to an edge, then degree of $\hat{v}$ is $2\pi + 4$ else $2\pi + 4 - m$ where m is number of polygons similar to $\hat{f}_e$ collapsed to an edge.*

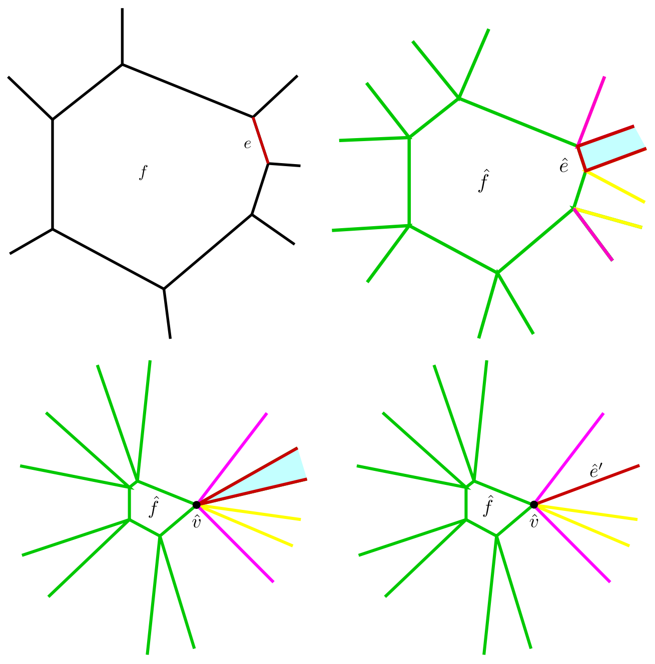

Fig. 7. f is a polygon in K (top left). Euler transformation of K into $\hat{K}$ (top right), where $\hat{f}$ is the Class 1 polygon corresponding to f, and $\hat{f}_e$ (blue) is the Class 2 polygon corresponding to edge e. Combinatorial changes are allowed in $\hat{K}$ in the bottom left figure, and edge $\hat{e}$ of $\hat{f}$ is collapsed to a point $\hat{v}$ where $\hat{f}_e$ (blue) is a triangle. As $\hat{e} \in \hat{f}_e$ is collapsed to a point, the degree of $\hat{v}$ in the 1-skeleton of $\hat{K}$ is $2(2)+4=8$. In the version of $\hat{K}$ shown in the bottom right figure, an edge of $\hat{f}_e$ shared with some Class 1 polygon other than $\hat{f}$ is also collapsed to a point. Here, $\hat{f}_e$ is collapsed to an edge $\hat{e}'$ (red) and the degree of $\hat{v}$ in the 1-skeleton of $\hat{K}$ is $2(2)+4-1=7$.

Proof. Since the polygon $\hat{f}$ is allowed to have combinatorial changes in $\hat{K}$, it will change degree of vertices in $\hat{f}$. π adjacent edges in $\hat{f}$ have $\pi+1$ vertices. Each end vertex of the path created by π adjacent edges adds 3 edges to $\hat{v}$, and each interior vertex of the path adds 2 edges to $\hat{v}$ (Fig. 7). Hence $\hat{v}$ has degree $2(\pi-1)+3+3=2\pi+4$ and 1-skeleton of $\hat{K}$ is still Euler. If $m>0$ Class 2 polygons sharing one of these adjacent edges are collapsed into edges, then two edges sharing $\hat{v}$ of each collapsed Class 2 polygon is replaced by one edge. Also, $m \le \pi$ since each distinct edge in any Class 1 polygon is shared by a unique Class 2 polygon in the Euler transformation. This implies $\hat{v}$ has degree $2\pi+4-2m+m=2\pi+4-m$ and 1-skeleton of $\hat{K}$ is Euler depending on m is even or odd (see Fig. 7). □

If combinatorial changes are allowed in Euler transformation, we should apply Euler transformation to a local complex for any odd degree vertices created. In the following lemma, we use the generalized Euler transformation to address the issue of odd degree vertices that may be created by combinatorial changes in $\hat{K}$.

Lemma 4.4. *Suppose $\hat{f}_e$ in $\hat{K}$ is collapsed to an edge $\hat{e}$, since combinatorial changes are allowed in $\hat{K}$. Suppose $\hat{k}$ is a sub 2-complex of $\hat{K}$ consisting of Class 3 polygons sharing any edge $\hat{e}$ in $\hat{K}$, and let $\hat{k}_j$ be a single component contained in $\hat{k}$, since $\hat{k}$ can have multiple components. Suppose $\hat{k}_j = \cup \tilde{k}_i$, where $\tilde{k}_i$ is sub 2-complex in $\hat{k}_j, \hat{K}$, and let $\tilde{k}_i, \tilde{k}_j$ do not share any edge in $\hat{K}$, if $i \neq j$. If $\check{k}_i$ is the generalized Euler transformation of $\tilde{k}_i$ and $\bar{k}_j = \cup \check{k}_i$ is single component in $\bar{k}$, then the 1-skeleton of $(\hat{K} \setminus \hat{k}) \cup \bar{k}$ is Euler.*

Proof. Since combinatorial changes are allowed in $\hat{K}$, let $\hat{v}$ be a vertex in $\hat{k}_j$ created after π adjacent edges of $\hat{f}$ are collapsed to a vertex. $\hat{k}_j$ is single component in $\hat{k}$, and only Class 3 polygons share an edge with any collapsed Class 2 polygons $\hat{f}_e$ is in $\hat{k}_j$. Hence $\hat{v}$ can be shared by some Class 3 polygon not in $\hat{k}_j$. If any Class 3 polygon sharing $\hat{v}$ is not contained in $\hat{k}_j$, this Class 3 polygon does not share an edge with any Class 2 polygon. Hence such cells do not belong to any component $\hat{k}_j$ in $\hat{k}$. Since Class 1 polygons are edge-disjoint from any Class 3 polygons, and there are some Class 3 polygons and one Class 1 polygon $(\hat{f})$ sharing vertex $\hat{v}$ in $\hat{K}$ but not contained in $\hat{k}_j$, we get that $\hat{v}$ is shared by an even number of additional edges not in $\hat{k}_j$. Since Class 1 and Class 3 polygons are edge-disjoint in $\hat{K}$, then any vertex $(\hat{v}')$ in some Class 3 polygon in $\hat{k}_j$ not similar to $\hat{v}$ has two more edges, not in $\hat{k}_j$ sharing $\hat{v}'$. Hence all the vertices in $\hat{k}_j$ have even number of additional edges in $\hat{K}$ not contained in $\hat{k}_j$ as shown in Fig. 8.

Let $\check{k}_i$ be the generalized Euler transformation of each sub 2-complex $\tilde{k}_i \in \hat{k}_j$. Then any vertex in $\hat{K} \setminus \hat{k} \cup \bar{k}$ has an even number of edges connected to it, since each $\bar{k}_j = \cup \check{k}_i$ contributes even number of edges to any vertex shared by $\bar{k}_j$ and each vertex in $\hat{k}$ has two more edges not contained in $\hat{k}$. Hence the 1-skeleton of $(\hat{K} \setminus \hat{k}) \cup \bar{k}$ is Euler. □

Let vertex $\hat{v}$ in the sub 2-complex $\tilde{k}_i$ of single component $\hat{k}_j$ in $\hat{k}$ have odd degree in $\tilde{k}_j$. We apply generalized Euler transformation to any such sub 2-complex $\tilde{k}_i$. Then by Lemma 4.4, the new 2-complex is Euler.

4.2. Euler Transformation with topological changes

When a polygon f in K is split into multiple Class 1 polygons in $\hat{K}$ after mitered offset, it is said to undergo a *topological change* (see Fig. 9). If a polygon in K is concave, then its mitered offset could create topological changes. Without loss of generality we assume the 2-complex K consists of polygons satisfying Assumption 2.3, and its polygons can be convex or concave.

