Volume xx (200y), Number z, pp. 1–12

Curvy: An Interactive Design Tool for Varying Density Support Structures

E. Ulu[1] N. Gecer Ulu[1] J. Li[2] and W. Hsiao[1]

[1]Palo Alto Research Center, USA
[2]UCLA HCI Research, USA

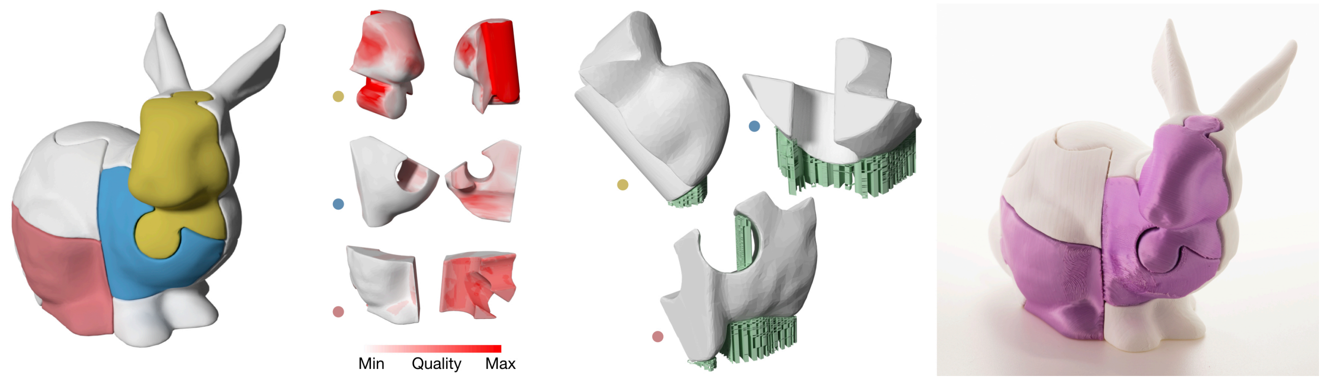


Figure 1: *Curvy enables users to design support structures implicitly, by providing high-level surface quality preferences (middle-left) directly on the target object (left). User preferences are translated into low-level support parameters to generate varying density curvy zigzag supports (middle-right). Resulting prints satisfy perceptual and functional intents of the user (right). Puzzle pieces fit together well while visually important surfaces are free of support contact marks. In the middle figures, puzzle pieces are marked with colors corresponding to the ones on the left model.*

Abstract

We introduce Curvy–an interactive design tool to generate varying density support structures for 3D printing. Support structures are essential for printing models with extreme overhangs. Yet, they often cause defects on contact areas, resulting in poor surface quality. Low-level design of support structures may alleviate such negative effects. However, it is tedious and unintuitive for novice users as it is hard to predict the impact of changes to the support structure on the final printed part. Curvy allows users to define their high-level preferences on the surface quality directly on the target object rather than explicitly designing the supports. These preferences are then automatically translated into low-level design parameters to generate the support structure. Underlying novel curvy zigzag toolpathing algorithm uses these instructions to generate varying density supports by altering the spacing between individual paths in order to achieve prescribed quality. Combined with the build orientation optimization, Curvy provides a practical solution to the design of support structures with minimal perceptual or functional impact on the target part to be printed.

CCS Concepts

• ***Theory of computation*** → ***Computational geometry;*** • ***Human-centered computing*** → ***Interactive systems and tools;*** • ***Applied computing*** → *Computer-aided design;*

1. Introduction

In many 3D printing approaches, support structures play an essential role in successful printing by *supporting* the overhang regions to prevent them from collapsing under gravity. While they may be necessary, these auxiliary structures often damage the surface quality resulting in visual as well as functional artifacts. Such artifacts are often unavoidable especially in widely used single material fused filament fabrication (FFF) process as there is a delicate balance between the amount of supports and the resulting surface quality. When supports are used excessively, they stick to the walls of the models and leave blemishes on the surface. On the other hand, when insufficient, sagging occurs between the support contact points due to the large bridging distance. Many 3D printing software, such as Autodesk MeshMixer [Aut20], Simplify3D [Sim20], Ultimaker Cura [Ult20] and Slic3r [Pru20], provide automated means to design support structures. However, without user involvement, resulting supports often lead to suboptimal surface quality even for simple shapes. User involvement required to alleviate the quality issues, on the other hand, is generally very low-level and tedious, such as manual placement of individual support pillars, local adjustments of support patterns, its orientation and spacing. Although such adjustments are attainable for experienced designers, a more intuitive and direct way of support design is required for novice users.

We present Curvy–an interactive tool to design support structures with minimal perceptual and functional impact on the object. Curvy takes as input users' high-level specifications on the desired surface quality and produces support toolpaths for each layer accordingly (Figure 1). As the user input is defined *directly* on the surface of the target model rather than the supports, users do not need to have a low-level knowledge on inner workings of the support generation algorithm to predict the impact of changes they are making on the final printed result.

Main challenge in such a direct approach lies in the translation of the high-level inputs on the target object to the low-level support design parameters. For such a transition to be possible, underlying support parametrization needs to be sufficiently flexible to comply with any arbitrary input whereas general enough to be applicable to variety of shapes. Additionally, resulting supports are required to exhibit common properties including easy removal and low additional cost to print.

Our approach overcomes this challenge using a novel toolpathing approach that *(i)* generates curvy zigzag paths conforming to the shape boundaries as well as the infill patterns of each layer and *(ii)* controls the spacing between individual paths (*i.e.*, density) locally. The former capability allows us to obtain easy to remove supports while consistently providing sufficient amount of contact points for successful bridging. The latter one, on the other hand, enables local control of the surface quality per user prescriptions on the target object. Combined with a build direction optimization approach, unnecessary supports as well as supports touching the regions that are intended to have high surface quality are minimized.

Our main contributions are:

- a novel curvy zigzag toolpathing algorithm that results in high surface quality while allowing easy removal,
- a method to manipulate support density locally,
- an interactive support structure design tool that allows users to define high-level preferences directly on the target object using above two ideas.

2. Related Work

Curvy aims to provide end-users an interactive way to design support structures implicitly by defining their high-level surface quality preferences directly on the target object. This goal cross-cuts three areas of prior work in 3D printing: *(i)* computational tools for design and process planning, *(ii)* support structures generation methods and *(iii)* build orientation optimization approaches.

2.1. Design and Process Planning Tools

As simple as it may seem, 3D printing is a complex process with many aspects that require low-level design and process planning for a successful operation. In order to make it more accessible to novice users and more convenient to experienced users, past research has explored variety of interactive tools targeting many different aspects of 3D printing. Recent examples include two-piece mold design [NAI*18], modeling through augmented reality [PBW*18], patching for minimum waste design iterations [TMG*15] and deformable object design [HPL*19]. Similar to these approaches, Curvy aims to alleviate the amount of low-level design users need to perform for a successful 3D print. In particular, Curvy focuses on making design of support structures easy and intuitive by eliminating the need for explicit manipulation of support structures. Instead, the support structures are designed implicitly, by specifying the high-level quality preferences directly on the target object.

Other approaches focus on providing optimal slicing to improve the surface quality or geometric accuracy. Recent examples of such approaches include adaptive [WCT*15; AHL17] and curved slicing [ERP*19] schemes. These methods improve the quality of the prints by mainly alleviating the so-called staircase effect. Our approach is complementary to these tools in that presented slicing schemes may be facilitated in printing the target object to further improve the surface quality while Curvy is minimizing the impact of supports on the final result.

2.2. Support Structure Generation

Generating supports is often composed of two main steps: *(i)* detection of surfaces requiring supports (*i.e.*, overhangs), and *(ii)* design of the support structure itself. For detection of overhangs, Chalsani *et al.* [CJR95] performed a boolean difference between two successive slices while Kirschman *et al.* [KJB*91] and Allen *et al.*[AD95] considered down-facing facets of the input mesh having angle too steep to print correctly as overhangs. In our build orientation optimization, we employ the latter as it is computationally more practical while being sufficiently accurate.

While commercial 3D printing software such as Simplify3D [Sim20], Ultimaker Cura [Ult20] and Slic3r [Pru20] provide general purpose space filling support structures in the form of regular patterns such as zigzags or concentric, past research has explored various specialized approaches targeting cost reductions in 3D printing. Sloped wall supports [HYW*09], tree-like space efficient approaches [SU14; Aut20; VGB14], bridge supports [SDLW16] and scaffolding style support structures [DHL14] have been explored. While these work focus on reducing the build time and amount of material used to print supports, Curvy has an emphasis on enhancing the overall surface quality of the resulting print.

