Continuous Toolpath Planning in Additive Manufacturing

Abstract

We develop a framework that creates a mesh representation of the 3D domain of a layer-by-layer 3D printing job that permits the identification of single, continuous tool paths covering each connected domain in every layer. Our framework uses the *Euler transformation* of polyhedral meshes as the foundational tool, the details of which we present in a separate conjointly submitted paper. Every vertex in an Euler transformed mesh has an even degree, and hence the 1-skeleton can be covered by an Eulerian tour, i.e., the edges could be printed in a continuous fashion without stops.

We start with a standard mesh K of the union of polygons obtained by projecting all layers to the plane. We then obtain the Euler transformation $\hat{K}$ of this mesh K. In the *slicing* step, we clip $\hat{K}$ at each layer i using the polygon P_i of that layer to obtain $\hat{K}_i$ (we repeat this process for each polygon P_i in layer i). This clipping process might produce odd-degree nodes in $\hat{K}_i$. We describe a procedure to *patch* $\hat{K}_i$ by adding edges such that it is Eulerian again. If any segments at the boundary of P_i are left out by this procedure, we print extra *support* edges in their place to ensure there are no edges without support in the next layer above. This step of printing extra support edges maintains the Euler nature of the complex. As the final step, we describe a tree-based search algorithm that builds the continuous tool path by traversing concentric circles in the Euler complex. Our algorithm produces a tool path that avoids material collisions and crossovers.

We also generalize the definition of Euler transformation in $d = 2$ presented in the conjoint paper by permitting combinatorial changes when computing mitered offsets of polygons. To this end, we present a *generalized* Euler transformation that produces an Euler mesh of a polygon by applying the default Euler transformation twice.

Our proposed framework can print each connected component in a layer in a continuous fashion irrespective of its complex geometry or topology (e.g., with voids or holes).

Keywords: 3D printing, Eulerian tour, continuous toolpath. slicing, mitered offset.

1. Introduction

The term 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 to form the object. 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 most 3D printing jobs, 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, 2], 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. In a conjointly submitted manuscript, we have proposed an approach to transform any cellular complex under certain assumptions into an *Eulerian* complex. Every vertex in the 1-skeleton (i.e., graph) of an Eulerian complex has an even degree, and hence a continuous tool path is guaranteed to exist.

1.1. Our contributions

We present a computational framework for 3D printing that identifies continuous tool paths for printing the infill lattice in each layer. At the heart of our method is the Euler transformation of polyhedral meshes, the details of which we present in a conjoint submission. We illustrate the steps in our framework in Figure 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., Figure 1a)) using a slicing software. Let $\mathcal{P}$ be the union of all of these polygons in 2D (Figure 1b). We fill the space in $\mathcal{P}$ with some infill lattice K, using any meshing algorithm (Figure 1c). We then apply Euler transformation to obtain a new infill lattice $\hat{K}$ that is guaranteed to be Euler (Figure 1d). In the next step, we *clip* $\hat{K}$ using P_i, a polygon in layer i (Figure 1e). Depending on the shape of P_i, this step could create leaf nodes 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 (Figure 1f). Finally, we propose a tool path algorithm (Section 4) 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. Since we might not print every boundary edge after the patching step, we also print *support* edges (details in section 2). The overall goal of our framework is to create the infill lattice of each layer that is Euler, and also prevent printing in free space so as to avoid print failures.

1.2. Related work

Background on 3D printing. We present a brief non-technical overview of 3D printing [3]. In the first *design* step, one starts with a CAD model of the domain to be printed, and converts it into STL file. The STL file encodes surface information of the CAD model using triangles. In the next *slicing* step, one slices

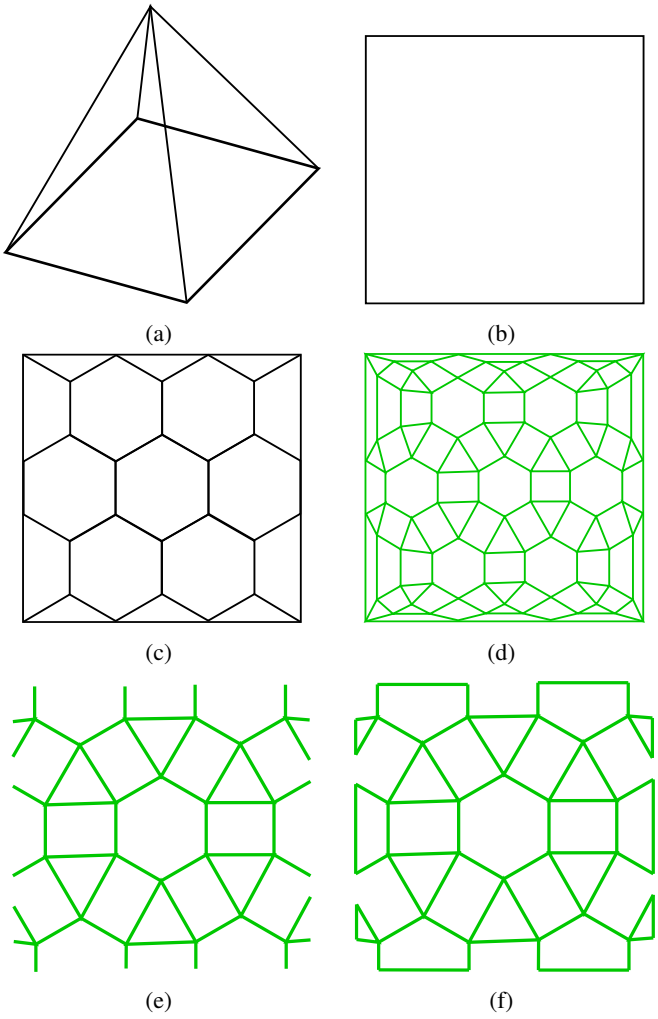

Figure 1: Illustration of our framework (see Section 1.1 for details).

the domain in the STL file using horizontal planes into multiple layers. In general, each layer has the same height. Each layer now has a set of polygons, representing the boundary of that layer. The *perimeter* for each layer is generated by offsetting each polygon inwards by one bead thickness. Then the *inset* is generated by offsetting the perimeter inward by one bead thickness into the region bounded by perimeter. The remaining interior region is filled with *infills*, which are usually represented as meshes. But standard are not guaranteed to be Euler. Since we can do Euler transformation of any complex, we can guarantee the existence of a continuous tool path for each polygon in a layer. Once we have a tool path for each layer, we can join all tool paths into a G-Code, which is read and implemented by the 3D printer. We need to ensure that each edge of the infill lattice of any given layer has material below it when it is printed. This requirement avoids printing in free space, which can sag or break the print due to its weight (except in the first layer).

Tool path. While the perimeter and inset can be typically printed as continuous loops, tool path planning for printing the infill lattice commonly generates tool paths resembling 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 [4] described coverage path planning as assignment of finding a path that passes through all the points without obstacles. Xu [5] showed 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. [6] 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 [7]. This tool path generation step involves filling interior regions and joining sub paths [8]. 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.

Use of graphical models in additive manufacturing was demonstrated recently by Dreifus et al. [9]. They mesh each layer of the print as a graph, and find a short 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 vertex, it stops printing, lifts above the printed material, moves to its matched vertex, and resumes printing. However, these stop and starts leave *teardrops* of material in their wake, as the extruder drags excess material behind it. The teardrops weaken the print. Also, stopping and starting repeatedly increases print times. Further, their approach to identify the Eulerian cycle created *crossovers*, with edges getting printed on top already printed edges.

1.3. Euler Transformation for $d = 2$

We introduce the Euler transformation of a pure complex in dimensions $d = 2, 3$ in a conjointly submitted paper. For the sake of completeness, we present here a brief summary of the $d = 2$ case, which is directly relevant for this paper. The input complex K is a pure 2-dimensional cellular complex (in $\mathbb{R}^2$). The Euler transformed complex $\hat{K}$ is obtained by adding three classes of 2-cells (polygons), referred to as Classes 1,2, and 3. Every 2-cell $f \in K$ generates a mitered offset $\hat{f} \in \hat{K}$ (Class 1). Intuitively, we shrink each cell inward. Each edge $e \in K$ generates a trapezium $\hat{f}_e \in \hat{K}$ (Class 2), and each vertex $v \in K$ generates a 2-cell $\hat{f}_v \in \hat{K}$ with as many edges as the degree of v in K (Class 3). An example of the Euler transformation is shown in Figure 2, illustrating the three classes of cells.