Lemma 4.5. *Let the complex $\hat{K}$ be created by Euler transformation with some polygons undergoing only topological (no combinatorial) changes. Then the degree of vertices in $\hat{K}$ is even. Furthermore, if K is connected, then so is $\hat{K}$.*

Proof. When there are no topological changes, any vertex $\hat{v}$ in polygon $\hat{f}$ is shared by edge-disjoint Class 1 and 3 polygons in $\hat{K}$. Let us allow topological changes to $\hat{f}$. There are two possible cases. In the first case, $\hat{f}$ splits into multiple polygons at vertex $\hat{v}'$ by joining two or more vertices in $\hat{f}$ due to mitered offset. Then $\hat{v}'$ is still shared by edge-disjoint Class 3 polygons as shown in Fig. 9. In the second case, if we further increase the offset distance then $\hat{v}'$ will split into q new vertices $(\hat{v}'_1, \dots, \hat{v}'_q)$, where q is the number of Class 3 polygons sharing $\hat{v}'$. It will also join q Class 3 polygons sharing vertex $\hat{v}'$ into one polygon (F) as shown in Fig. 9. Then any $\hat{v}'_i$ is shared by edge-disjoint F and Class 1 polygons. Hence degree of vertices in $\hat{K}$ is even. There exists a path from any vertex of the split polygon $(\hat{f}_i)$ to any other vertex of $\hat{f}_j$, and hence $\hat{K}$ is connected. □

We have discussed cases when only combinatorial or only topological changes occur after transformation. But if *both* combinatorial and topological changes occur, then odd degree vertices may be created due to combinatorial changes. In this case, we apply local Euler transformation to some subcomplex of $\hat{K}$, and then by Lemma 4.4 the 1-skeleton of $\hat{K}$ is again Euler.

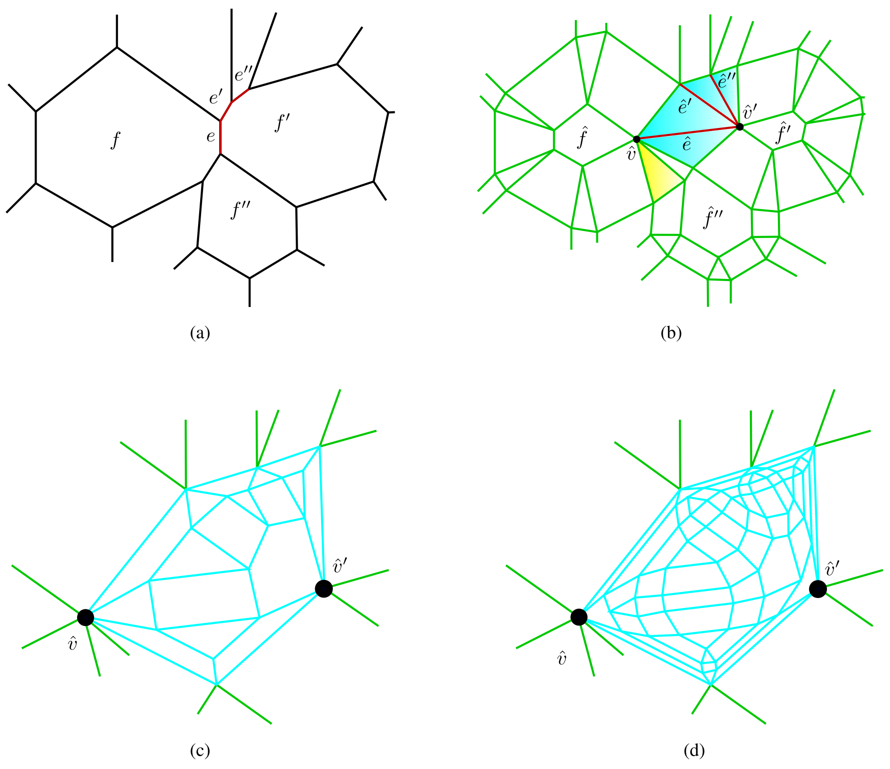

Fig. 8. (a) f, f', f'' are polygons in 2-complex K. (b) 2-complex $\hat{K}$ after Euler transformation with combinatorial changes to some polygons in $\hat{K}$. Class 2 polygons $\hat{f}_e, \hat{f}_{e'}, \hat{f}_{e''}$ corresponding to e, e', e'' in K are collapsed to edges $\hat{e}, \hat{e}', \hat{e}''$. Sub 2-complex $\tilde{k}_i$ (in blue) of some $\hat{k}_j$ in $\hat{K}$ consists of Class 3 polygons sharing edges $\hat{e}, \hat{e}', \hat{e}''$ in the 1-skeleton of $\tilde{k}_i$, and has vertices $\hat{v}, \hat{v}'$ with odd degree 7. The number of edges (green) at each vertex of $\tilde{k}_i$ not in $\tilde{k}_i$ are even, and $\hat{v}$ has one Class 3 polygon (yellow) not contained in $\tilde{k}_i$. (c) and (d) show generalized Euler transformation of $\tilde{k}_i$. (d) shows $\check{k}_i$ (blue), the generalized Euler transformation of $\tilde{k}_i$. Now $\hat{v}, \hat{v}'$ and other vertices of $\check{k}_i$ have even degrees in the 1-skeleton of $\hat{K} \setminus \tilde{k}_i \cup \check{k}_i$.

5. Slicing

The goal of our 3D printing approach is to have maximum continuous print path and minimum travel path (i.e., non-print path) in each layer. Further, when printing multiple layers on top of each other, we want to ensure there is no printing in free space. Ensuring we avoid printing in free space depends crucially on the geometric complexity of the object as well as on the first round of slicing. We first formalize the condition that the sequence of layers generated by slicing must satisfy in order to prevent printing in free space (Section 5.1). We assume this condition is satisfied by the layers of the input to our *clipping* procedure, which produces meshes for each polygon in a layer that are guaranteed to be Euler (Section 5.2).

5.1. ϵ-Continuous layers

Let $\mathcal{P}_i = \{P_{ij}\}$ and $\mathcal{P}_{i+1} = \{P_{i+1,j}\}$ are sets of the polygons in two consecutive layers created by slicing. The two layers are said to be ϵ-continuous if for every point $\mathbf{x} \in P_{i+1,j}$ there exists a point $\mathbf{y}$ in *some* $P_{ij} \in \mathcal{P}_i$ such that $d(\mathbf{x}, \mathbf{y}) \leq \epsilon$ for *all* $P_{i+1,j} \in \mathcal{P}_{i+1}$, where $\epsilon = cr$ with $0 \leq c \leq 1$ and r being the radius of extruder. The parameter c determines the maximum *overhang* allowed for the material deposited in a layer over the material in the layer immediately below. We assume there are sufficient numbers of perimeters in each layer to support the boundary edges in the layer above. Value of c is chosen based on various design and material considerations. There are alternative approaches to handle overhangs in specific cases, e.g., using self-supporting rhombic infill structures [17]. For general applicability of our framework, we assume the output of the slicing step in the design process produces layers that are ϵ-continuous in consecutive pairs.

5.2. Clipping

Suppose $\hat{K}$ is the Euler transformation of K, which meshes the union of polygons $\cup_i \mathcal{P}_i = \cup_{i,j} P_{ij}$ from all layers, where $P_{ij} \in \mathcal{P}_i$ is the jth polygon in ith layer. Each polygon P_{ij} has a region R_{ij} to be filled with infill lattice (note that $R_{ij} \subset P_{ij}$ can happen as some polygons may have edges along the boundary of the print domain). Suppose $\tilde{R}_{ij}$ is the inward Minkowski offset with a ball of radius r, the extruder radius, of the region R_{ij}. We will use $\tilde{R}_{ij}$ instead of R_{ij} to generate the infill lattice for P_{ij}. The reason behind this step is explained in Step 6 (to print the Support Perimeter). Let polygons R_{ij} and $\tilde{R}_{ij}$ be represented by the clockwise-ordered sequence of vertices $\{v_1, \ldots, v_n\}$ and $\{\tilde{v}_1, \ldots, \tilde{v}_n\}$, respectively. We define the *clip* operation for intersecting (or clipping) a 2-complex with a polygon, which may produce a 2-complex that may have multiple components, and may not be pure. We also define the *patch* operation that converts the 2-complex produced by a clip operation back into a connected pure 2-complex.