In addition to aforementioned work on design of external support structures, variety of approaches has focused on developing internal

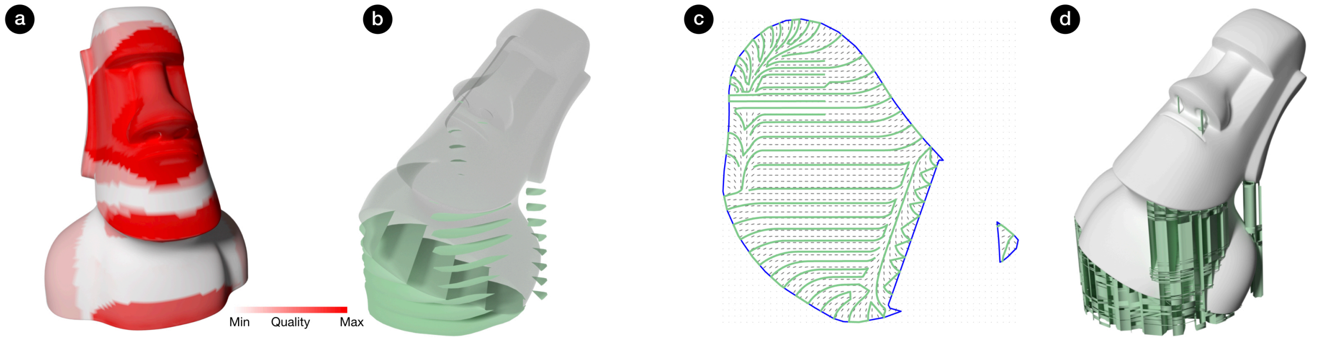


Figure 2: *Given an input 3D geometry in an arbitrary initial orientation and users' preferences in desired surface quality (a), Curvy optimizes the build orientation to avoid supports in high quality regions as much as possible (b) and generates variable density curvy zigzags in each support polygon (c) to create a final support structure (d). Green represents support polygons in (b) and generated support toolpaths in (c-d).*

supports for printing hollow objects [HLDC16; HL18; WWZW16; LL17]. While not our primary focus, our approach is inherently capable of generating internal support structures the same way it generates the external ones.

2.3. Build Orientation Optimization

Effects of build orientation on build time and cost [AKL07; AAD98], mechanical properties [UKY*15], surface roughness [DTS16; WZK16], manufacturability [UGHN20] as well as the support structure [EME15] and its impacts on the target object [ZLP*15] have been studied extensively and automated means to select the *best* orientation are proposed to minimize such directional biases. Instead of selecting a single best orientation, other approaches use robotic printing platforms to manipulate the build orientation actively during the print process in order to avoid the need for support structures [WDF*17; WDF*19; DWW*18; GZN*15; XLCT19]. In our approach, we select a single build orientation that minimizes amount of support contact on surfaces that are intended to exhibit high surface quality when printed.

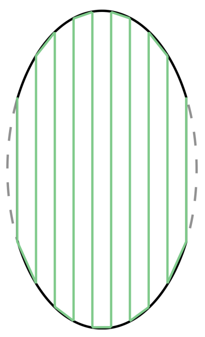
Among all, Curvy is closest to [ZLP*15] that build orientation is adjusted to avoid support structures touching the perceptually important surfaces. However, our approach incorporates individual user's preferences rather than completely relying on a single generalized model. Curvy uses saliency map only as a starting point to guide the users in their selection. This approach allows Curvy to mitigate not only perceptual but also functional impact of supports on the target object.

3. Curvy Support Design

3.1. Overview

Figure 2 illustrates the overview of our interactive support structure design process. Given an input 3D geometry in an arbitrary initial orientation, the user defines their preferences on desired surface quality by simply painting on the surface of the object. Then, Curvy optimizes the build orientation *(i)* to avoid the support requirement at regions that are depicted to be high quality by the user as much as possible and *(ii)* to minimize the support contact area everywhere else. At the optimum orientation, the volume that is needed to be filled with support structure is computed and sliced into layers to generate *support polygons* (green in Figure 2(b)). Note that layers here corresponds to layers/slices in 3D printing process. Then, each support polygon is filled with curvy zigzag toolpaths. Spacing between the individual toolpaths (*i.e.*, density) is adjusted based on user preferences on the surface quality. For higher quality surface regions, spacing is decreased in corresponding parts of support polygons below it. Smaller spacing between support toolpaths provide a higher quality bridging, thereby resulting in better surface quality. On the other hand, it increases the amount of material used for supports as well as the overall print time. To mitigate such effects in printing cost, density of supports are reduced in parts of support polygons corresponding to lower quality requirement areas. For example, note the density difference between the top left part of the slice and the rest of it in Figure 2(c). Finally, Curvy compiles a machine instruction (gcode) file by accumulating all the support toolpaths from each layer together with the toolpaths required to print the target object. To illustrate the resulting support structure as a whole, we construct a 3D model from the generated toolpaths in Figure 2(d).

Our motivation for generating curvy zigzags comes from two major observations in FFF: *(i)* for high quality perimeter prints, a *good* bridging distance should be maintained at the polygon boundaries and *(ii)* for easy removal, the support toolpaths should be perpendicular to both perimeter and infill toolpaths as much as possible. The former property comes from the fact that sagging will occur when the bridging distance between two supporting contact points is large. The latter one is due to the adhesion characteristics between the support and object layers. When the support paths are parallel to the infill or perimeter paths, the adhesion area increases, resulting in stronger bonding between them. The most commonly used regular zigzag support pattern does



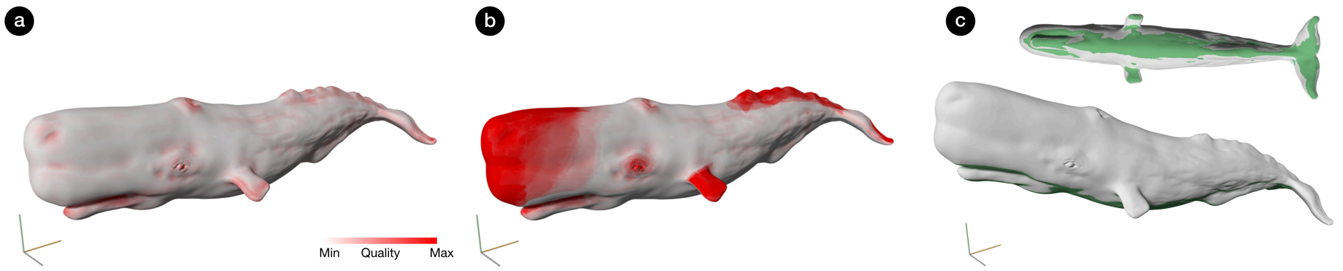


Figure 3: *Overview of how user input is collected. As a starting point, mesh saliency is provided (a). User paints over it to convey their surface quality preferences (b). Curvy then optimizes the build orientation (c). Overhanging regions that require support are highlighted in green in bottom view inset.*

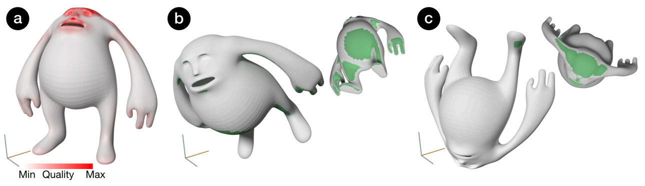


Figure 4: *Effect of build orientation optimization weight. User preference (a) and corresponding optimum build orientation considering only user preference,* $\omega = 1$ *(b) and only support area,* $\omega = 0$ *(c). Overhang regions that require support are highlighted in green in bottom view insets.*

not satisfy these properties as the toolpaths are oriented in a predetermined direction. For an arbitrary shape boundary or arbitrary infill paths, this direction may become parallel or close to parallel to them, leaving some parts unsupported (parts of the ellipse perimeter shown as dashed) or resulting in very strong adhesion. Yet, both of these properties directly affect the surface quality as the perimeters mainly constitute the visible surface of the print and post-processing artifacts are often alleviated when the adhesion between the supports and the object is lower. Our curvy zigzag toolpaths demonstrate both of these properties by complying to a bidirectional field (Figure 2(c)) governed by the boundary of the polygon as well as the predetermined infill pattern.