In our original definition of the Euler transformation, we assume there are no combinatorial or topological changes when generating Class 1 cells as mitered offsets. This assumption might be considered restrictive, especially when the complex has short edges. In Section 3, we relax this assumption to allow combinatorial changes in the $d = 2$ case. We also define a *generalized Euler transformation*, which guarantees the Euler nature of the output complex $\hat{K}$ by applying steps of the default Euler transformation twice.

2. 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 mutiple 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 cruicially 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 2.1). 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 2.2).

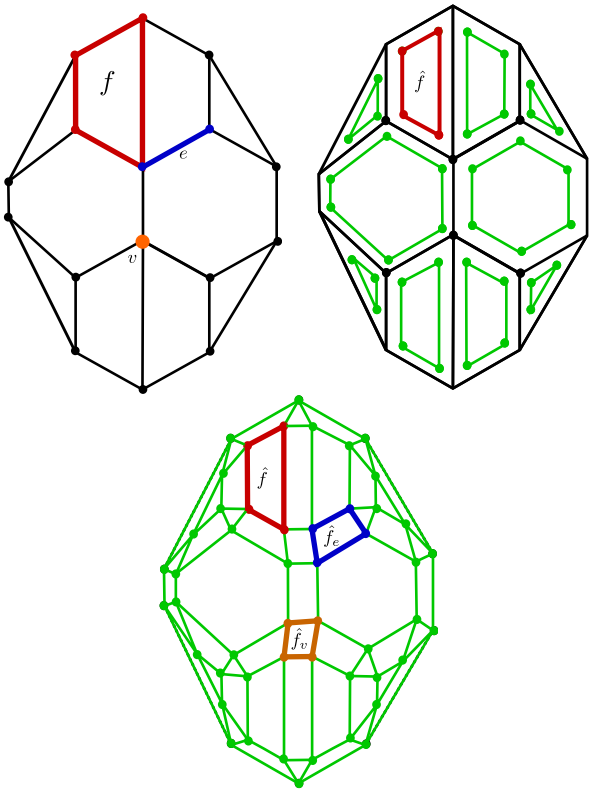

Figure 2: Illustration of Euler transformation in $d = 2$. A polygon f, edge e, and a vertex v highlighted in an input complex K (top left), an intermediate complex showing only the copies of original polygons in K that are included in $\hat{K}$, i.e., of Class 1 (top right), and the final Euler transformation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

2.1. ϵ-Continuous Layers

Let $\mathcal{P} = \{P_i\}$ and $\tilde{\mathcal{P}} = \{\tilde{P}_i\}$ 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 $\tilde{x} \in \tilde{P}_i$, there exists a point x in *some* $P_i \in \mathcal{P}$ such that $d(\tilde{x}, x) \leq \epsilon$ for *all* $\tilde{P}_i \in \tilde{P}$, where $\epsilon = cr$ for the radius of extruder r and $0 < c \leq 1$ (see Figure 3). 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 the number of perimeters is the same for all layers. Assuming the number of perimeters the two layers are the same, if two consecutive layers are not ϵ-continuous then the print may collapse due to printing in free space. The parameter c is chosen based on various design and material considerations. We assume the output of the slicing step in the design process produces layers that are ϵ-continuous in consecutive pairs.

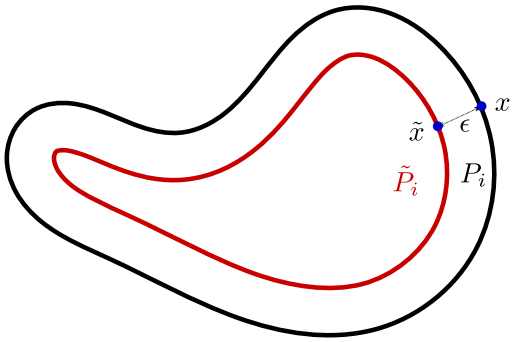

Figure 3: Red and Black polygons $\tilde{P}_i$ and P_i are in consecutive layers. Points $\tilde{x} \in \tilde{P}_i$ and $x \in P_i$.

2.2. Clipping

Suppose $\hat{K}$ is the Euler transformation of K, which meshes the union of polyhedra from all layers. Let clipping $\hat{K}$ with polygon P_i produce the 2-complex $\tilde{K}$. Then $\tilde{K}$ consists of 2-cells in K interior to P_i and free edges formed by trimming those 2-cells in K which intersect P_i. $\tilde{K}$ is not guaranteed to be a pure 2-complex, since free edges do not belongs to any 2-cells. The 1-skeleton of $\tilde{K}$ is guaranteed to have an *even* number of *free edges*. This result follows from the handshake lemma that says an undirected graph has even number of odd-degree vertices. Since $\hat{K}$ is Euler, only the vertices at the ends of free edges in $\tilde{K}$ have odd degree (of 1 each). Hence there must be an even number of free edges.

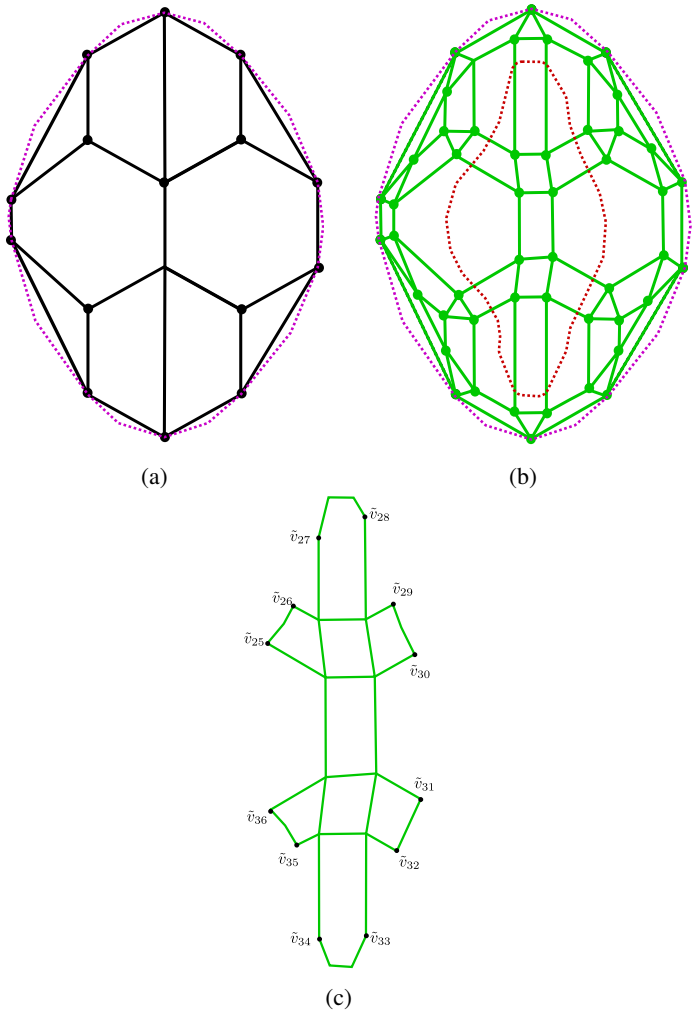

Figure 4: (a)Projected polygon P(purple) and initial 2-complex K is in black. (b) Euler transformation 2-complex $\hat{K}$(green), Region $\tilde{R}_j$(red dot) of some polygon P_i. (c) $\hat{K}$ is clipped using $\tilde{R}_j$ and patched to 2-complex $\tilde{K}$(green) after step 5, where $\tilde{R}_j$ has $\{\tilde{v}_1, ..., \tilde{v}_{24}\}$ sequence of vertices and $\{\tilde{v}_{25}, ..., \tilde{v}_{36}\}$ are points of intersection of $\tilde{R}_j$ and 1-skeleton of $\hat{K}$.

Our framework for continuous tool path planning includes the following steps.

1. ***Slice*** an STL file of the design. This step creates a sequence of layers, and each layer can have multiple polygons. Let $\{P_i\}$ be the set of all the polygons in layer i with or without holes. we assume the complete sequence of layers generated by slicing are ϵ-continuous.
2. ***Project*** all polygons $\{P_i\}$ in each layer i on the horizontal plane. Take the convex hull of the ***union*** of these projections

to create a new polygon P. Here we assume the design has a single component. If not, we can repeat the procedure for each component.