Definition 5.1 (*Clip*). We define how to construct $\tilde{K}$, the output of clipping the 2-complex $\hat{K}$ with polygon $\tilde{R}_{ij}$. Add to $\tilde{K}$ the polygons, edges, and vertices of $\hat{K}$ contained in $\tilde{R}_{ij}$. For edges in $\hat{K}$ cut by the boundary of $\tilde{R}_{ij}$, add to $\tilde{K}$ the portions inside $\tilde{R}_{ij}$ as new edges, and their points of intersection on the boundary of $\tilde{R}_{ij}$ as new vertices.

Note that the result of a clip operation may not necessarily be a pure 2-complex, and can have multiple components (see Fig. 10).

Definition 5.2 (*Patch*). Let $\tilde{K}$ be the output of a clip operation as specified in Definition 5.1. Suppose $S = \{\tilde{v}_{n+1}, \tilde{v}_{n+2}, \ldots, \tilde{v}_{n+m}\}$ is a clockwise ordered sequence of all points of intersection of

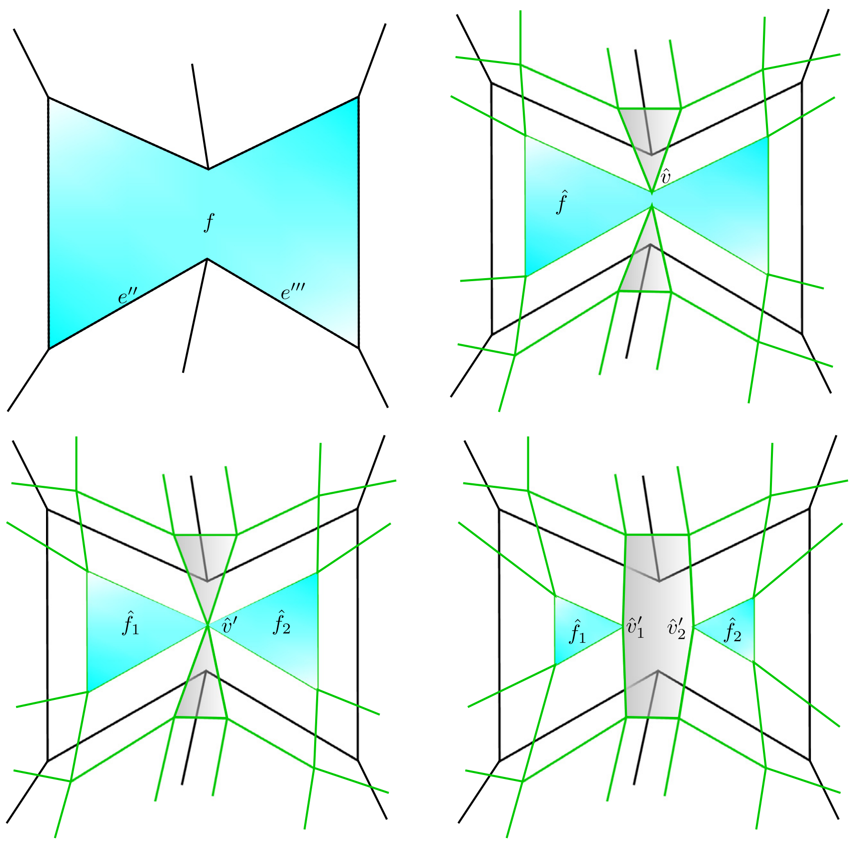

Fig. 9. Top left: Non-convex polygon f (blue) in K. Top right: Polygon $\hat{f}$ (blue) and Class 3 polygons (gray) in $\hat{K}$. Bottom left: $\hat{f}$ split into two polygons $\hat{f}_1, \hat{f}_2$ (blue) by joining two vertices of $\hat{f}$ to $\hat{v}'$. Since $\hat{f}_1, \hat{f}_2$ and Class 3 polygons (gray) at $\hat{v}'$ are edge-disjoint, $\hat{v}'$ has degree 8 and $q = 2$. Bottom right: With higher mitered offset, $\hat{f}_1, \hat{f}_2$ are completely disjoint and $\hat{v}'$ split up into $\hat{v}'_1, \hat{v}'_2$ creating one polygon F (gray), where $\hat{v}'_1, \hat{v}'_2$ has degree 4.

$\tilde{R}_{ij}$ and the 1-skeleton of $\hat{K}$ with odd degrees in the 1-skeleton of $\tilde{K}$ (note that m will be even). Since $\tilde{R}_{ij}$ can intersect edges in $\hat{K}$ between or at their end point(s), vertices in S can be terminal or boundary vertices in the 1-skeleton of $\tilde{K}$. Join alternate pair of vertices in S by a clockwise path on $\tilde{R}_{ij}$ as shown in Fig. 11. There are two possible choices of joining alternate pairs of vertices (1-2, 3-4, ... or 2-3, 4-5, ...). Pick the option that ensures the end vertices of components represented by subsequences of S are connected by these paths to end vertices of adjacent components. Also add new polygons to $\tilde{K}$ whose edges include the edges in new paths added as described above, new edges added by the clip operation on $\hat{K}$, and the edges of $\hat{K}$ contained in $\tilde{R}_{ij}$.

The Patch operation restores the Euler and connected nature of the input complex, except when the Clip step produces isolated simple paths. In the latter case, the Patch operation still leaves each component Euler.

Lemma 5.3. *Let $\hat{K}$ be a connected pure 2-complex and its 1-skeleton is Euler. Then $\tilde{K}$ produced by the patch operation on $\hat{K}$ is a connected pure 2-complex and its 1-skeleton is Euler, assuming none of the components in $\tilde{K}$ after the Clip step is a simple path.*

Proof. Clipping $\hat{K}$ with region $\tilde{R}_{ij}$ can create multiple components in the infill lattice if $\tilde{R}_{ij}$ intersects any polygon in $\hat{K}$ more than two times, or an edge more than once, or all edges connected to a vertex (see Fig. 10). Each component created in this process has an even number of odd degree vertices (by handshake lemma).

Let $S' = \{\tilde{v}_1, \ldots, \tilde{v}_p\}$ be a clockwise ordered subsequence of vertices in S of some component (with $p < m$). As specified in Definition 5.2, we join alternate pairs of vertices in S' by a path on R_{ij} (also see the Clipping Step 5) such that edge $\{\tilde{v}_1, \tilde{v}_2\}$ is not included. Since p is even, edge $\{\tilde{v}_{p-1}, \tilde{v}_p\}$ is also not included. Hence the first and last vertices in S' ($\tilde{v}_1$ and $\tilde{v}_p$) are left unpaired, but all intermediate vertices are now connected to $\tilde{K}$. In this case, the patch operations (in the Clipping step) will necessarily pair $\tilde{v}_1$ with a similar unpaired end vertex of the previous component, and also pair $\tilde{v}_p$ with the unpaired start vertex of the next component. Hence the extra edges added by the patch steps ensure that we get a single connected component. Also note that each odd degree vertex gets one additional edge, thus making its degree even. Hence the 1-skeleton of $\tilde{K}$ is Euler. Finally, the new polygons added to $\tilde{K}$ (as specified at the end of Definition 5.2) ensure that the resulting complex is pure. □

5.3. Continuous tool path planning framework: Steps

Our framework for continuous tool path planning consists of the following steps.