3.2. Collecting User Preferences

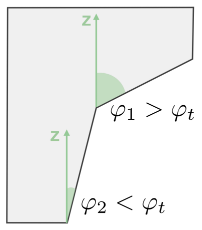
At the heart of Curvy lies the idea of implicit support design through collecting user preferences *directly* on the target object rather than explicitly designing supports. Figure 3 provides an overview of user interaction and how user input is incorporated into support structure generation process. We start by computing mesh saliency on the input object such that salient features correspond to regions that require high surface quality Figure 3(a). Then, the user paints over the saliency information expressing quality specifications for their own preferences and functional requirements Figure 3(b). This quality information is, then, used to find the optimum build orientation such that support contacts are minimized avoiding high quality regions as much as possible. We store user preference as a scalar field $Q \in [0,1]$ defined on the vertices of the object mesh representing *quality* requirements. Then, Q is mapped to individual slices later during the support generation process where denser supports are created for regions with higher quality requirements.

We compute mesh saliency as presented in [LVJ05]. The saliency computation utilizes curvature information on the input mesh and identifies visually interesting regions that are likely to be perceptually important. While perceptual saliency methods excel in detecting facial features such as eyes and nose, mesh saliency approaches often lack in capturing functionally important features that are crucial for 3D printed objects. For this reason, we allow users to paint over mesh saliency to express functional considerations. In that sense, saliency information maybe used as a starting point and guide users' preference. On the other hand, saliency information may partially or fully be removed on the regions that are not visually or functionally important to the user.

3.3. Selecting the Build Orientation

Contact surfaces where support structures touch the target object often have poor quality due to imperfect bridging or adhesion. Here, we present an optimization method to select a build orientation that results in minimal contact area at the regions designated to be high quality by the user.

To determine if an overhang area requires support to be printed, how much the overhang tilts, φ from the build direction, z is measured and compared to a threshold overhang angle, φ_t. If $\varphi > \varphi_t$, the overhang requires support structures. Using this principle, an object can be oriented to minimize overhang surface area that require support structures. Instead of simply minimizing the contact surface area, in this work, we incorporate user preferences to the objective function using the quality field, Q.



Since overhangs are defined on mesh faces, we map Q that is defined on the mesh vertices to mesh faces by averaging over vertices of each face, $q_j = \sum_k^{nv} Q_k / n_V$. Here, q_j is the averaged quality value of face j, n_V is the number vertices on the face, and k is the vertex

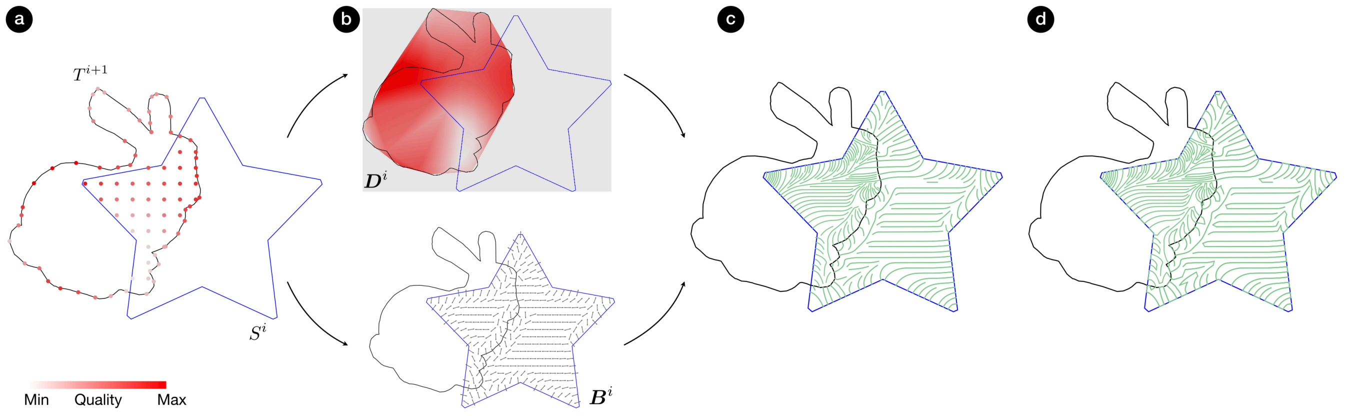


Figure 5: *Given the support polygons and corresponding slice of the target object with the quality preferences (a), we construct a scalar density field (b-top) and a bidirectional field (b-bottom). We then integrate the bidirectional field to generate a set of streamlines. The spacing between the individual streamlines is dictated by the density field (c). Connected streamlines constitute the curvy zigzag toolpaths for this particular slice of the support structure (d).*

index. Then, we formulate our cost function as

$$\kappa_j(\theta) = \begin{cases} 0, & \text{if } \varphi(\theta) \leq \varphi_t \\ (1-\omega)A_j + \omega A_j q_j^{1/p}, & \text{otherwise.} \end{cases} \tag{1}$$

where $\theta = [\alpha, \beta]$ represents rotations around z and x axes such that vertices of the rotated mesh can be calculated as $V_r = R_x(\beta)R_z(\alpha)V$. Here, R_z and R_x are rotation matrices, V is a matrix storing vertex positions, $A_j \in [0,1]$ is normalized area of face j, $p > 0$ is a penalization factor and $\omega \in [0,1]$ is a weight parameter. When ω is chosen to be 0, optimization minimizes total contact area without considering the quality input. On the other hand, when $\omega = 1$, contact area overlapping with higher quality regions are minimized. In other words, optimization works towards eliminating any support contact at high quality regions. Figure 4 demonstrates effect of ω parameter on an example case. While printing the object upside down (Figure 4(c)) results in the least amount of contact area (*i.e.*, $\omega = 0$), this build orientation may not be desirable as the head is a perceptually important part of a figurine object.

Formal definition of our build orientation optimization problem is as follows:

$$\begin{aligned} \min_{\theta} \quad & \sum_{j}^{n_F} \kappa_j(\theta) \\ \text{s.t.} \quad & \alpha \in [-\pi, \pi] \\ & \beta \in [0, \pi] \end{aligned} \tag{2}$$

where n_F is the number of faces. In this formulation, only overhang angle, φ needs to be recomputed for each objective evaluation as the object is reoriented. Thus, we have an optimization problem with a reasonably fast objective evaluation and only two optimization variables. We solve this problem using simulated annealing method [KGV83] that finds global minimum when sufficient number of iterations are performed. For the examples of this paper, optimization converged to a solution in a few seconds achieving practical computation times for an interactive tool.

3.4. Generating Streamlines

We construct curvy zigzag toolpaths as connected streamlines generated in a bidirectional field created inside a support polygon. Figure 5 illustrates the main steps of our toolpath generation process. Let S^i be a set of support polygons at layer i and T^{i+1} be the slice of the target object supported by S^i. We start by computing the quality requirements corresponding to the ith layer of our supports, $\boldsymbol{Q}^i$. This is done by simply taking a slab of $\boldsymbol{Q}$ between the layers i and $i+1$, and projecting the per-vertex quality values on the surface of this slab onto the ith layer. This results in a set of samples on the perimeter and inside of T^{i+1} with their corresponding scalar quality values (Figure 5(a)). Then, we construct two fields: *(i)* a scalar density field $\boldsymbol{D}^i$ and *(ii)* a bidirectional field $\boldsymbol{B}^i$ (Figure 5(b)). The density field corresponds to the linear interpolation of the quality samples in $\boldsymbol{Q}^i$. The bidirectional field is obtained as the interpolation of the T^{i+1} and S^i boundary normals as well as a predetermined infill direction. Then, we generate streamlines to fill inside the support polygon by integrating $\boldsymbol{B}^i$ (Figure 5(c)). Here, the spacing between the streamlines are adjusted locally according to the density field $\boldsymbol{D}^i$. Finally, the streamlines are trimmed to the boundaries of S^i and neighboring ones are connected by their start or end points to create final curvy zigzag toolpaths (Figure 5(d)).

For a 3D model with n slices (*i.e.*, $i \in [0, n]$ where $i = 0$ and $i = n$ corresponds to bottom and top slices, respectively), we start from $(n-1)$th slice and process down to $i = 0$. For any layer, if there exists a non-empty set of polygons $C^i = S^i \setminus T^{i+1}$, support toolpaths in this area is required to align with the support toolpaths in the layer above it, S^{i+1} for a successful print. In order to guarantee such a property, we carry over the streamlines in S^{i+1} to S^i and trim them with C^i. Then, we only create new streamlines in the remaining region of S^i, $E^i = T^{i+1} \cap S^i$.