3. ***Mesh*** P with a pure 2-complex K.
4. Apply ***Euler Transformation*** to K to create $\hat{K}$. If K satisfies certain input conditions for the transformation, 1-skeleton of $\hat{K}$ is Euler. We assume the mitered offset of any 2-cell in $\hat{K}$ by the radius of extruder will not cause any combinatorial and topological changes.
5. Regions $\{R_j\}$ of P_i are assigned to the infill lattice. Suppose $\tilde{R}_j$ is the inward Minkowski offset with a ball of radius r of the region R_j. Let polygons R_j and $\tilde{R}_j$ be the clockwise-ordered sequence of vertices $\{v_1, ..., v_n\}$ and $\{\tilde{v}_1, ..., \tilde{v}_n\}$ on these regions. ***Clipping*** $\hat{K}$ with $\tilde{R}_j$ will generate the 2-complex $\tilde{K}$ which contains an even number of points of intersection of odd degree in $\tilde{K}$ (by handshake lemma). We assume that if any 2-cell in $\hat{K}$ is intersected by the polygon $\tilde{R}_j$, it will intersect only twice. Suppose $S = \{\tilde{v}_{n+1}, \tilde{v}_{n+2}, ..., \tilde{v}_{n+m}\}$ is a clockwise order sequence of all points of intersection of $\tilde{R}_j$ and 1-skeleton of $\hat{K}$ with odd degree in 1-skeleton of $\tilde{K}$, where m is even. Since $\tilde{R}_j$ can intersect edges between ends points or at the end point of edges in the 1-skeleton of $\hat{K}$, all the vertices in S can be terminal or boundary vertices in the 1-skeleton of $\tilde{K}$. Join alternating pair of vertices in S by a clockwise path on $\tilde{R}_j$ as shown in Figure 4. There are two possible choices of pairing alternate vertices (1-2, 3-4,..., or 2-3, 4-5, ...). We choose the one that has the longer total length of edges, so as to maximize the printed edges.
   Add new 2-cells consisting of edges contained in the added new path, edges of trimmed 2-cells in $\hat{K}$ now part of the boundary in $\tilde{K}$, and free edges. $\tilde{K}$ is now a pure 2-complex and connected.
6. Since we are not printing all the edges at the boundary of $\tilde{R}_j$, we could have some overhanging boundary edges in $\tilde{K}$. Let $\tilde{p}$ is non printed path on $\tilde{R}_j$ between $\tilde{v}_{n+i}$, $\tilde{v}_{n+i+1}$ vertices of S on $\tilde{R}_j$. Add circles of radius r on $\tilde{v}_{n+i}$,$\tilde{v}_{n+i+1}$ 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 $\tilde{v}'_j$. Total gap between the circles we can have is $2r - \delta$, where $0 < \delta < 2r$ as shown in Figure 5a. We can uniformly distribute the gap of $\frac{2r-\delta}{\eta+1}$ between the circles as shown in Figure 5b. Since $\tilde{R}_j$ is the inward Minkowski offset of R_j with a 2-ball of radius r, then for any $\tilde{v}'_j$ there exists a point a such that line segment $\{\tilde{v}'_j, a\}$ is perpendicular to $\tilde{R}_j$ and R_j, and $d(\tilde{v}'_j, a) = 2r$. Let v_{n+k}, v_{n+k+1} be points of intersection of the circle of radius r centered at a with R_j. Create a corner by adding edges $\{v_{n+k}, a\}$, $\{v_{n+k+1}, a\}$ as shown in Figure 5c. Connect end points of the corners by a path on R_j if corner line segments do not intersect each other. Else change end points to points of intersection to form simple closed polygon as shown in Figure 5d.

Remark 2.1. In Step 6 above, 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 Figure 5d. An alternative approach to printing the support is to print the individual

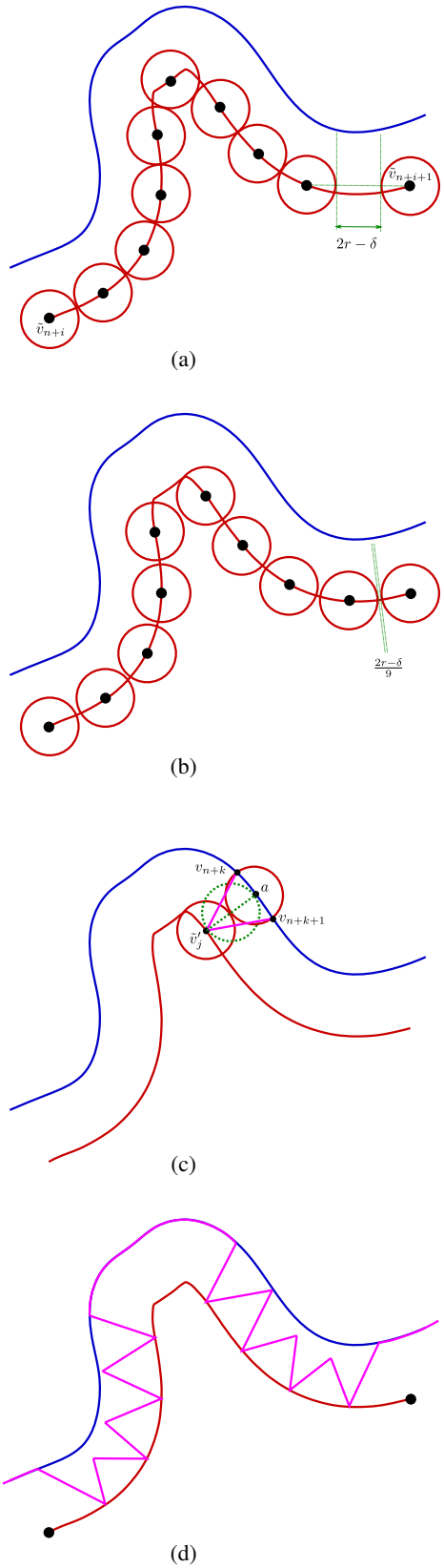

Figure 5: (a) Portion of R_j (blue) and $\tilde{R}_j$ (red), $\tilde{p}$ (red), with total gap between circles being $2r-\delta$. (b) Uniformly distribute the gap between neighboring circles $\frac{2r-\delta}{9}$. (c) $\{v'_j, a\}$ is perpendicular to $\tilde{R}_j$ and R_j, circle centered at a intersects R_j 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 simple closed polygon.

non-printed sections in $\tilde{R}_j$ 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.

This step ensures each edge of infill lattice is supported by the material below it and is Euler. Since 2-complex $\hat{K}$ is generated on projected polygon P, the infill lattice for any polygon P_i in a given layer generated by clipping can cause the following problems. We also present ways to deal with each problem.

- Clipping $\hat{K}$ with region ($\tilde{R}_j$) can create *multiple components* in infill lattice if $\tilde{R}_j$ intersects any 2-cell in $\hat{K}$ more than two times, or an edge more than once, or all edges connected to a vertex is intersected as shown in Figure 6. Each component after clipping $\hat{K}$ has even number of odd-degree vertices by handshake lemma. Let $s = \{v_1, ..., v_m\}$ be a clockwise ordered subsequence of vertices in S of some component. Join alternating pairs of vertices in s by a path on R_j according to step 5 such that there is no edge $\{v_1, v_2\}$. Since m is even, there is no edge $\{v_{m-1}, v_m\}$ either. Suppose s_1, s_2 are clockwise ordered subsequences of all vertices in S of two components. If there exist a path (p) on R_j between first and last vertex in some s_1, s_2 or s_2, s_1 such that vertices between start and end vertices of the path do not belong to any component, then they are *neighboring* components. Start and end vertices of a clockwise ordered subsequence in S of a component are joined by a path p, with end or start vertex of a clockwise ordered subsequence in S of neighboring component. Hence $\tilde{K}$ is connected and its 1-skeleton is Euler as shown in Figure 6. Since each component is connected and neighboring components are connected by an edge.

- Any 2-cell in $\hat{K}$ is convex since P is a convex hull. Although boundary 2-cells in $\tilde{K}$ can be non-convex if P_i is non-convex. Some boundary 2-cells in $\tilde{K}$, if not contained in $\hat{K}$, can cause *material collision* as shown in Figure 7a, since those 2-cells become *thin* and assumption in Step 4 is not necessarily satisfied. Let $\tilde{f}$ be a boundary 2-cell in $\tilde{K}$, $\bar{f}$ is its mitered offset with offset distance r (radius of extruder) and l is set of 1-cells added into $\tilde{K}$ in Step 5 to create 2-cell $\tilde{f}$. Note that $\tilde{f}$ can be shrinkable or unshrinkable with mitered offset distance r.