1. ***Slicing:*** Slice an STL file of the design. This step creates a sequence of layers, and each layer can have multiple polygons. Let $\mathcal{P}_i = \{P_{ij}\}$ be the set of all the polygons in layer i with or without holes. We assume the layers generated by slicing are ϵ-continuous.
2. ***Projecting:*** Project all polygons $\{P_{ij}\}$ in each layer $\mathcal{P}_i$ on to the horizontal plane. Take the **union** of all projected polygons (from all layers). This union can have an irregular shape depending on the input. Let P be the convex hull of the union of projected polygons. Note that taking the convex hull will avoid irregularities. We are assuming the

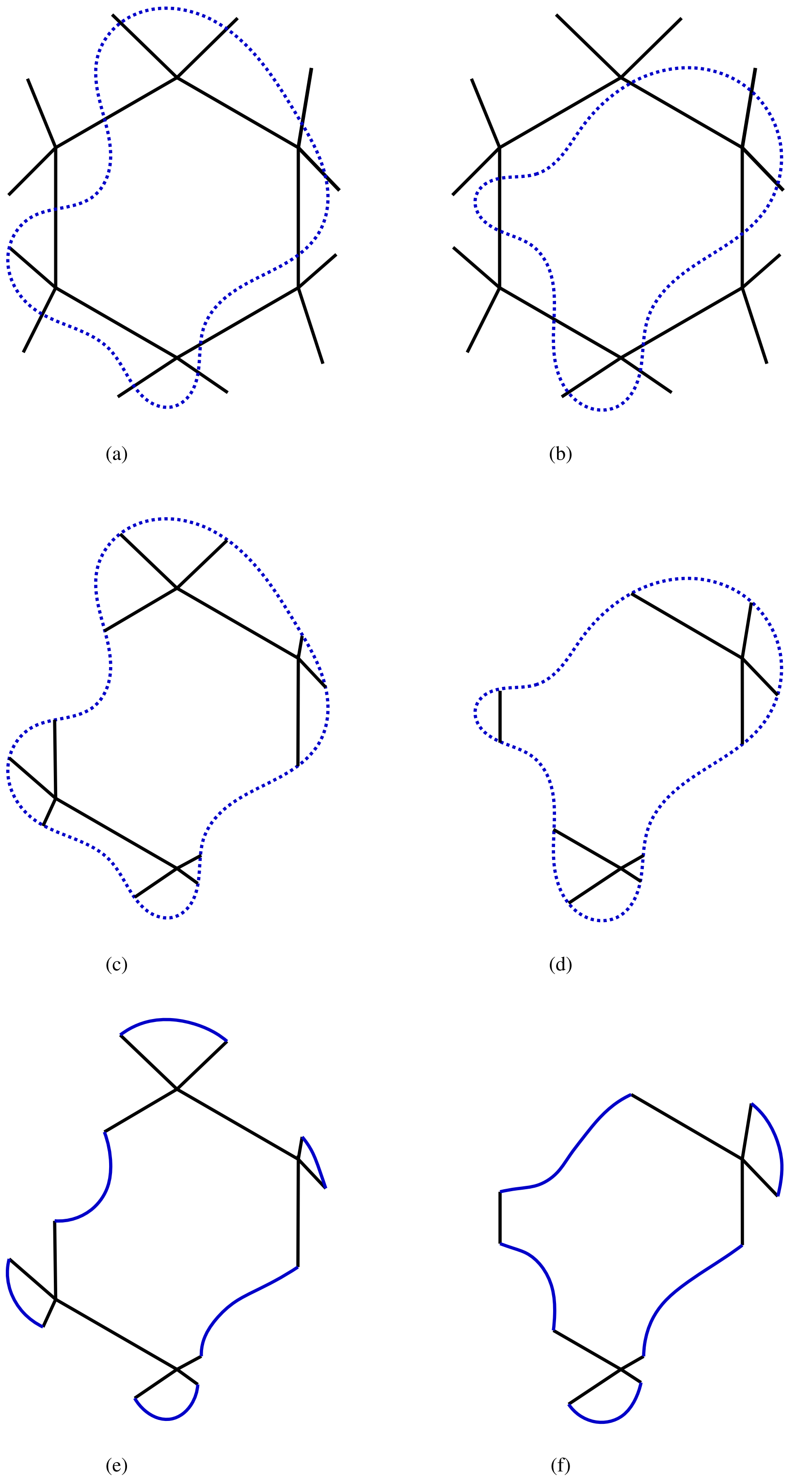

Fig. 10. Figures (a) and (b) show a 2-cell (black) of $\hat{K}$. Figures (c) and (d) show multiple components after Clip operations on $\hat{K}$ with respective polygons (dotted blue). Figures (e) and (f) show subsequent Patch operations connecting the multiple components with solid blue lines.

input design has a single component. If not, we can repeat the procedure for each component.

3. ***Meshing:*** Mesh P with a pure 2-complex K. We assume K satisfies Assumption 2.3.
4. ***Euler Transformation:*** Create $\hat{K}$ by Euler Transformation on K.
5. ***Clipping:*** $\tilde{K}$ is a 2-complex contained in $\tilde{R}_{ij}$. It is generated by Clip (Definition 5.1) and Patch (Definition 5.2) operations on $\hat{K}$ with respect to $\tilde{R}_{ij}$ (see Fig. 11).

   By Lemma 5.3, we get that $\tilde{K}$ is connected and its 1-skeleton is Euler since $\hat{K}$ has a single component and its 1-skeleton is Euler. There are two possible choices

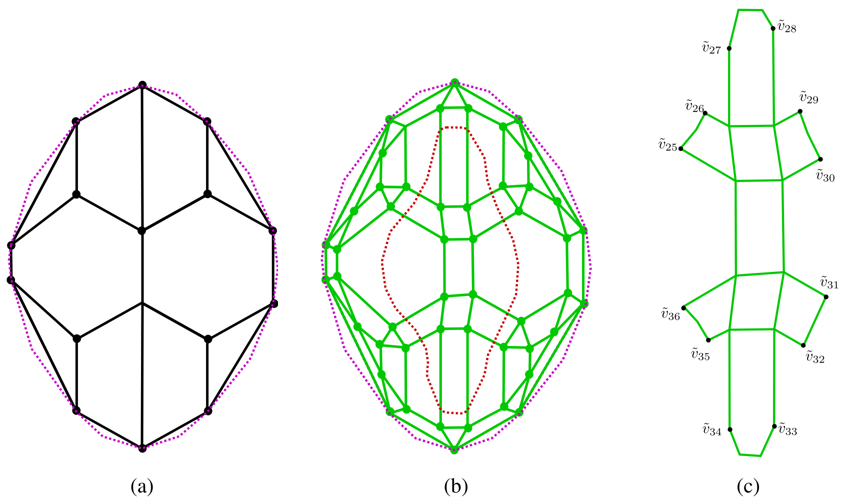

Fig. 11. (a) Projected polygon P (purple) and initial 2-complex K (black). (b) Euler transformation $\hat{K}$ (green) and Region $\tilde{R}_{ij}$ (red dots) of polygon P_{ij}. (c) $\hat{K}$ is clipped using $\tilde{R}_{ij}$ and patched to 2-complex $\tilde{K}$ (green; enlarged for better visibility) after Clipping Step 5, where $\tilde{R}_{ij}$ has $\{\tilde{v}_1, \ldots, \tilde{v}_{24}\}$ sequence of vertices and $\{\tilde{v}_{25}, \ldots\ldots, \tilde{v}_{36}\}$ are points of intersection of $\tilde{R}_{ij}$ and 1-skeleton of $\hat{K}$.

of pairing alternate vertices for patch operation. We can choose either option when $\tilde{K}$ is a single component. If $\tilde{K}$ has multiple components, either option leaves all resulting components Euler.

Printing the infill lattice in each layer amounts to printing edges in the 1-skeleton of $\tilde{K}$. We clip the 2-subcomplex $\tilde{K}$ for each layer from $\hat{K}$ to prevent printing in free space. Each edge is supported by an edge in the layer below it except for boundary edges in $\tilde{K}$. We will add a support perimeter to support boundary edges in $\tilde{K}$, as discussed in the next step.