3.4.1. Bidirectional Field

Suppose the infill direction $\boldsymbol{f}$ is predetermined and let $\boldsymbol{d}$ be a vector perpendicular to it. In order to ensure that the streamlines generated

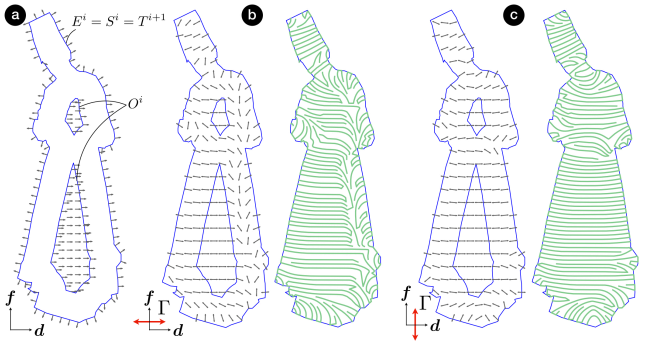


Figure 6: *Construction of an example bidirectional field. For the same input support polygon (a), selection of Γ along* $\boldsymbol{f}$ *results in longer and smoother paths in comparison to Γ along* $\boldsymbol{d}$ *(b-c).*

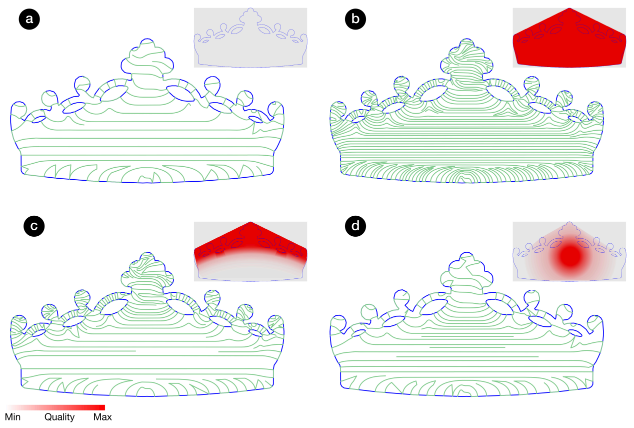


Figure 7: *Effect of density fields on generated toolpaths: constant density fields (a)-(b) versus arbitrary varying density fields (c)-(d).*

inside S^i are perpendicular to both T^{i+1} perimeter and the infill inside it, we construct a field complying with the boundary normals of E^i and $\boldsymbol{d}$. We first compute an *effective infill region* by offsetting E^i inwards. Inside this offset polygon set O^i, we enforce the field to be aligned with $\boldsymbol{d}$. On the boundary of E^i, we constrain the field to be aligned with the normals. Then, the region bounded by the boundaries of E^i and O^i constitutes the *transition region* where the field orientation is obtained by linear interpolation. This allows us to obtain a smooth field inside E^i to generate our streamlines.

Simply computing a vector field through interpolation of boundary normals and $\boldsymbol{d}$, however would result in a large number of singularity points that would prevent us from creating long and smooth paths. Consider a case where two points are located across each other on opposite sides of a rectangular polygon. As the normals of the polygon point outward, they are assigned vectors in opposite directions. Although there may be a single path connecting these two points while satisfying our criteria above, interpolation of a vector field in between these two points would result in a singularity point where the magnitude of the vector field becomes zero. Integration of such a vector field would result in a broken path between these two points. As the paths does not have directionality, we use a bidirectional field to avoid such problems. For bidirectional field interpolation, we represent each vector with the smallest angle it makes with a predetermined fixed axis, Γ. Figure 6 demonstrates an example case. In this example, we assume $E^i = S^i = T^{i+1}$ (*i.e.*, $C^i = \emptyset$ and there are no streamlines carried over from the $(i+1)$th layer) for simplicity.

Bidirectional fields allow singularities in the neighborhood of which the field turns π radians [VO19]. In our formulation, this means that the singularities will occur around Γ. In order to minimize the number of singularities in the resulting field, we select Γ along the infill direction $\boldsymbol{f}$. As the interpolation is often between the boundary normals of E^i and $\boldsymbol{d}$, we observed that selecting Γ this way keeps the active interpolation region away from the problematic area. An example comparison is provided in Figure 6(b) and (c). For the same input configuration, we obtain longer and smoother paths by selecting Γ along $\boldsymbol{f}$ in comparison Γ along $\boldsymbol{d}$.

In cases where $C^i \neq \emptyset$, we additionally use boundary normals of S^i while computing the bidirectional field (see Figure 5). This allows us to expand the field to the entire S^i and extend the streamlines towards C^i. We found such an approach particularly useful when S (and therefore, E) is significantly small at a layer and gradually increases in size as moved towards the bottom. In such a case, streamlines in E become too short in the top layers and get filtered out (as practically it may not be possible to print such short paths), thereby effectively skipping that particular support layer. In the following layers below it, extending streamlines to entire S allow us to compensate for the filtered out streamlines and prevent skipping layers that have large S to generate long enough streamlines.

3.4.2. Streamline Spacing

We adopt a similar approach to [JL97] in creating streamlines with controlled density. Different from this method, our approach controls the spacing between adjacent streamlines locally using the density field $\boldsymbol{D^i}$ rather than a global density parameter. We generate streamlines by performing numerical integration of the bidirectional field $\boldsymbol{B}^i$. Seed point for a new streamline is chosen at a distance d_{sep} away from an existing streamline. As the streamlines are represented as polylines (series of points), this corresponds to simply offsetting a point on a streamline in its normal direction. Then, starting from the seed point, a new streamline is iteratively elongated in both directions until it hits the domain boundary (*i.e.*, boundary of S^i), reaches to a singular point where magnitude of the field is close to 0 or comes closer to another existing streamline than a distance d_{test}. Streamline generation stops when there is no more valid seed points. In our approach, an arbitrary point on the boundary of E^i is selected as the starting seed point to initialize the algorithm. In order to avoid leaving large gaps or skipping unconnected components of S^i, we use additional seed points placed on a regular grid created inside the bounding box of S^i.

In order to control the local spacing between the streamlines, we compute d_{sep} as

$$d_{sep} = d_{min} + (1 - \boldsymbol{D^i}(x,y))(d_{max} - d_{min}) \tag{3}$$

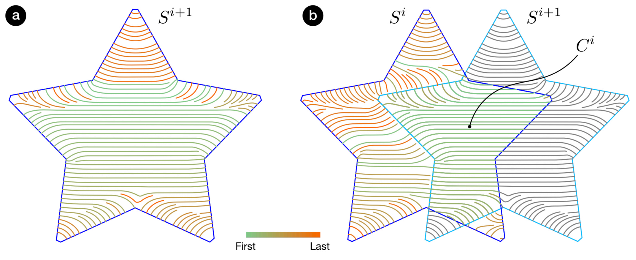


Figure 8: *The order at which each streamline is created reveals our seed point selection approach. Different from $C^{i+1} = \emptyset$ (a), end points of carryover streamlines are used as seed points first when $C^i \neq \emptyset$ (b). Gray parts of the streamlines belong to S^{i+1} and are not carried over to S^i.*

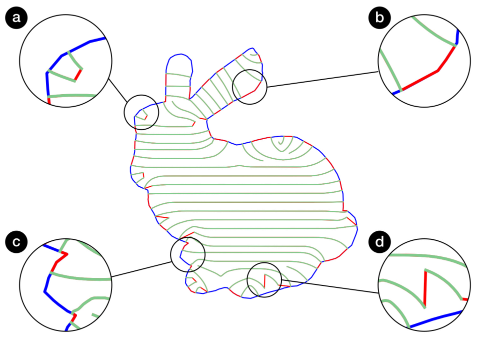


Figure 9: *Two types of connecting paths: Straight path (a)-(d) and boundary following path (b)-(c). All connecting paths are shown in red.*

where $\boldsymbol{D^i}(x,y)) \in [0,1]$ is value of the density field at a location (x,y). Here, d_{min} and d_{max} are minimum and maximum allowed spacing, respectively. d_{max} is dictated by the process and corresponds to the maximum distance for successful bridging. On the other hand, d_{min} may theoretically be 0, corresponding to solid filling of the support area. In our examples, we set $d_{min} = 0.3d_{max}$ to avoid impractically strong adhesion between the supports and the object. We use d_{test} as a percentage of d_{sep}. This relaxes the constraints on streamline generation and helps us obtain longer streamlines by increasing the minimal distance at which the integration of the streamline will be stopped [JL97]. We found that $d_{test} = 0.7d_{sep}$ provides us a good balance between obtaining longer streamlines and having a stricter control on the spacing between streamlines. Figure 7 illustrates the effect of density field on the spacing between streamlines. Toolpaths generated for arbitrary $\boldsymbol{D^i}$'s are demonstrated (Figure 7 (c)-(d)). Two extreme cases where $\boldsymbol{D^i} = 0\ \forall(x,y) \in S^i$ (*i.e.*, $d_{sep} = d_{max}$) and $\boldsymbol{D^i} = 1\ \forall(x,y) \in S^i$ (*i.e.*, $d_{sep} = d_{min}$) are also provided as baseline cases (Figure 7 (a)-(b)).