  Case 1. *$\tilde{f}$ is shrinkable.*

  If $\tilde{f}$ is shrinkable, then $\bar{f}$ can differ from $\tilde{f}$ both combinatorially and topologically. Let $S = \{\bar{v}_1, ..., \bar{v}_n\}$ is clockwise ordered sequence of vertices in $\bar{f}$. If any edge $\{\tilde{v}_i, \tilde{v}_{i+1}\}$ in S is a supporting edge of some edge $\{\bar{v}_j, \bar{v}_{j+1}\}$ in $\bar{f}$, then add these edges into new sequence T, the clockwise ordered sequence of all vertices of $\bar{f}$. If $\{\tilde{v}_{k-1}, \tilde{v}_k\}$ is a supporting edge for $\{\bar{v}_{j-1}, \bar{v}_j\}$ and $\{\bar{v}_j, \bar{v}_{j+1}\}$ is not an edge or $\{\bar{v}_j, \bar{v}_{j+1}\}$ is an edge with supporting edge $\{\tilde{v}_{k+m}, \tilde{v}_{k+m+1}\}$, then find point of intersection of perpendicular line through $\bar{v}_j$ to $\{\tilde{v}_k, \tilde{v}_{k+1}\}$, if $\{\tilde{v}_k, \tilde{v}_{k+1}\} \in l$. Find all of these point of intersection and add it to $\tilde{f}$. Let $U = \{\tilde{v}_1, ..., \tilde{v}_k, v'_i, \tilde{v}_{k+1}, ..., \tilde{v}_n, ...\}$ is clockwise ordered sequence of vertices of $\tilde{f}$ with new vertices. If $\{\tilde{v}_{k-1}, \tilde{v}_k\}$ is neighboring edge to $\{\tilde{v}_k, v'_i\}$ in U and $\{\tilde{v}_{k-1}, \tilde{v}_k\}$ is not a supporting edge to $\bar{f}$, then $\{\tilde{v}_k, v'_i\}$ is a pseudo edge or travel edge. If $\{v'_{i-1}, \tilde{v}_k\}$ is neighboring edge to $\{\tilde{v}_k, v'_i\}$ in U, then both $\{v'_{i-1}, \tilde{v}_k\}$ and $\{\tilde{v}_k, v'_i\}$ are pseudo edges. An example is shown in Figure 7.

  Case 2. *$\tilde{f}$ is unshrinkable*

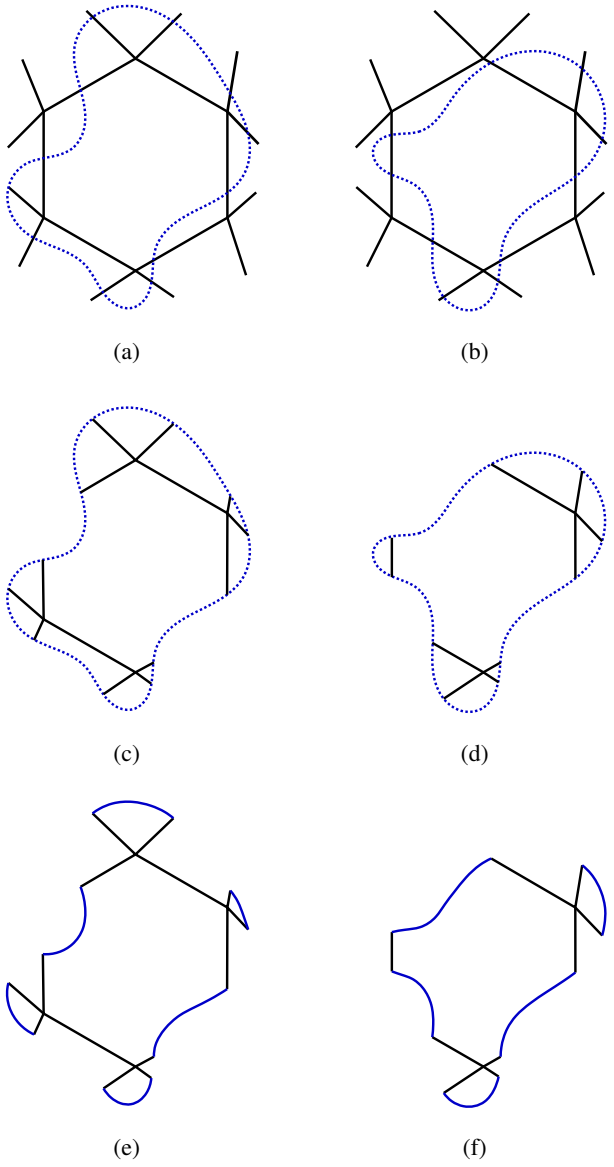

Figure 6: (a) and (b) show one 2-cell of complex $\hat{K}$. Dotted blue lines in (a), (b), (c), and (d) show clipping region $\tilde{R}_j$ of some layer. $\hat{K}$ in (a) and (b) is clipped into disconnected components shown in (c) and (d). They are patched using step 5 with solid blue lines into $\tilde{K}$ in (e) and (f).

If $\tilde{f}$ is not shrinkable, then the edges created to make 2-cell $\tilde{f}$ in step 5 is made of pseudo edges. An example is shown in Figure 8.

Note that If $\hat{K}$ is a dense mesh, then the number of thin 2-cells at the boundary of $\tilde{K}$ will be more since the maximum number of thin boundary 2-cells in $\tilde{K}$ we can have is $m/2$, where m is number of vertices in the S sequence.

3. Euler Transformation for $d = 2$ with Combinatorial changes

Suppose $\hat{f}, \hat{f}_e, \hat{f}_v$ are 3 classes of 2-cells corresponding to a polygon (f), edge (e), and a vertex(v) in K. Maximum mitered offset in a 2-cell 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 $\hat{f}$.

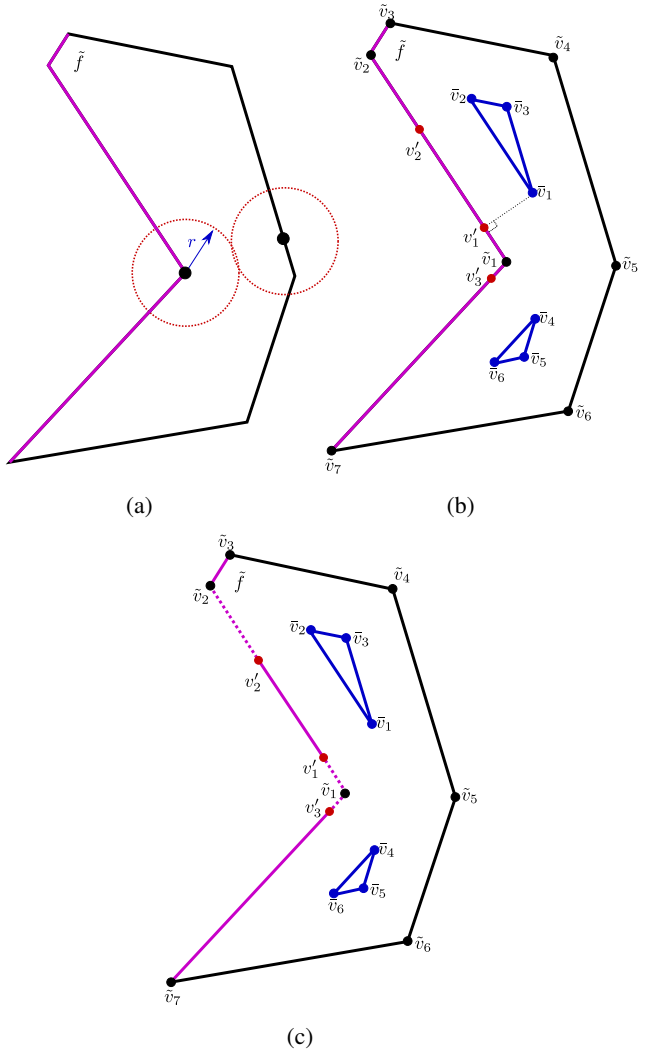

Figure 7: Red dotted circle is extruder cross-section of radius r in (a). $\tilde{f}$ is 2-cell of $\tilde{K}$ created in step 5 shown in (a), (b), (c). $\bar{f}$ is a set of 2-cells created from $\tilde{f}$ (mitered offset), in solid blue in (b), (c). The set of 1-cells created in step 5 is in pink. $\tilde{v}_i \in \{\tilde{v}_1, ..., \tilde{v}_7\}$ of original vertices of $\tilde{f}$ and $v'_i \in \{v'_1, v'_2, v'_3\}$ point of intersection of line through some $\bar{v}_k \in \{\bar{v}_1, ..., \bar{v}_6\}$ and perpendicular to supporting edge in $\tilde{f}$ shown in (b), (c). $\{v'_3, \tilde{v}_1\}$, $\{v'_1, \tilde{v}_1\}$, $\{v'_2, \tilde{v}_2\}$ are pseudo edges in $\tilde{f}$ in (c).