6. ***Support Perimeter:*** Let $\tilde{R}_{ij} \in P_{ij}$ and $\tilde{R}_{i+1,j} \in P_{i+1,j}$ be such that $\tilde{R}_{i+1,j}$ is supported by $\tilde{R}_{ij}$ through ϵ-continuity as shown in Fig. 12. There are two possible ways we can select alternate pairs of vertices for subcomplex $\tilde{K}$ to join in $\tilde{R}_{i+1,j}$: {{12, 13}, {14, 15}, {16, 11}} or {{11, 12}, {13, 14}, {15, 16}}. Then the edges {12, 13} in the first case and {11, 12}, {13, 14} in second case are not supported if {2, 3}, {4, 5}, {6, 7}, {8, 9}, and {10, 1} are the vertices pairs selected for $\tilde{R}_{i+1,j}$. To solve this problem we need a way to print all the edges at boundary of the polygons.

Since we are not printing all the edges at the boundary of $\tilde{R}_{ij}$, we could have some overhanging boundary edges in $\tilde{K}$. Let $\tilde{P}$ be the nonprinted path on $\tilde{R}_{ij}$ between $\tilde{v}_{n+l}$ and $\tilde{v}_{n+l+1}$, which are vertices of S on $\tilde{R}_{ij}$. Add circles of radius r, the extruder radius, on $\tilde{v}_{n+l}$ and $\tilde{v}_{n+l+1}$ and on path $\tilde{P}$ such that neighboring circles do not intersect. We assume the circles only intersect neighboring line segments on the path. Suppose η is the maximum number of circles of radius r that can be added on path $\tilde{P}$, assuming there are 2 circles of radius r centered at end points of the path. Add η possible circles on path $\tilde{P}$, where the center of the j^{th} circle is $\tilde{v}'_j$. The total gap we can have between the circles is $2r - \delta$, where $0 < \delta < 2r$ as shown in Fig. 13(a). We can uniformly distribute the gap of $(2r - \delta)/(\eta + 1)$ between the circles as shown in Fig. 13(b) assuming η is at least one. Since $\tilde{R}_{ij}$ is the inward Minkowski offset of R_{ij} with a 2-ball of radius r, for any $\tilde{v}'_j$ there exists a point a such that line segment $\{\tilde{v}'_j, a\}$ is perpendicular to $\tilde{R}_{ij}$ and R_{ij}, and $d(\tilde{v}'_j, a) = 2r$. Let v_{n+k} and v_{n+k+1} be points of intersection with R_{ij} of the circle of radius r centered at a. Create a corner by adding edges $\{v_{n+k}, a\}$, $\{v_{n+k+1}, a\}$ as shown in Fig. 13(c). Connect end points of the corners by a path on R_{ij} if corner line segments do not intersect each other. Else change end points to the points of intersection so as to form a simple closed polygon as shown in Fig. 13(d). We illustrate in Fig. 14 the support perimeter around $\tilde{K}$ shown in Fig. 11(c).

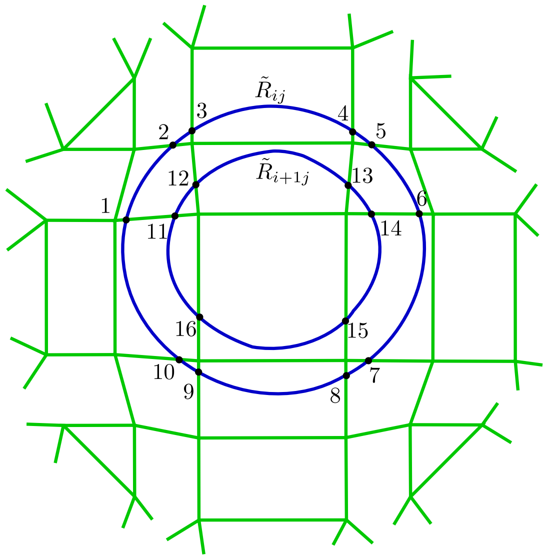

Fig. 12. $\tilde{R}_{ij}$ (blue) in P_i, $\tilde{R}_{i+1j}$ (blue) in P_{i+1} intersect the complex $\hat{K}$ (green) at points $\{1, \ldots, 10\}$ and $\{11, \ldots, 16\}$, respectively.

Remark 5.4. In Step 6, we add corners only if we can add circles of radius r on path $\tilde{P}$, given there are circles of radius r at the end point of the path. It is not guaranteed that line segments of $\tilde{P}$ will be covered by the support, since the coverage depends on the curvature of $\tilde{P}$ as shown in Fig. 13(d). An alternative approach to printing the support is to print the individual non-printed sections in $\tilde{R}_{ij}$ while making non-print travel moves in between. But this approach will have $m/2$ starts and stops. If $\hat{K}$ is a highly dense 2-complex, then m will be large and we will have a large number of starts and stops in this case.

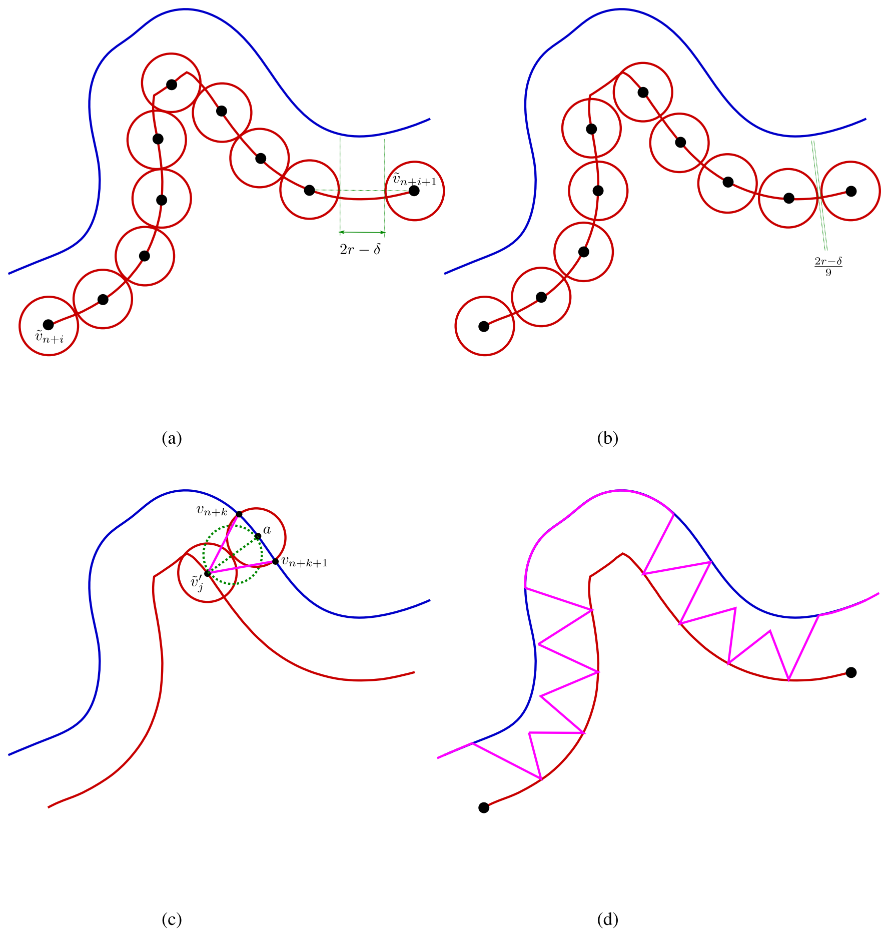

Fig. 13. (a) Portion of R_{ij} (blue) and $\tilde{R}_{ij}$ (red), $\tilde{P}$ (red), with total gap between circles being $2r-\delta$. (b) Uniformly distribute the gap into $(2r-\delta)/9$ parts between neighboring circles. (c) $\{v'_j, a\}$ is perpendicular to $\tilde{R}_{ij}$ and R_{ij}, circle centered at a intersects R_{ij} at v_{n+k}, v_{n+k+1}, and corner (pink) after adding edges $\{v'_j, v_{n+k}\}$, $\{v'_j, v_{n+k+1}\}$. (d) Neighboring corners joined (pink) to form a simple closed polygon.