For the integration, we use a fixed step Euler integrator where each new point of a streamline is calculated as

$$p_{k+1} = p_k + h\bar{\boldsymbol{b}}(p_k). \tag{4}$$

Here, h is the integration step and p_k is the last point of the streamline. $\bar{\boldsymbol{b}}(p_k)$ represents the unit direction vector at point p_k and it is obtained from the field $\boldsymbol{B^i}$. As $\boldsymbol{B^i}$ is bidirectional, there are two possible directions to extend the streamline at any arbitrary point p_k. Suppose $\boldsymbol{B^i}(p_k) = \{+\boldsymbol{b}, -\boldsymbol{b}\}$ where $\boldsymbol{b}$ is a vector. Among the two possible direction, we select the one that deviates the least from the previous step as

$$\bar{\boldsymbol{b}}(p_k) = \underset{\boldsymbol{x} \in \boldsymbol{B}^i(p_k)}{\operatorname{argmax}} (p_k - p_{k-1}) \cdot \boldsymbol{x}. \tag{5}$$

In order to avoid streamline making abrupt turns, we stop the integration when the angle between $\bar{\boldsymbol{b}}$ and the last line segment of the streamline is larger than a certain threshold. In our examples, we found $\pi/3$ radians to work well as this threshold value.

When $C^i \neq \emptyset$, we first elongate the streamlines that are carried over from S^{i+1} as much as possible before starting to generate new streamlines in E^i. For this purpose, end points of carryover streamlines are used as seed points for the integration. New seed points are then created by offsetting the elongated streamlines. Figure 8 demonstrates an example case. Order at which each streamline is created reveals our seed point selection process.

3.4.3. Toolpath Generation

Given the streamlines in each layer of support S^i, the last step in our support generation approach involves connecting neighboring ones to generate long and continuous curvy zigzag paths. To do that, we first trim streamlines to the boundary of S^i. Then, starting from the shortest streamline, our algorithm visits each streamline l_j in order and connects it to another unvisited streamline l_k in close proximity. For a successful connection to happen between l_j and l_k, we look for the following criteria: *(i)* an end point l_j is sufficiently close to an end point of l_k, *(ii)* the connecting path does not intersect with other streamlines or other connecting paths and *(iii)* the connecting path does not leave the boundaries of S_i. When a successful connection occurs between l_j and l_k, extension continues from the opposite end of l_k until there is no more valid connection available.

In our algorithm, we create two types of connecting paths between streamlines– straight path and boundary following path. The former one is created when the connecting end points of both l_j and l_k are inside S^i. In this case, the end points are simply connected with a straight line. The latter one is created when the connecting end points are on the boundary of S^i. This time our algorithm uses the shortest boundary segment connecting these two points as the connecting path. This approach allow us to create toolpaths that supports the perimeters well and maintains the boundary details. An example case is illustrated in Figure 9.

For slices with $C^i \neq \emptyset$, we do not carryover the connecting paths and re-evaluate the connections after all the streamlines are created in the current slice. This is mainly because *better* connections may be created after the carryover streamlines are extended in the new layer. Here, the term better often indicates shorter connection distance.

In the presence of carryover streamlines, C^1 continuity is often broken, reducing to C^0, during the elongation process described

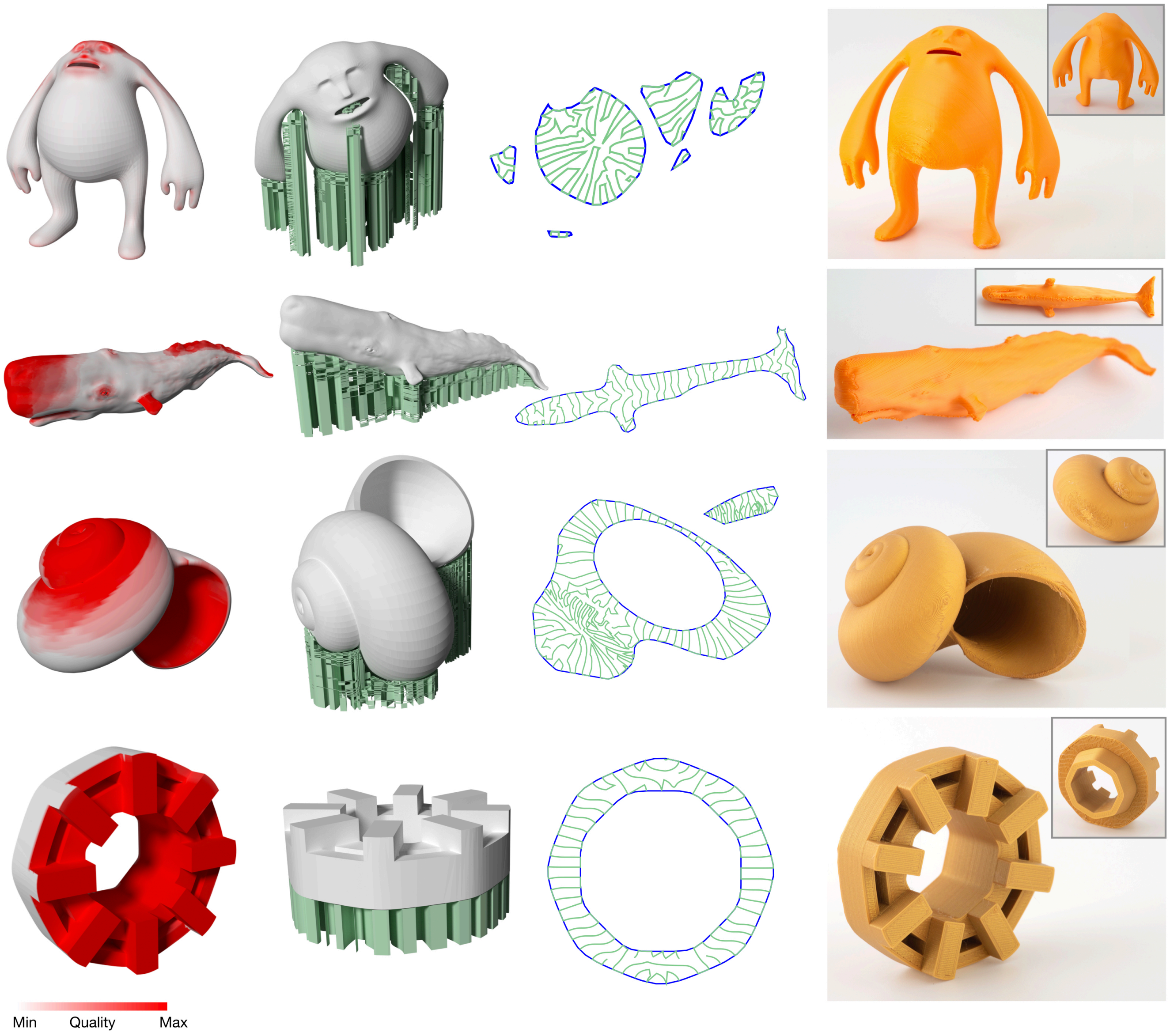


Figure 10: *Example results. Left-to-right: quality preferences, resulting supports in optimum orientation, example slices of support structures and 3D printed results.*

earlier. The reason behind this is that the bidirectional fields often change from one slice to another. For example, carryover streamlines are created by integrating $\boldsymbol{B}^{i+1}$ while the extensions are done in a new field $\boldsymbol{B}^{i}$. In such cases, we smooth the streamline around the junction point using Laplacian smoothing [BKP*10]. Resulting paths are more suitable for FFF type printing as issues related to acceleration/deceleration at sharp corners are eliminated, resulting in an improvement in print time.

After the individual paths are created, machine instructions are compiled in the form of commonly accepted gcode. Here, both our support toolpaths and the target object toolpaths are accumulated together for all layers. We use gsSlicer[Gra20] to generate toolpaths for the target object as well as to convert all the toolpaths to gcode.