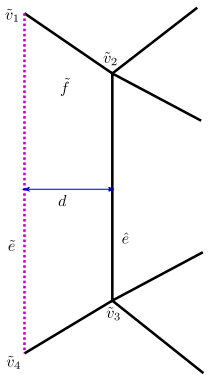

Figure 8: 2-cell $\tilde{f}$ is added into $\tilde{K}$ with creation of edge $\tilde{e}$ in step 5, where $\tilde{e}, \hat{e}$ are parallel with perpendicular distance $d < 2r$. $\tilde{f}$ is unshrinkable with offset distance r. Hence $\tilde{e}$ is pseudo edge.

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 2-cells and if both of those edges are reduced, then $\hat{f}_e$ will collapse into an edge as shown in Figure 9. Since Class 3 2-cells ($\hat{f}_v$) do not share edges with any Class 1 2-cells, combinatorial changes in the Euler transformation will not affect the number of edges in $\hat{f}_v$.

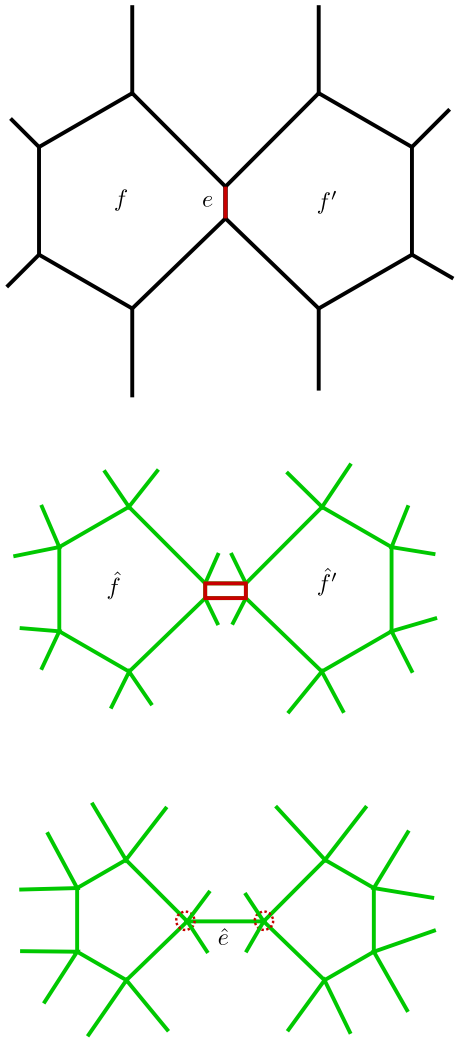

Figure 9: Two 2-cells of a 2-complex K in the plane (top). Euler transformation of K into $\hat{K}$ (middle), where $\hat{f}, \hat{f}'$ are Class 1 2-cells and $\hat{f}_e$ (red) is a Class 2 2-cell corresponding to edge e (red) in K. Class 2 2-cell $\hat{f}_e$ is collapsed to an edge $\hat{e}$ and red circled vertices have odd degree (bottom).

Lemma 3.1. *Let $\hat{v}$ is a vertex in a Class 1 2-cell $\hat{f}$ of $\hat{K}$ created after collapsing π adjacent edges in $\hat{f}$, where $\hat{f}$ is allowed to have combinatorial changes. Let $\hat{f}_e$ is some Class 2 2-cell that contains one of these collapsed adjacent edges. If no $\hat{f}_e$ is collapsed into an edge, then degree of $\hat{v}$ is $2\pi + 4$ else $2\pi + 4 - m$ where m is number of 2-cell similar to $\hat{f}_e$ collapsed into an edge.*

Proof. Since a 2-cell $\hat{f}$ is allowed to have combinatorial changes in $\hat{K}$, it will change degree of vertices in $\hat{f}$. π adjacent edges in

$\hat{f}$ has $\pi + 1$ vertices. Each end vertices of the path created by π adjacent edges add 3 edges to $\hat{v}$, and each interior vertex of the path adds 2 edges to $\hat{v}$ as shown in Figure 10. This implies $\hat{v}$ has degree $2(\pi - 1) + 3 + 3 = 2\pi + 4$ and 1-skeleton of $\hat{K}$ is *still Euler*. If $m > 0$ Class 2 2-cells sharing one of these adjacent edges are allowed to collapse into an edge, then 2 edges sharing $\hat{v}$ of each collapsed Class 2 2-cell is replaced by one edge. Also, $m \leq \pi$ since each distinct edge in any Class 1 2-cell is shared by a unique Class 2 2-cell 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 upon m is even or odd, as shown in Figure 10. □

3.1. Generalized Euler Transformation

Consider a 2-complex K consisting of a single 2-cell f. K does not satisfy the input condition for Euler transformation, since adjacent edges are shared with the outside (Figure 11). Nevertheless, we apply the transformation to K. In the resulting $\hat{K}$, all vertices will have odd degree (circled in Figure 11). But this $\hat{K}$ satisfies the input condition, since any pair of adjacent edges of a 2-cell in K now belong to two new 2-cells in $\hat{K}$. 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, even if adjacent edges of a 2-cell in K are boundary edges. More generally, we define this process as the generalized Euler transformation.

Definition 3.2. *(Generalized Euler Transformation in $d = 2$) Let K be a 2-dimensional cell complex in $\mathbb{R}^2$ with 2-cells possibly having adjacent boundary edges. Apply the Euler transformation on K to obtain $\hat{K}$, whose 1-skeleton may not be Euler. Now apply the Euler transformation to $\hat{K}$ to obtain $\check{K}$. The 1-skeleton on $\check{K}$ is guaranteed to be Euler.*

Notice that the density of the mesh is increased by this process. Hence 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. In the following result, 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 3.3. *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 2-cells 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 2-cells share an edge with any collapsed Class 2 2-cells $\hat{f}_e$ is in $\hat{k}_j$. Hence $\hat{v}$ can be shared by some Class 3 2-cell not in $\hat{k}_j$. If any Class 3 2-cell sharing $\hat{v}$ is not contained in $\hat{k}_j$, this Class 3 2-cell does not share an edge with any Class 2 2-cell. Hence such cells do not belong to any component $\hat{k}_j$ in $\hat{k}$. Since Class 1 2-cells are edge disjoint from any Class 3 2-cells, and there are some Class 3 2-cells and 1 Class 1 2-cell ($\hat{f}$) 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