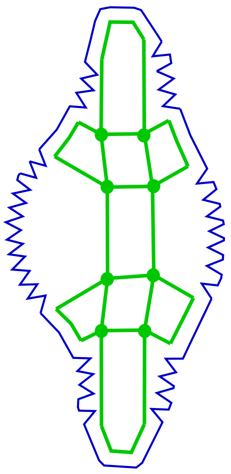

Fig. 14. Support perimeter for $\tilde{K}$ shown in Fig. 11(c) is shown in blue here.

6. Tool path algorithm

Since $\tilde{G}$, the 1-skeleton of $\tilde{K}$, is Euler, we can construct a tool path that consists only of the print path, i.e., all edges with none of them repeated. In general, such a tour can cross over itself at a vertex, creating a special case of material collision termed *crossover*. We present a tool path algorithm that chooses the subcycles in an Eulerian tour of $\tilde{G}$ carefully so as to avoid all crossovers. First we construct a circuit tree that represents $\tilde{G}$, with the vertices of the tree representing edge-disjoint circuits in $\tilde{G}$. Second, we add edge traversal restrictions in order to avoid crossovers in the tool path, which is described by specifying a traversal order of the circuit tree.

6.1. Circuit tree

Algorithm 1: CircuitTree.

```
1:  Unmark all the edges in $\tilde{K}$
2:  CircuitList = FindBoundaryCircuits($\tilde{K}$, $\emptyset$)   ▷ Initial CircuitList contains one circuit $C_{\text{Init}}$
3:  Mark all the edges in $\tilde{K}$ of $C_{\text{Init}}$
4:  pred($C_{\text{Init}}$) = $\emptyset$   ▷ Predecessor of $C_0$ is empty
5:  while CircuitList $\neq \emptyset$ do
6:      $C$ = CircuitList.pop(0)
7:      Circuits = FindBoundaryCircuits($\tilde{K}$, $C$)   ▷ Returns list of circuits
8:      CircuitList = CircuitList ∪ Circuits
9:      while Circuits $\neq \emptyset$ do
10:         $C_s$ = Circuits.pop(0)
11:         pred($C_s$)= $C$   ▷ predecessor of $C_s$ is $C$
12:         Mark all the edges in $\tilde{K}$ of $C_s$
```

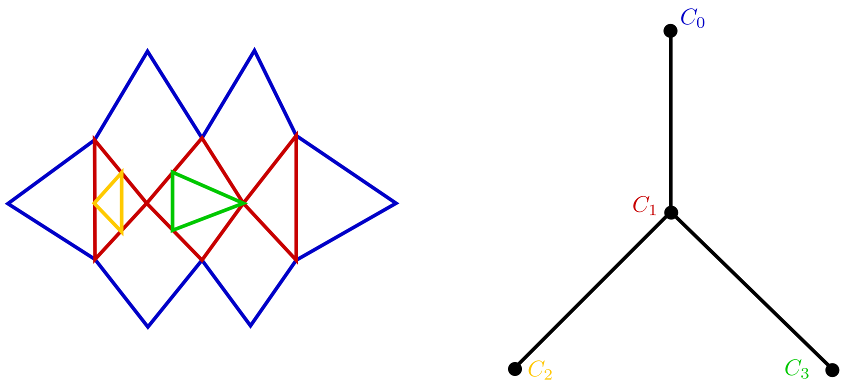

Fig. 15. $\tilde{K}$ (left) has C_0 (blue), C_1 (red), C_2 (yellow), C_3 (green) circuits. In the circuit tree (right), child circuits C_2, C_3 of C_1 are disjoint.

Algorithm 2: FindBoundaryCircuits($\tilde{K}$, C_0).

```
1: Find collection of edges $H$ based on $C_0$        ▷ edges of cells
   intersecting $C_0$ not in $C_0$
2: Circuits = {}
3: while $H \neq \emptyset$ do
4:     Find circuit $C$ using Modified Hierholzer's algorithm
5:     Remove all edges of $C$ from $H$
6:     Circuits = Circuits ∪ {C}
7: return Circuits
```

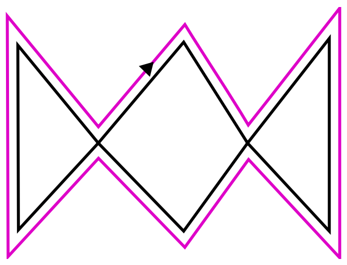

Fig. 16. Traversal of C_1 (black) from circuit tree in Fig. 15 in clockwise orientation with no crossovers is shown in pink.

Algorithm 1 constructs the circuit tree. It works by finding the outermost circuit in $\tilde{K}$ first, and continues to find all inner circuits in $\tilde{K}$, finishing with the innermost circuits. The outermost circuit in the 1-skeleton of $\tilde{K}$ consists of boundary edges in $\tilde{K}$. An innermost circuit in the 1-skeleton of $\tilde{K}$ contains no other circuit in its underlying space in $\tilde{K}$. The outermost circuit corresponds to the root node in the circuit tree, while innermost circuits are leaf nodes. The union of all circuits in the circuit tree is $\tilde{K}$. Every node in the Circuit tree represents a circuit. The predecessor of a circuit C (pred(C)) is another circuit connected to C at one or more vertices. All interior vertices in $\tilde{K}$ have even degree with at least degree 4 (some vertices may have even degree at least 6 due to applications of local Euler Transformation). Some vertices at the boundary of $\tilde{K}$ may have degree 2 as mentioned in step 5 of Section 5.2. But there is at least one boundary vertex of $\tilde{K}$ with degree at least 4, if there is more than one node in the circuit tree. This implies every circuit C can have at most d consecutive descendants in a path in the circuit tree, where $2d$ is the maximum degree of a vertex in C. Based on its construction, any two circuits in the circuit tree are disjoint if they are not on the same path starting at the root node in the circuit tree. An example circuit tree is shown in Fig. 15.

Given a circuit C_0, the algorithm finds inner circuits as follows. Suppose $\tilde{f}$ is a 2-cell in $\tilde{K}$ sharing edges $\{\tilde{e}_i\}$ in C_0 and none of its edges are marked in $\tilde{K}$ except $\{\tilde{e}_i\}$. Then all other edges of $\tilde{f}$ except $\{\tilde{e}_i\}$ are part of successor circuits in the circuit tree. Let H be the collection of all the edges in all 2-cells of the form $\tilde{f}$, except edges in C_0 (see Algorithm 2). Then H represents the next "onion layer" of boundary circuits. If C_0 is empty, then H is the collection of all the boundary edges in $\tilde{K}$.

Modified Hierholzer's algorithm: The original Hierholzer's algorithm [18] was designed for a connected graph, but in our case H can have multiple components. We want to orient the circuits in order to identify the tool path avoiding crossovers. We assume without loss of generality that each circuit is oriented clockwise as shown in Fig. 16. Hence subtours of this circuit are oriented clockwise as well. Pick a vertex in H, find all connected subtours $\{S_j\}$ and join them to obtain a circuit. Delete all the edges and vertices in this circuit from H. Repeat the process until H is empty.

Correctness: Since the 1-skeleton of $\tilde{K}$ is Euler, H consists only of circuits, and hence Algorithm 2 is guaranteed to terminate. It runs in $O(|E|)$ time, where E is set of edges in $\tilde{K}$.

Complexity: Identification of H, the collection of boundary edges in $\tilde{K}$, takes $O(|E|)$ time. For a given H, the modified Hierholzer's algorithm runs in $O(|E|)$ time. and we can have at most $|E|/3$ iterations of the outermost **while** loop in Algorithm 1. Hence the circuit tree algorithm runs in $O(|E|^2)$ time.

6.2. Traversal

Recall that we assume all circuits are oriented clockwise. To identify the tool path traversal that avoids crossovers, we traverse edges along the circuits in the circuit tree such that each parent and child circuit pair is traversed in opposite orientation. Let $\tilde{v}$ be a vertex in the circuit C_1 that is shared by two or more circuits in the tree, and has a degree $2d$. Then there are $q \leq d$ circuits sharing vertex $\tilde{v}$. Further, there exists a path $P = C_1 \rightarrow C_2 \rightarrow \ldots \rightarrow C_q$ in the circuit tree sharing vertex $\tilde{v}$ where C_1 is the ancestor and C_q is the descendant of all circuits in P. Orientations for all other circuits in the tree are uniquely determined, if orientation of root circuit is fixed. We assume without loss of generality that the traversal of edges in the root circuit in $\tilde{K}$ is clockwise.