4. Results and Discussion

We demonstrate the performance of our method on a variety of models and validate it with 3D printed results. For all the examples, we set $\boldsymbol{f} = [0,1]$, $\varphi_t = \pi/4$, layer height to be $0.125mm$ and $\omega = 1$, unless otherwise stated. Resulting structures are printed on a custom reprap FFF printer using PLA material. In order to show the resulting surface quality objectively, we remove the supports roughly without performing any detailed cleaning or post-processing.

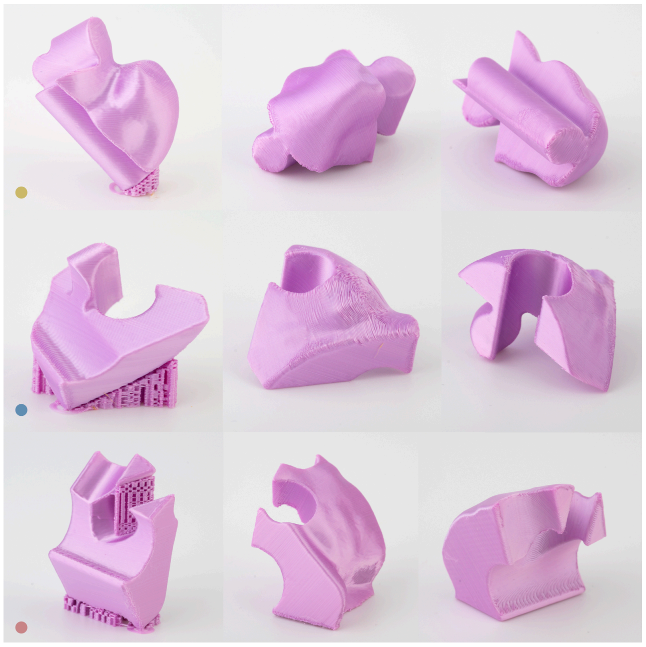


Figure 11: *Detail views of 3D printed bunny puzzle pieces. Each row is marked with the color indicating correspondence to Figure 1.*

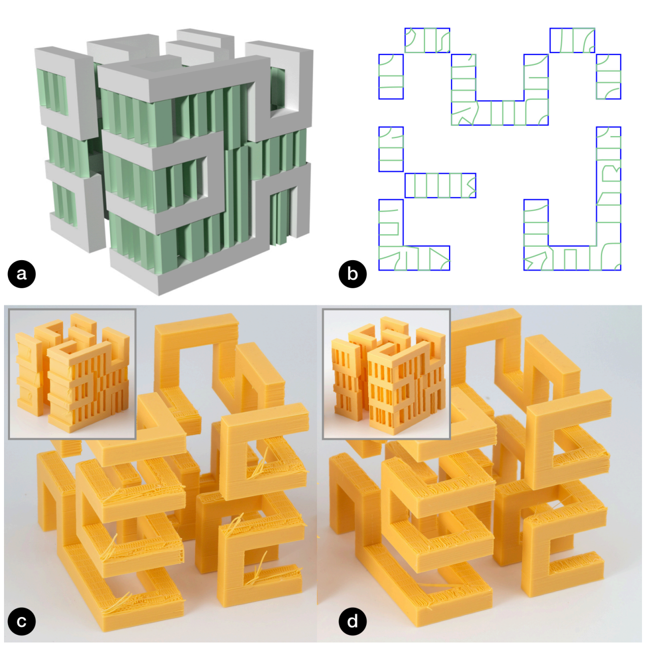


Figure 12: *Hilbert Cube. Curvy zigzag supports (a), an example slice of the support structure (b). Comparison of 3D printed results using regular zigzag (c) and our curvy zigzag (d) supports. Inset figures in (c)-(d) shows the printed models before support removal.*

Figure 10 illustrate example results obtained using Curvy. All the example models are successfully printed in corresponding optimal orientations with curvy zigzag supports. For cases where the build orientation optimization is able to avoid supports in all high quality regions (*e.g.*, Mr. Humpty model in the first row, or mechanical model in the last row), curvy zigzags are generated with constant spacing of d_{max}. On the other hand, when there is no possible orientation that supports can be avoided at high quality regions all together, significant density variations are observed. For example, notice the dense region on the bottom left area of the example slice in the shell model (third row) in comparison to rest of the slice.

In our approach, user selection may be driven by visual preferences (*e.g.*, first three rows of Figure 10) as well as functional requirements. Figure 1 and last row of Figure 10 demonstrate example cases where the motivation for the user selection is mainly functional. In these examples, the surfaces that are required to have tight geometric tolerances due to contact with other parts are assigned high quality values. In resulting prints, supports contacting those surfaces are completely avoided or designed to have high density curvy zigzags to achieve best print quality. Figure 11 shows detail views of 3D printed puzzle pieces. For the face piece (marked with yellow), perceptually important face as well as functionally important tabs and grooves are marked as high quality regions. Therefore, they are left support free as much as possible. For other pieces (marked with blue and red), only tabs and grooves are assigned high quality requirements, therefore some of their outward facing surfaces show support contact marks in resulting prints.

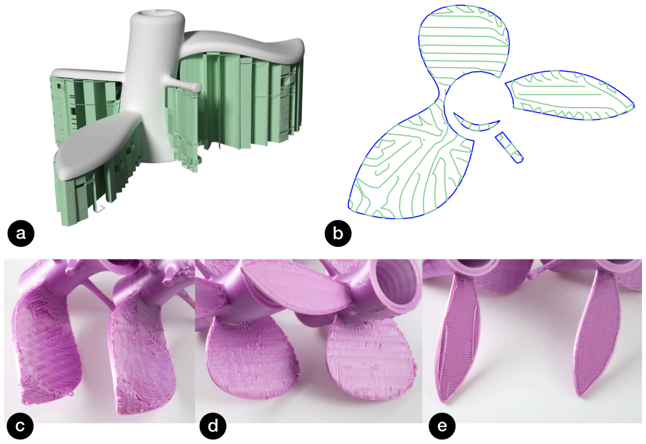


Figure 13: *Plant. Curvy zigzag supports (a), an example slice of the support structure (b). Comparison of 3D printed results using regular zigzag (left) and our curvy zigzag supports (right) in (c)-(e).*

4.1. Comparison

We compare the performance of our curvy zigzag supports with the regular zigzags generated using Simplify3D[Sim20]. For an accurate comparison, we do not optimize the build orientation and we use a constant d_{sep} that is equivalent to regular zigzag spacing. In Figure 12 and Figure 13, we illustrate example cases. In Hilbert

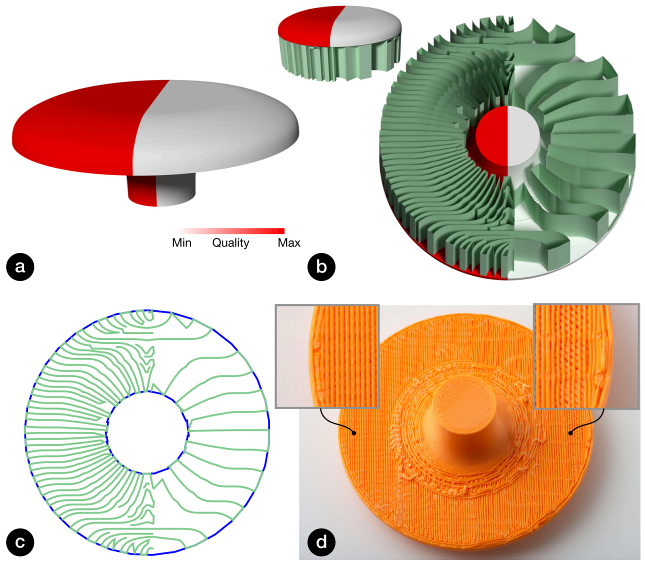
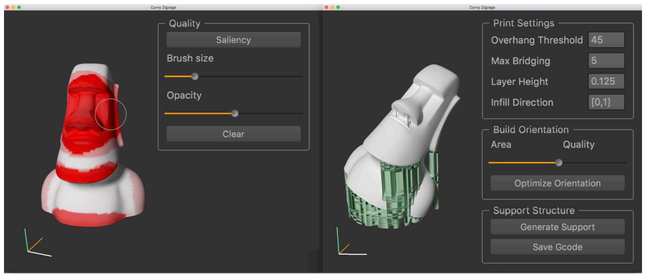