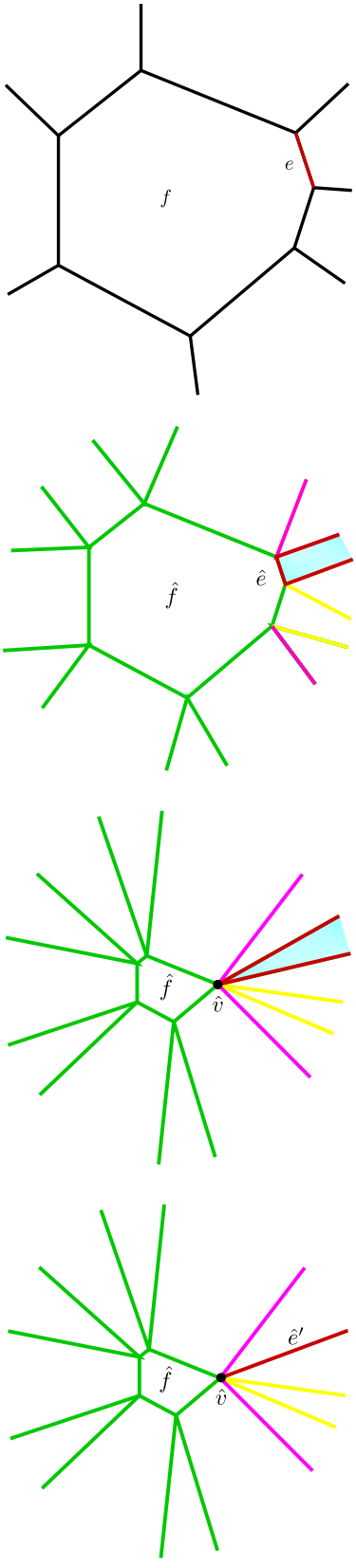

Figure 10: f is 2-cell in 2-complex K (top). Euler transformation of K into $\hat{K}$ (second), where $\hat{f}$ is the Class 1 2-cell corresponding to f, and $\hat{f}_e$ (blue) is the Class 2 2-cell corresponding to edge e. Combinatorial changes are allowed in $\hat{K}$ in the third figure, and edge $\hat{e}$ 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 figure, an edge of $\hat{f}_e$ shared with some Class 1 2-cell 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 1-skeleton of $\hat{K}$ is $2(2) + 4 - 1 = 7$.

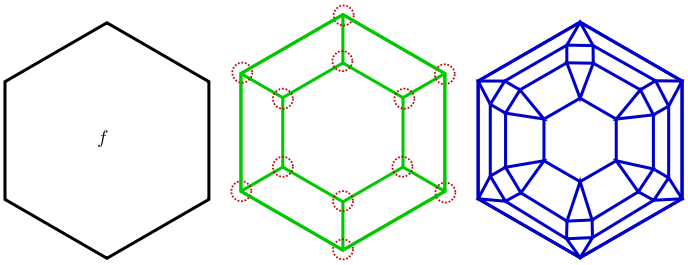

Figure 11: $\hat{K}$ consists of 2-cell f (left), 2-complex $\hat{K}$ in green is Euler transformation of $\tilde{f}$(middle), where red circled vertices have odd degree, 2-complex in blue is Euler transformation of $\hat{K}$ and its 1- skeleton is Euler (right).

in $\hat{k}_j$. Since Class 1 and Class 3 2-cells are edge disjoint in $\hat{K}$, then any vertex($\hat{v}'$) in some Class 3 2-cell in $\hat{k}_j$ not similar to $\hat{v}$ has 2 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 Figure 12.

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 2 more edges not contained in $\hat{k}$. Hence the 1-skeleton of $(\hat{K} \setminus \hat{k}) \cup \bar{k}$ is Euler. □

Let any vertex $\hat{v}$ in the sub 2-complex $\tilde{k}_i$ of single component $\hat{k}_j$ in $\hat{k}$ has odd number of edges connected to it in $\tilde{k}_j$. We apply generalized Euler transformation to any such sub 2-complex $\tilde{k}_i$ for any component $\hat{k}_j$ in $\hat{k}$. Then by Lemma 3.3, the new 2-complex is Euler.

4. Tool Path Algorithm

Since the 1-skeleton of $\tilde{K}$ is Euler, we can construct a tool path that consists only of the print path except some boundary edges of $\tilde{K}$ where we can have travel paths if cells are too thin to prevent *material collision* and avoid print path *crossover*. Since tool path mainly consists of print path it will improve quality of print, reduce print time.

4.1. Cycle Tree

Algorithm 1 CycleTree

1: *Unmark all the edges in* $\tilde{K}$
2: *CycleList = FindBoundaryCycles($\tilde{K}$, ∅)* ▷ Initial CycleList contains one cycle(C_0)
3: *Mark all the edges in* $\tilde{K}$ *of* C_0
4: *pred(C_0) = 0* ▷ Predecessor of C_0 is empty
5: **while** *CycleList* ≠ ∅ **do**
6: C = *CycleList.pop(0)*
7: *Cycles = FindBoundaryCycles($\tilde{K}$, C)* ▷ Returns list of Cycles
8: *CycleList = CycleList* ∪ *Cycles*
9: **while** *Cycles* ≠ ∅ **do**
10: C_s = *Cycles.pop(0)*
11: *pred(C_s)= C* ▷ predecessor of C_s is C
12: *Mark all the edges in* $\tilde{K}$ *of* C_s

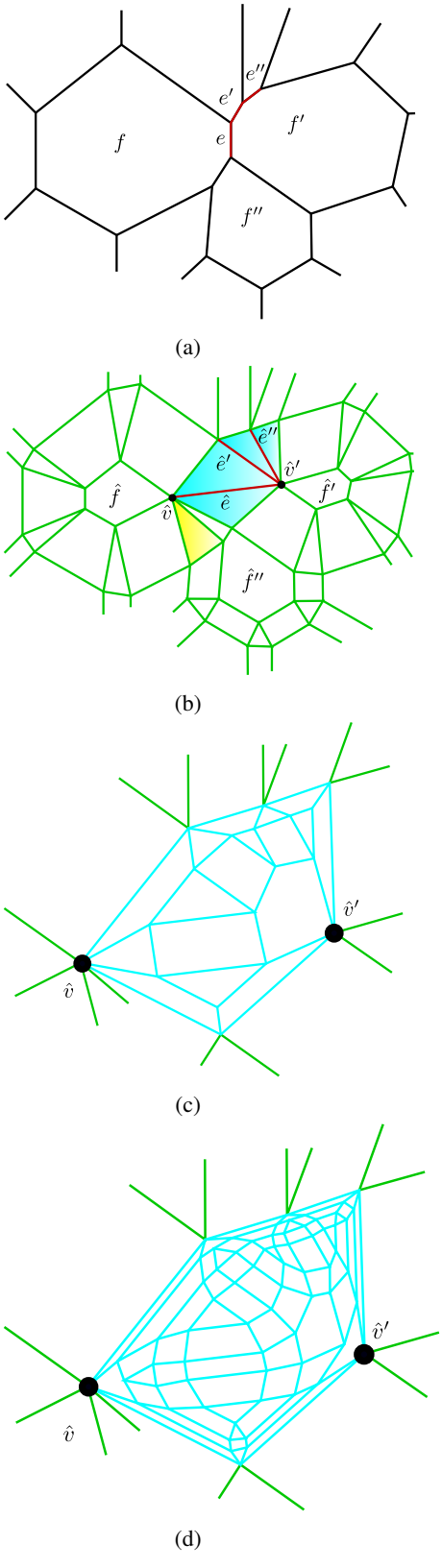

Figure 12: (a) f, f', f'' are 2-cells in 2-complex K. (b) 2-complex $\hat{K}$ after Euler transformation with combinatorial changes to some 2-cells in $\hat{K}$. Class 2 2-cells $\hat{f}_e, \hat{f}_{e'}, \hat{f}_{e''}$ corresponding to e, e', e'' in K are collapsed into edges $\hat{e}, \hat{e}', \hat{e}''$. Sub 2-complex $\tilde{k}_i$ (in blue) of some $\hat{k}_j$ in $\hat{K}$ consists of Class 3 2-cells 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, $\hat{v}$ has 1 Class 3 2-cell(yellow) not contained in $\tilde{k}_i$. (c),(d) shows generalized Euler transformation of $\tilde{k}_i$. (d) $\check{k}_i$(blue) is generalized Euler transformation of $\tilde{k}_i$ and $\hat{v}, \hat{v}'$ including other vertices of $\check{k}_i$ has even degree in 1-skeleton of $\hat{K} \setminus \tilde{k}_i \cup \check{k}_i$.

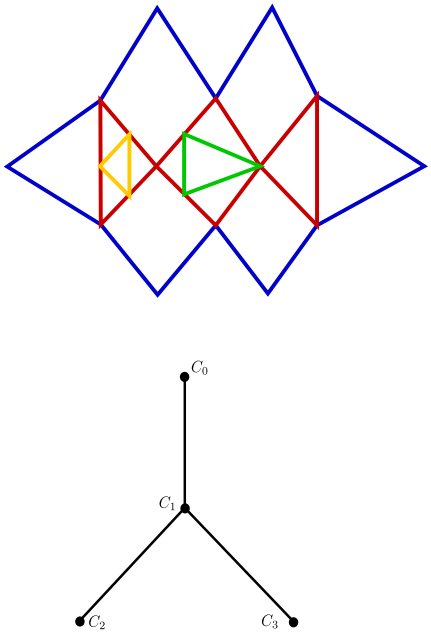

Figure 13: $\tilde{K}$(left) has C_0(blue), C_1(red), C_2(yellow), C_3(green) cycles of cycle tree. Cycle tree(right) where union of C_0, C_1, C_2, C_3 is in 1-skeleton of $\tilde{K}$, child cycles C_2, C_3 of C_1 are disjoint and $|C_0| \supset |C_1| \supset |C_2| \supset |C_3|$.