There are two types of possible subpath crossovers while traversing the circuit tree (our goal is to avoid all crossovers). The first type of crossover is within a circuit (*type-1*) and the second type of crossover occurs while traversing from a parent to a child circuit (*type-2*). If any circuit C_i of the circuit tree is traversed along its orientation (as shown in Fig. 16, for instance), then it is guaranteed that there is no subpath crossover of (*type-1*). To prevent subpath crossovers of *type-2*, traverse edges of C_i and pred(C_i) in the circuit tree in *opposite* orientations along with

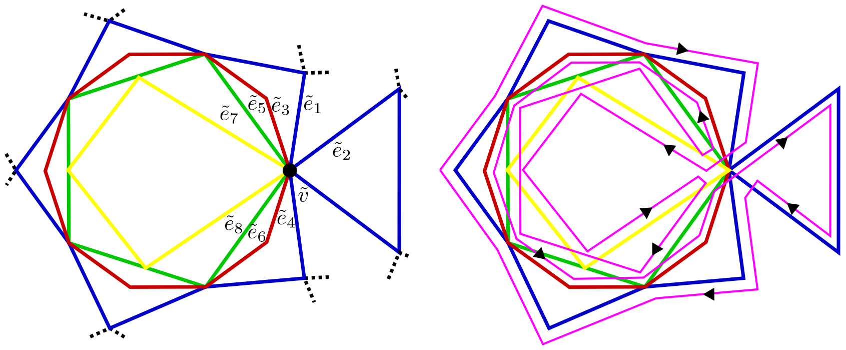

Fig. 17. Left figure shows a $q = 4$ case, with C_1 (blue), C_2 (red), C_3 (green), and C_4 (yellow) being the only circuits in a circuit tree sharing vertex $\tilde{v}$, where C_1 is an ancestor and C_4 is a descendant of all circuits in the path $p = \{C_1, C_2, C_3, C_4\}$ on the circuit tree. Right figure shows traversal (pink) of edges in $\tilde{K}$ of circuits C_1, C_2, C_3, C_4 with no crossover where $\tilde{e}_1 \longrightarrow \tilde{e}_7, \tilde{e}_8 \longrightarrow \tilde{e}_6, \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$ are edge traversal restrictions. Traversal of edges in C_1 is clockwise and C_4 is counterclockwise for $\tilde{K}$.

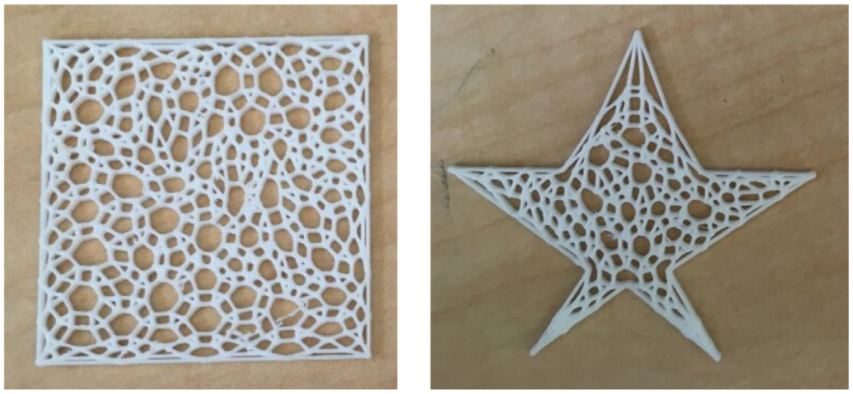

Fig. 18. Test prints of a square and a star domain with 10 layers each.

certain edge traversal restrictions. Thus we traverse edges of C_q clockwise when q is odd, else counterclockwise. The tool path traversal steps are detailed below.

1. Let $(\tilde{e}_{2j-1}, \tilde{e}_{2j})$ be a pair of clockwise ordered adjacent edges on a clockwise circuit path of C_j, sharing vertex $\tilde{v}$. Let $\tilde{e}_i \longrightarrow \tilde{e}_j$ imply we traverse $\tilde{e}_j$ immediately after we traverse $\tilde{e}_i$ in $\tilde{K}$. Add the following edge traversal restrictions in $\tilde{K}$ if q is odd: $\tilde{e}_1 \longrightarrow \tilde{e}_{2q}, \tilde{e}_{2q-1} \longrightarrow \tilde{e}_{2q-3}, \tilde{e}_{2q-2} \longrightarrow \tilde{e}_{2q-4}, \ldots, \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$, where $\tilde{e}_1 \longrightarrow \tilde{e}_{2q}$ means we traverse edge $\tilde{e}_1$ of cycle C_1 followed by $\tilde{e}_{2q}$ of cycle C_q. Similarly, $\tilde{e}_{2(q-i)-1} \longrightarrow \tilde{e}_{2(q-i-1)-1}$ means we traverse edge $\tilde{e}_{2(q-i)-1}$ of C_{q-i} followed by $\tilde{e}_{2(q-i-1)-1}$ of C_{q-i-1} and $\tilde{e}_{2(q-i)} \longrightarrow \tilde{e}_{2(q-i-1)}$ means we traverse edge $\tilde{e}_{2(q-i)}$ of C_{q-i} followed by $\tilde{e}_{2(q-i-1)}$ of C_{q-i-1}. If q is even, we add the restrictions $\tilde{e}_1 \longrightarrow \tilde{e}_{2q-1}, \tilde{e}_{2q} \longrightarrow \tilde{e}_{2q-2}, \tilde{e}_{2q-3} \longrightarrow \tilde{e}_{2q-5}, \ldots, \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$. Mark all the edges of the circuit tree on path P. An example with $q = 4$ is shown in Fig. 17.
2. Repeat Step 1 until all edges are marked in the circuit tree.
3. Start by traversing edges of the root circuit in clockwise direction, and follow traversal restrictions.

Complexity: We examine $O(|E|)$ edges in $\tilde{K}$ to add each edge restriction. Since the circuit tree can have at most $(|E|/3) - 1$ edges, the runtime of traversal restriction algorithm is $O(|E|^2)$.

7. Implementation

We first printed a few proof-of-concept shapes to test our Euler transformation framework. We printed the shapes shown in Fig. 18 with 10 layers each. We then scaled up the jobs to bigger sizes. The dimensions of the pyramid shown in Fig. 1(h) were 609.6 mm × 609.6 mm × 609.6 mm, and each layer had a height of 4.26 mm, resulting in a total of 143 layers.

We illustrate our complete framework on an object with non-trivial geometry and intermediate layer topology—the Stanford bunny. The height of the bunny is 83mm. We sliced the bunny into 415 layers each with height 0.2 mm, and we use an extruder diameter of 0.35 mm. First we found the minimal square (86mm × 86mm) containing the union of polygons from every layer. We triangulated this square using Pymesh. We then applied Euler transformation on this triangle mesh with a mitered offset of 1mm. Finally, we applied clipping and patching operations along with support perimeter, and used the tool path traversal algorithm to generate the infill toolpath for each layer. We also printed an extra perimeter in each layer to smooth out the surface. The entire computation ran in 1.5 hrs on a laptop. See Fig. 19 for illustrations of salient features of the print object.