Figure 14: *Mushroom. Desired surface quality (a), resulting curvy zigzag supports (b), an example slice of the support structure (c) and 3D printed result demonstrating the effect of the input user preferences on the surface quality (d). Close-up views of the surface are given in insets in (d).*



Print Settings

Overhang Threshold	45
Max Bridging	5
Layer Height	0.125
Infill Direction	[0,1]

Figure 15: *Interface of our design tool. Quality toolkit for user input (left) and build orientation optimization and support structure generation options (right).*

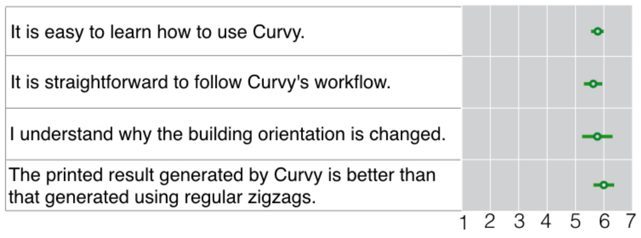


Figure 16: *Survey results for selected questions (1: Strongly Disagree, 7: Strongly Agree). Circles and error bars represent mean values and standard errors, respectively.*

cube example, our curvy zigzags provide better support to overhang surfaces by changing the zigzag orientation to comply with their boundaries (Figure 12(b)) as opposed to constant orientation in regular zigzags. As a result, bridging distance is well maintained overall, thereby allowing a better quality print. Similarly, in plant model, our curvy toolpaths result in significantly better perimeters while quality degrades on perimeter regions that are close to parallel to zigzag direction in regular zigzags.

For the same spacing value, we measured 2% increase in total print time and 10% increase in total material amount with curvy zigzags in comparison to regular zigzags. This increase mainly comes from the fact that curvy toolpaths can cover an arbitrary polygon better and leave smaller amount of empty regions compared to regular zigzags aligned on a predetermined grid.

4.2. Density

We demonstrate the effect of support density on the surface quality on an example model in Figure 14. We set half of the mushroom model to have the highest quality while the other half is assigned the lowest quality value. We observe significant improvement on the perimeters as well as the infill region on the half supported by dense curvy zigzags over the other half supported by low density zigzags. On the low density half, gaps are formed between individual extrusions due to sagging in the infill region and accuracy diminishes due to large bridging distance on the perimeter. Yet, low density half of the supports can be printed using $\sim 70\%$ less material which translates to similar savings in build time.

4.3. Design Tool

We implemented the user interface in C++ and the backend toolpathing algorithm in Python. We use Shapely[†] for geometric operations and SciPy[‡] for field interpolations.

Figure 15 demonstrates the interface of our design tool. After providing the surface quality input, users have an option to perform the build orientation optimization. For the support toolpath generation, users can calibrate the system to their own printer by adjusting d_{max}, $\boldsymbol{f}$, φ_t and layer height. The system outputs G-code files.

4.3.1. Preliminary User Feedback

In order to collect user feedback to evaluate and validate Curvy, we conducted an informal, guided preliminary user study with 8 participants (aged 16-48). 7 out of 8 participants have reported that they are familiar with 3D printing and 6 of them have used a 3D printing software that is capable of generating support structures of some form.

The participants joined the 45 minute remote study where they were introduced the concept of support structures in 3D printing and their effects on surface quality. Then, they are shown a video demonstrating support structure design process in Simplify3D[Sim20] as well as Curvy on the Moai Statue model. Once

[†] https://pypi.org/project/Shapely/
[‡] https://www.scipy.org/

they are familiar with the tools, they are asked to use Curvy to design support structures on an arbitrary model. Participants are asked to use their favorite 3D model or the provided Moai Statue model. They were also shown 3D printed results of Hilbert Cube (Figure 12), plant (Figure 13) and mushroom (Figure 14) models to compare the resulting surface quality with regular zigzag supports. Finally, participants are asked to complete a survey evaluating their overall experience with Curvy and quality of the resulting prints by comparing them with ones obtained using regular zigzag supports (in the Likert scale 1-7).

Survey results for a selected set of questions are reported in Figure 16. Overall, we found that the participants saw a clear benefit of using Curvy for designing support structures. They reported that Curvy is easy and straightforward to use. Yet, for some, adjustment of build orientation optimization weight was not intuitive. Participants found the painting style input easy and intuitive to express quality expectations. They reported to like the immediate feedback they are getting as opposed to conventional way of explicit support design. One important comment we received was that such free-form surface painting was useful for organic shapes but for man-made geometries such as mechanical models, one participant suggested that a surface selection tool would be useful for better precision. In general, participants expressed that the print quality obtained using Curvy is better than what is achieved with regular zigzags. 50% of them indicated strong preference of using Curvy over other conventional software for designing support structures. Only one of the participants reported to prefer conventional software as opposed to Curvy.

4.4. Limitations and Future Work

Our curvy toolpath generation algorithm is most suitable for sparse filling of support polygons. For dense filling, it may create large number of short streamlines. Although this may be technically viable, it may not be practically desirable for toolpath generation. In connecting the streamlines, we use a naive approach of joining streamlines with closest end points. In some cases, this may result in suboptimal connections where the resulting toolpaths are possibly shorter than what could be obtained with an optimization approach.

In our approach, we collect user preferences only on surface quality to design support structures. However, it is possible use similar user interaction to gather high-level preferences on other qualities such as structural performance, model accuracy etc. and translate them to other process parameters in 3D printing similar to our approach. A natural extension to our approach would be to design variable density infill toolpaths with structural considerations.

We provide a saliency map to the user as a starting point for acquiring surface quality preferences. Although the saliency map is useful for guiding users on perceptually important features of the model, it does not provide any information on functionally important parts. In the future, our approach could be extended or complemented with a *functional saliency* map to provide better prediction or guidance on the quality requirements.

5. Conclusion

In this paper, we present an interactive tool for implicit design of support structures through high-level user preferences defined directly on the target object. Perceptual and functional impact of the support structure on the object is attenuated automatically by *(i)* selecting the build orientation that minimize the contact with regions that are intended to have high surface quality, *(ii)* generating curvy toolpaths that conform to the shape boundaries as well as the infill patterns and *(iii)* adjusting the density of these toolpaths locally. Compared to previous methods, combination of these attributes makes our approach a practical and intuitive way to design support structures without requiring low-level knowledge on underlying support generation algorithm.

References

[AAD98] ALEXANDER, PAUL, ALLEN, SETH, and DUTTA, DEBASISH. "Part orientation and build cost determination in layered manufacturing". *Computer-Aided Design* 30.5 (1998), 343–356. ISSN: 0010-4485 3.

[AD95] ALLEN, SETH and DUTTA, DEBA. "Determination and evaluation of support structures in layered manufacturing". *Journal of Design and Manufacturing* 5 (1995), 153–162 2.

[AHL17] ALEXA, MARC, HILDEBRAND, KRISTIAN, and LEFEBVRE, SYLVAIN. "Optimal Discrete Slicing". *ACM Trans. Graph.* 36.1 (Jan. 2017). ISSN: 0730-0301. DOI: 10.1145/2999536 2.

[AKL07] AHN, DAEKEON, KIM, HOCHAN, and LEE, SEOKHEE. "Fabrication direction optimization to minimize post-machining in layered manufacturing". *International Journal of Machine Tools and Manufacture* 47.3-4 (2007), 593–606 3.

[Aut20] AUTODESK, INC. *Autodesk MeshMixer*. http://www.meshmixer.com/. Accessed: 2020-04-06. 2020 2.

[BKP*10] BOTSCH, MARIO, KOBBELT, LEIF, PAULY, MARK, et al. *Polygon mesh processing*. CRC press, 2010 8.

[CJR95] CHALSANI, KUMAR, JONES, LARRY, and ROSCOE, LARRY. "Support generation for fused deposition modeling". *1995 International Solid Freeform Fabrication Symposium*. 1995 2.

[DHL14] DUMAS, JÉRÉMIE, HERGEL, JEAN, and LEFEBVRE, SYLVAIN. "Bridging the gap: automated steady scaffoldings for 3D printing". *ACM Transactions on Graphics (TOG)* 33.4 (2014), 1–10 2.

[DTS16] DELFS, P, TOWS, M, and SCHMID, H-J. "Optimized build orientation of additive manufactured parts for improved surface quality and build time". *Additive Manufacturing* 12 (2016), 314–320 3.