Algorithm 2 FindBoundaryCycles($\tilde{K}$, C)

Find Collection E and Adjacency list A based on C
Cycles = {}
while $E \neq \emptyset$ **do**
    Cycle = {}
    Use Hierholzer's algorithm to find a Euler circuit(Cycle)
    Remove all edges of Cycle from E
    Cycles = *Cycles* $\cup$ {*Cycle*}
return Cycles

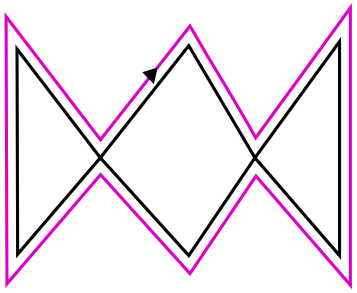

Figure 14: C_1 (black) of cycle tree in Figure 13 is an Euler Circuit, and a clockwise ordered sequence of vertices in C_1 shown in pink.

Algorithm 1 constructs cycle tree. It starts with finding outermost cycle in $\tilde{K}$ to inner most cycles in $\tilde{K}$, where outermost cycle is a root, inner most cycles are leaf vertices and union of all cycles in cycle tree is $\tilde{K}$. Every vertex in the Cycle tree represents a cycle and cycle can be an Euler Circuit. Predecessor of C (pred(C)) is connected with C with atleast one vertex. All interior vertices in $\tilde{K}$ have even degree with at least degree 4 , since every vertex in 1-skeleton of $\tilde{K}$ has degree 4, if initial conditions are satisfied in K and some vertices may have even degree at least 6 due to local euler transformation of subcomplex $\hat{k}$ mentioned in Section 3. Some vertices at the boundary of $\tilde{K}$ may have degree 2 as mentioned in step 5 of Section 2.2 but there is at least one boundary vertex of $\tilde{K}$ with at least degree 4 if $\tilde{K}$ has interior edges, since $\tilde{K}$ is connected. This implies every parent cycle(C) can have at most m consecutive descendants on a path in a tree, where $2m$ is maximum degree of vertices of C in 1-skeleton of $\tilde{K}$ since we can have $2t \leq 2m$ edges at $\tilde{v}$ belongs to same cycle in case cycle is an Euler circuit. Based on construction of cycle tree, any two cycles on cycle tree are disjoint if they are not on same path starting at root cycle on cycle tree. An example of cycle tree is shown in Figure 13.
Suppose $\tilde{f}$ is a 2-cell in $\tilde{K}$ sharing edges $\tilde{e}$ in C (cycle) and none of its edges are marked in $\tilde{K}$, except $\tilde{e}$. Then all other edges of $\tilde{f}$ except $\tilde{e}$ are part of successor cycles in the tree. Let E is collection of all the edges in all 2-cells similar to $\tilde{f}$ except edges in C for algorithm 2 . If C is empty then, E is collection of all the boundary edges in $\tilde{K}$.

4.2. Traversal

Let vertex $\tilde{v}$ is vertex in cycle C_1 is shared by more than one cycle in the tree and has a degree $2m$. Then there are $q < m$ cycles sharing vertex $\tilde{v}$. According to cycle tree construction, we can assume underlying space in R^2 of these cycles follows, $|C_q| \subset |C_{q-1}|$,..., $|C_2| \subset |C_1|$. Then there exists a path $p = \{C_1, C_2, ..., C_q\}$ on the cycle tree sharing vertex $\tilde{v}$ and C_1 ancestor of all the cycles in p, C_q is descendent of all the cycles in p. *Assuming* without loss of generality, traversal of edges of root cycle in $\tilde{K}$ is clockwise.*Assuming* edges in $\tilde{K}$ of any cycle C_i of the cycle tree is traversed such that there is no subpath *crossover* in case C_i is an Euler circuit as shown in Figure 14. To prevent *crossover* of subpaths, while traversing edges of $\tilde{K}$ in any C_i and *pred*(C_i) of cycle tree should have *opposite orientation*. This implies in case q is odd then traversal of edges of C_q is clockwise else counterclockwise. It also implies that if we assume traversal orientation for the root cycle in the tree, then orientation for other cycles in the tree is uniquely determined. Without loss of generality, let traversal of edges in C_1 is clockwise ordered.

1. Start with adding edge traversal restrictions in $\tilde{K}$.

   Case 3. $q = 2$

   Let $(\tilde{e}, \tilde{e}')$, $(\tilde{e}'', \tilde{e}''')$ are pair of clockwise ordered pair of adjacent edges on clockwise cycle paths of C_1 and C_2, sharing vertex $\tilde{v}$. Add restriction on edges traversal in $\tilde{K}$ of cycles C_1, C_2. After traversing edge $\tilde{e}$ in $\tilde{K}$ of cycle C_1 traverse $\tilde{e}''$ of cycle C_2 and after traversing edge $\tilde{e}'''$ traverse $\tilde{e}'$. Mark edge $\{C_1, C_2\}$ on cycle tree. An example of $q = 2$ is shown in Figure 15.

   Case 4. $q > 2$

   Let $(\tilde{e}_{2j-1}, \tilde{e}_{2j})$ is pair of clockwise ordered adjacent edges on a clockwise cycle path of C_j, sharing vertex $\tilde{v}$. Add restriction on edge traversal in $\tilde{K}$ of cycles $C_1, ..., C_q$. Let $\tilde{e}_i \longrightarrow \tilde{e}_j$ means once we traversed edge $\tilde{e}_i$ traverse edge $\tilde{e}_i$ in $\tilde{K}$.Add edge traversal restriction in$\tilde{K}$ if q is odd $\tilde{e}_1 \longrightarrow \tilde{e}_{2q}, \tilde{e}_{2q-1} \longrightarrow \tilde{e}_{2q-3}, \tilde{e}_{2q-2} \longrightarrow \tilde{e}_{2q-4}, ..., \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$ else $\tilde{e}_1 \longrightarrow \tilde{e}_{2q-1}, \tilde{e}_{2q} \longrightarrow \tilde{e}_{2q-2}, \tilde{e}_{2q-3} \longrightarrow \tilde{e}_{2q-5}, ..., \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$. Mark all the edges of the cycle tree on path p. An example of $q = 3, 4$ is shown in Figure 16.

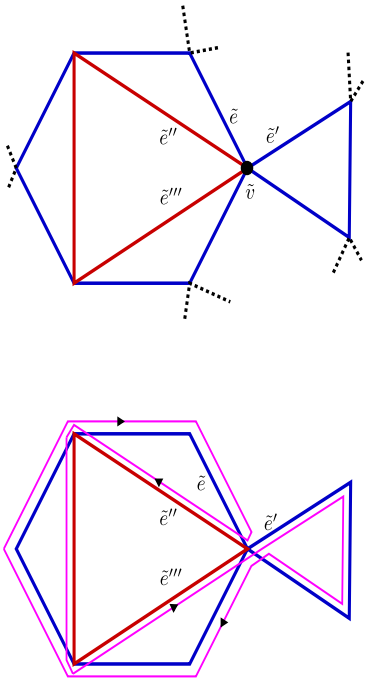

Figure 15: C_1 (blue) and C_2 (red) cycles of the cycle tree in $\tilde{K}$ (top figure). C_1 is the predecessor of C_2 and $\tilde{v}$ is shared only by cycles C_1, C_2. Traversal of edges (pink) in C_1 and C_2 cycles in the 1-skeleton of $\tilde{K}$ with no crossover (bottom figure).

2. Keep on following step 1, until all the edges are marked in cycle tree.
3. Start with traversing edges of root cycle in clockwise order and follow traversal restrictions. Stay on same cycle until next edge traversal restriction.