8. Discussion

The bottleneck for computational complexity of the Euler transformation is determined by the mitered offsets it creates for each cell in K. The number of cells in $\hat{K}$ are clearly linear

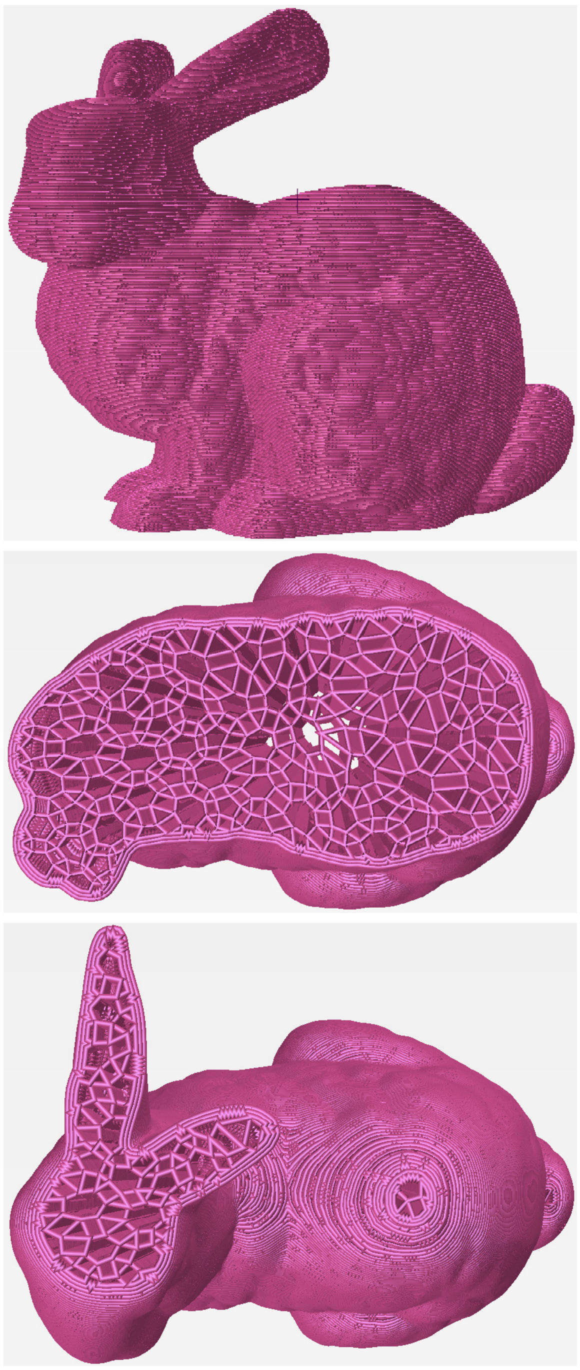

Fig. 19. Complete print of the Stanford bunny (top) using 415 layers, and views of layer 212 (middle) and layer 324 (bottom). Layer 324 consists of disconnected polygons. Support perimeter is evident in the intermediate layers.

in the number of cells in K (Lemma 3.3). For $d = 2$, if K in $\mathbb{R}^d$ has m d-cells, each of which has at most p facets, the time complexity of Euler transformation is $O(mp^d)$ [19,20]. Not all cells in the Euler transformation $\hat{K}$ are guaranteed to be convex, even when all cells in K are (see Fig. 4). We could triangulate the non-convex cells so that all cells in $\hat{K}$ are convex. But could we do so while maintaining even degrees for all vertices? A related problem is that of finding a triangulation (rather than a cell complex) of a given domain that minimizes the number of odd-degree vertices. The total Euclidean length of edges is $\tilde{K}$ is going to be at least double compared to that in the original complex in K. Hence it is better to start with a sparse input complex K (i.e., with a smaller total Euclidean length of edges). We have described a complete framework for continuous tool path planning in layer-by-layer 3D printing. The clipping step will be bottlenecked by the computation of intersection of the Euler transformed complex with each polygon in each layer. We have generalized the Euler transformation defined to allow combinatorial changes when computing mitered offsets of cells. What about allowing topological changes? It appears applying the generalized Euler transformation should be able to generate an Euler complex even when topological changes are allowed. But there might be some new geometric challenges generated in this process, which would have to be taken care of. We will address this question in future work. Another promising generalization of our approach would be to *non-planar* 3D printing. Many of our results should generalize to the non-planar realm as long as underlying support is guaranteed by the design.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was supported in part by an appointment of Gupta to the Oak Ridge National Laboratory (ORNL), United States of America ASTRO Program, sponsored by the U.S. Department of Energy (DOE) and administered by the Oak Ridge Institute for Science and Education. Krishnamoorthy acknowledges partial funding from the National Science Foundation, United States of America through grants 1661348 and 1819229. Dreifus acknowledges funding from the Manufacturing Demonstration Facility (MDF) of ORNL, United States of America and DOE, United States of America.

References

[1] Wu J, Aage N, Westermann R, Sigmund O. Infill optimization for additive manufacturing—approaching bone-like porous structures. IEEE Trans. Vis. Comput. Graphics 2018;24(2):1127–40.
[2] Cai L, Yang B. Parameterized complexity of even/odd subgraph problems. J. Discrete Algorithms 2011;9(3):231–40.
[3] Boesch FT, Suffel C, Tindell R. The spanning subgraphs of Eulerian graphs. J. Graph Theory 1977;1(1):79–84.
[4] Jin Y, He Y, Fu G, Zhang A, Du J. A non-retraction path planning approach for extrusion-based additive manufacturing. Robot. Comput.-Integr. Manuf. 2017;48:132–44.
[5] Zhao H, Gu F, Huang Q-X, Garcia J, Chen Y, Tu C, et al. Connected Fermat spirals for layered fabrication. ACM Trans. Graph. 2016;35(4):100:1–100:10.
[6] Catmull E, Clark J. Recursively generated b-spline surfaces on arbitrary topological meshes. Comput. Aided Des. 1978;10(6):350–5.
[7] Doo D, Sabin M. Behaviour of recursive division surfaces near extraordinary points. Comput. Aided Des. 1978;10(6):356–60.
[8] Galceran E, Carreras M. A survey on coverage path planning for robotics. Robot. Auton. Syst. 2013;61(12):1258–76.
[9] Xu L. Graph planning for environmental coverage [Ph.D. thesis], Pittsburgh, PA: Carnegie Mellon University; 2011.
[10] Cao ZL, Huang Y, Hall EL. Region filling operations with random obstacle avoidance for mobile robots. J. Robot. Syst. 1988;5(2):87–102.
[11] Ding D, Pan Z, Cuiuri D, Li H. A practical path planning methodology for wire and arc additive manufacturing of thin-walled structures. Robot. Comput.-Integr. Manuf. 2015;34:8–19.
[12] Jin Y, He Y, Du J. A novel path planning methodology for extrusion-based additive manufacturing of thin-walled parts. Int. J. Comput. Integr. Manuf. 2017;30(12):1301–15.
[13] Kuipers T, Wu J, Wang CC. CrossFill: Foam structures with graded density for continuous material extrusion. Comput. Aided Des. 2019;114:37–50, arXiv:1906.03027.

[14] Dreifus G, Goodrick K, Giles S, Patel M, Foster R, Williams C, et al. Path optimization along lattices in additive manufacturing using the Chinese postman problem. 3D Print. Addit. Manuf. 2017;4(2):98–104.

[15] Gupta P, Krishnamoorthy B. Euler transformation of polyhedral complexes. 2018, CoRR abs/1812.02412, arXiv:1812.02412.

[16] Aichholzer O, Aurenhammer F, Alberts D, Gärtner B. A novel type of skeleton for polygons. J. UCS 1995;1(12):752–61.

[17] Wu J, Wang CCL, Zhang X, Westermann R. Self-supporting rhombic infill structures for additive manufacturing. Comput. Aided Des. 2016;80:32–42.

[18] Hierholzer C, Wiener C. Ueber die Möglichkeit einen Linienzug ohne Wiederholung und ohne Unterbrechung zu umfahren. Math. Ann. 1873;6(1):30–2.

[19] Aurenhammer F, Walzl G. Three-dimensional straight skeletons from bisector graphs, in: Proceedings of 5th international conference analytical number theory and spatial tessellations, 2013, pp. 15–29.

[20] Aurenhammer F, Walzl G. Straight skeletons and mitered offsets of nonconvex polytopes. Discrete Comput. Geom. 2016;56(3):743–801.