[DWW*18] DAI, CHENGKAI, WANG, CHARLIE CL, WU, CHENMING, et al. "Support-free volume printing by multi-axis motion". *ACM Transactions on Graphics (TOG)* 37.4 (2018), 1–14 3.

[EME15] EZAIR, BEN, MASSARWI, FADY, and ELBER, GERSHON. "Orientation analysis of 3D objects toward minimal support volume in 3D-printing". *Computers & Graphics* 51 (2015). International Conference Shape Modeling International, 117–124. ISSN: 0097-8493 3.

[ERP*19] ETIENNE, JIMMY, RAY, NICOLAS, PANOZZO, DANIELE, et al. "CurviSlicer: Slightly Curved Slicing for 3-Axis Printers". *ACM Trans. Graph.* 38.4 (July 2019). ISSN: 0730-0301. DOI: 10.1145/3306346.3323022 2.

[Gra20] GRADIENTSPACE. *gsSlicer*. http://www.gradientspace.com/opensource. Accessed: 2020-04-06. 2020 8.

[GZN*15] GAO, WEI, ZHANG, YUNBO, NAZZETTA, DIOGO C., et al. "RevoMaker: Enabling Multi-Directional and Functionally-Embedded 3D Printing Using a Rotational Cuboidal Platform". *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. UIST '15. Charlotte, NC, USA: Association for Computing Machinery, 2015, 437–446. ISBN: 9781450337793 3.

[HL18] HORNUS, SAMUEL and LEFEBVRE, SYLVAIN. "Iterative Carving for Self-supporting 3D Printed Cavities". *EG 2018 - Short Papers*. Ed. by DIAMANTI, OLGA and VAXMAN, AMIR. The Eurographics Association, 2018 3.

[HLDC16] HORNUS, SAMUEL, LEFEBVRE, SYLVAIN, DUMAS, JÉRÉMIE, and CLAUX, FRÉDÉRIC. "Tight printable enclosures and support structures for additive manufacturing". *Proceedings of the Eurographics Workshop on Graphics for Digital Fabrication*. 2016, 11–21 3.

[HPL*19] HE, LIANG, PENG, HUAISHU, LIN, MICHELLE, et al. "Ondulé: Designing and Controlling 3D Printable Springs". *Proceedings of the 32nd Annual ACM Symposium on User Interface Software and Technology*. 2019, 739–750 2.

[HYW*09] HUANG, XIAOMAO, YE, CHUNSHENG, WU, SIYU, et al. "Sloping wall structure support generation for fused deposition modeling". *The International Journal of Advanced Manufacturing Technology* 42.11-12 (2009), 1074 2.

[JL97] JOBARD, BRUNO and LEFER, WILFRID. "Creating evenly-spaced streamlines of arbitrary density". *Visualization in Scientific Computing'97*. Springer, 1997, 43–55 6, 7.

[KGV83] KIRKPATRICK, SCOTT, GELATT, C DANIEL, and VECCHI, MARIO P. "Optimization by simulated annealing". *science* 220.4598 (1983), 671–680 5.

[KJB*91] KIRSCHMAN, CF, JARA-ALMONTE, CC, BAGCHI, A, et al. "Computer aided design of support structures for stereolithographic components". *Proceedings of the 1991 ASME Computers in Engineering Conference*. 1991, 443–448 2.

[LL17] LEE, JUSUNG and LEE, KUNWOO. "Block-based inner support structure generation algorithm for 3D printing using fused deposition modeling". *The International Journal of Advanced Manufacturing Technology* 89.5-8 (2017), 2151–2163 3.

[LVJ05] LEE, CHANG HA, VARSHNEY, AMITABH, and JACOBS, DAVID W. "Mesh saliency". *ACM SIGGRAPH 2005 Papers*. 2005, 659–666 4.

[NAI*18] NAKASHIMA, KAZUTAKA, AUZINGER, THOMAS, IARUSSI, EMMANUEL, et al. "CoreCavity: interactive shell decomposition for fabrication with two-piece rigid molds". *ACM Transactions on Graphics (TOG)* 37.4 (2018), 1–13 2.

[PBW*18] PENG, HUAISHU, BRIGGS, JIMMY, WANG, CHENG-YAO, et al. "RoMA: Interactive fabrication with augmented reality and a robotic 3D printer". *Proceedings of the 2018 CHI Conference on Human Factors in Computing Systems*. 2018, 1–12 2.

[Pru20] PRUSA RESEARCH S.R.O. *Slic3r (Prusa Edition)*. https://www.prusa3d.com/prusaslicer/. Accessed: 2020-04-06. 2020 2.

[SDLW16] SHEN, ZHEN-HONG, DAI, NING, LI, DA-WEI, and WU, CHANG-YOU. "Bridge support structure generation for 3D printing". *Materials, Manufacturing Technology, Electronics and Information Science (MMTEI2015) Proceedings for the 2015 International Workshop on Materials, Manufacturing Technology, Electronics and Information Science (MMTEI2015)*. World Scientific. 2016, 141–149 2.

[Sim20] SIMPLIFY3D, INC. *Simplify3D*. https://www.simplify3d.com/. Accessed: 2020-04-06. 2020 2, 9, 10.

[SU14] SCHMIDT, RYAN and UMETANI, NOBUYUKI. "Branching support structures for 3D printing". *ACM SIGGRAPH 2014 Studio*. 2014, 1–1 2.

[TMG*15] TEIBRICH, ALEXANDER, MUELLER, STEFANIE, GUIMBRETIÈRE, FRANÇOIS, et al. "Patching physical objects". *Proceedings of the 28th Annual ACM Symposium on User Interface Software & Technology*. 2015, 83–91 2.

[UGHN20] ULU, ERVA, GECER ULU, NURCAN, HSIAO, WALTER, and NELATURI, SAIGOPAL. "Manufacturability Oriented Model Correction and Build Direction Optimization for Additive Manufacturing". *Journal of Mechanical Design* 142.6 (2020) 3.

[UKY*15] ULU, ERVA, KORKMAZ, EMRULLAH, YAY, KUBILAY, et al. "Enhancing the structural performance of additively manufactured objects through build orientation optimization". *Journal of Mechanical Design* 137.11 (2015) 3.

[Ult20] ULTIMAKER B.V. *Ultimaker Cura*. https://ultimaker.com/software/ultimaker-cura. Accessed: 2020-04-06. 2020 2.

[VGB14] VANEK, JURAJ, GALICIA, JORGE A GARCIA, and BENES, BEDRICH. "Clever support: Efficient support structure generation for digital fabrication". *Computer graphics forum*. Vol. 33. 5. Wiley Online Library. 2014, 117–125 2.

[VO19] VIERTEL, RYAN and OSTING, BRAXTON. "An approach to quad meshing based on harmonic cross-valued maps and the Ginzburg–Landau theory". *SIAM Journal on Scientific Computing* 41.1 (2019), A452–A479 6.

[WCT*15] WANG, WEIMING, CHAO, HAIYUAN, TONG, JING, et al. "Saliency-Preserving Slicing Optimization for Effective 3D Printing". *Computer Graphics Forum* 34.6 (2015), 148–160. DOI: 10.1111/cgf.12527 2.

[WDF*17] WU, CHENMING, DAI, CHENGKAI, FANG, GUOXIN, et al. "RoboFDM: A robotic system for support-free fabrication using FDM". *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2017, 1175–1180 3.

[WDF*19] WU, CHENMING, DAI, CHENGKAI, FANG, GUOXIN, et al. "General Support-Effective Decomposition for Multi-Directional 3-D Printing". *IEEE Transactions on Automation Science and Engineering* (2019) 3.

[WWZW16] WU, JUN, WANG, CHARLIE CL, ZHANG, XIAOTING, and WESTERMANN, RÜDIGER. "Self-supporting rhombic infill structures for additive manufacturing". *Computer-Aided Design* 80 (2016), 32–42 3.

[WZK16] WANG, W. M., ZANNI, C., and KOBBELT, L. "Improved Surface Quality in 3D Printing by Optimizing the Printing Direction". *Computer Graphics Forum* 35.2 (2016), 59–70. DOI: 10.1111/cgf.12811 3.

[XLCT19] XU, K., LI, Y., CHEN, L., and TANG, K. "Curved layer based process planning for multi-axis volume printing of freeform parts". *Computer-Aided Design* 114 (2019), 51–63 3.

[ZLP*15] ZHANG, XIAOTING, LE, XINYI, PANOTOPOULOU, ATHINA, et al. "Perceptual models of preference in 3D printing direction". *ACM Transactions on Graphics (TOG)* 34.6 (2015), 1–12 3.