If any two child cycle(C_2, C_3) of a parent cycle(C_1) on a cycle tree has same clockwise ordered pair of adjacent edges of parent cycle. This implies C_2, C_3 are same child cycle since any two cycles in a cycle tree do not share edges. Hence we have at most one edge common between two clockwise pair of adjacent edge of a parent cycle (C_1) at $\tilde{v}, \tilde{v}'$ and we can pick any $\tilde{v}, \tilde{v}'$ on parent cycle for edge restrictions without any conflict of edge restrictions for edge traversal in $\tilde{K}$ since common edge is out-edge for $\tilde{v}$ and in-edge for $\tilde{v}'$ as shown in Figure 17.

Since we are traversing edges along the cycle, until we traverse an edge with restriction. Edge restriction allows to traverse edges between cycles with no crossovers. Hence edge traversal in $\tilde{K}$ has *no crossover*.

5. Discussion

We have described a complete framework for continuous tool path planning in layer-by-layer 3D printing. The computational complexity of our framework will be decided by the Euler transformation steps, the details of which we address briefly in the conjoint submission. The clipping step will be bottlenecked by the computation of intersection of the Euler transformed complex with each polygon in each layer.

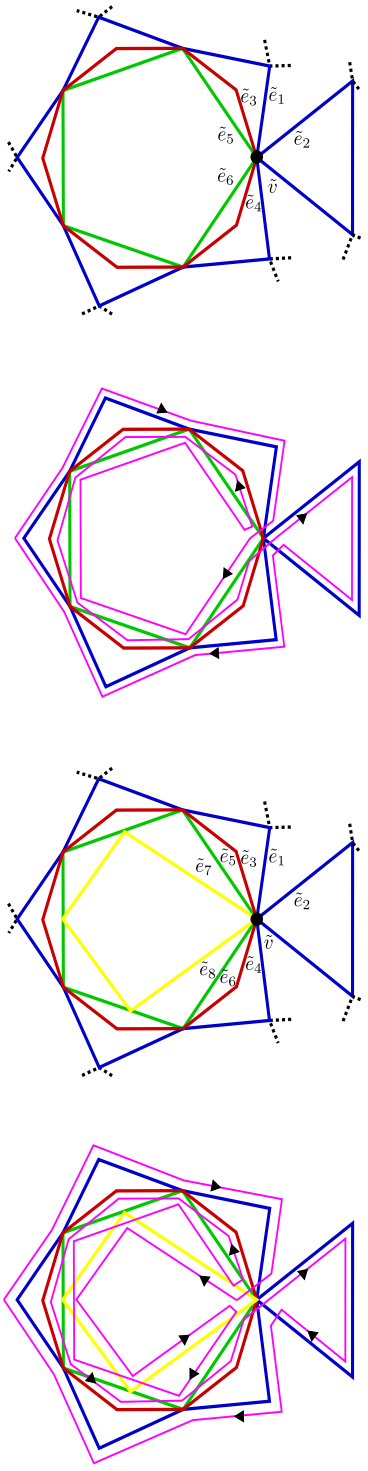

Figure 16: For $q = 3$, C_1 (blue), C_2 (red), and C_3 (green) are the only cycles in a cycle-tree sharing vertex $\tilde{v}$, where C_1 is an ancestor and C_3 is a descendant of all cycles in the path $p = \{C_1, C_2, C_3\}$ on the cycle tree (top). Traversal (pink) of edges in $\tilde{K}$ of cycles C_1, C_2, C_3 with no crossover where $\tilde{e}_1 \longrightarrow \tilde{e}_6, \tilde{e}_5 \longrightarrow \tilde{e}_3, \tilde{e}_4 \longrightarrow \tilde{e}_2$ are edge traversal restrictions is shown next. Traversal of edges in C_1, C_3 is clockwise (second figure) for $\tilde{K}$. Third figure shows a $q = 4$ case, with C_1 (blue), C_2 (red), C_3 (green), and C_4 (yellow) being the only cycles in a cycle tree sharing vertex $\tilde{v}$, where C_1 is an ancestor and C_4 is descendant of all cycles in the path $p = \{C_1, C_2, C_3, C_4\}$ on the cycle tree. Bottom figure shows traversal (pink) of edges in $\tilde{K}$ of cycles 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 counter-clockwise for $\tilde{K}$.

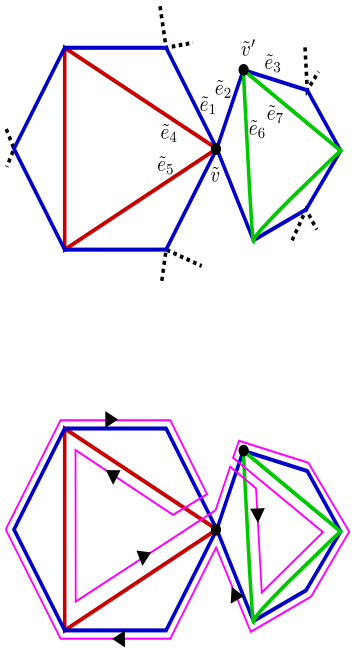

Figure 17: C_1 (blue), C_2 (red), and C_3 (green) cycles where C_1, C_2 are child of C_1 in the cycle tree, with C_1, C_2 and C_2, C_3 sharing vertex $\tilde{v}, \tilde{v}'$, respectively (top). Bottom figure shows edge traversal restrictions $\tilde{e}_1 \longrightarrow \tilde{e}_4, \tilde{e}_5 \longrightarrow \tilde{e}_2, \tilde{e}_2 \longrightarrow \tilde{e}_6, \tilde{e}_7 \longrightarrow \tilde{e}_3$. Traversal (pink) of edges in C_1, C_2, and C_3 shown has no conflict with the edge restrictions, since $\tilde{e}_2$ is an out-edge at $\tilde{v}$ and an in-edge at $\tilde{v}'$.

We have generalized the Euler transformation defined in the conjoint submission 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 the results we presented here should generalize to the non-planar realm as well, as long as underlying support is guaranteed by the design.

References

[1] J. Brennan-Craddock, D. Brackett, R. Wildman, R. Hague, The design of impact absorbing structures for additive manufacture, Journal of Physics: Conference Series 382 (1) (2012) 012042.
URL http://stacks.iop.org/1742-6596/382/i=1/a=012042

[2] J. Wu, N. Aage, R. Westermann, O. Sigmund, Infill optimization for additive manufacturing—approaching bone-like porous structures, IEEE Transactions on Visualization and Computer Graphics 24 (2) (2018) 1127–1140. doi:10.1109/TVCG.2017.2655523.

[3] B. Garret, B. Redwood, F. Schöffer, The 3D Printing Handbook: Technologies, design and applications, 3D Gubs, 2017.

[4] E. Galceran, M. Carreras, A survey on coverage path planning for robotics, Robotics and Autonomous Systems 61 (12) (2013) 1258–1276. doi:10.1016/j.robot.2013.09.004.

[5] L. Xu, Graph planning for environmental coverage, Ph.D. thesis, Carnegie Mellon University, Pittsburgh, PA (August 2011).

[6] Z. L. Cao, Y. Huang, E. L. Hall, Region filling operations with random obstacle avoidance for mobile robots, Journal of Robotic Systems 5 (2) (1988) 87–102.
URL https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.4620050202

[7] D. Ding, Z. Pan, D. Cuiuri, H. Li, A practical path planning methodology for wire and arc additive manufacturing of thin-walled structures, Robotics and Computer-Integrated Manufacturing 34 (2015) 8–19. doi:https://doi.org/10.1016/j.rcim.2015.01.003.
URL http://www.sciencedirect.com/science/article/pii/S0736584515000162

[8] Y. Jin, Y. He, J. Du, A novel path planning methodology for extrusion-based additive manufacturing of thin-walled parts, International Journal of Computer Integrated Manufacturing 30 (12) (2017) 1301–1315. arXiv:https://doi.org/10.1080/0951192X.2017.1307526, doi:10.1080/0951192X.2017.1307526.
URL https://doi.org/10.1080/0951192X.2017.1307526

[9] G. Dreifus, K. Goodrick, S. Giles, M. Patel, R. M. Foster, C. Williams, J. Lindahl, B. Post, A. Roschli, L. Love, V. Kunc, Path optimization along lattices in additive manufacturing using the chinese postman problem, 3D Printing and Additive Manufacturing 4 (2) (2017) 98–104. arXiv:https://doi.org/10.1089/3dp.2017.0007, doi:10.1089/3dp.2017.0007.
URL https://doi.org/10.1089/3dp.2017.0007