Toolpath Planning for Continuous Extrusion Additive Manufacturing

Chloë Fleming, Stephanie Walker, Callie Branyan, Austin Nicolai, Geoffrey Hollinger, Yiğit Mengüç

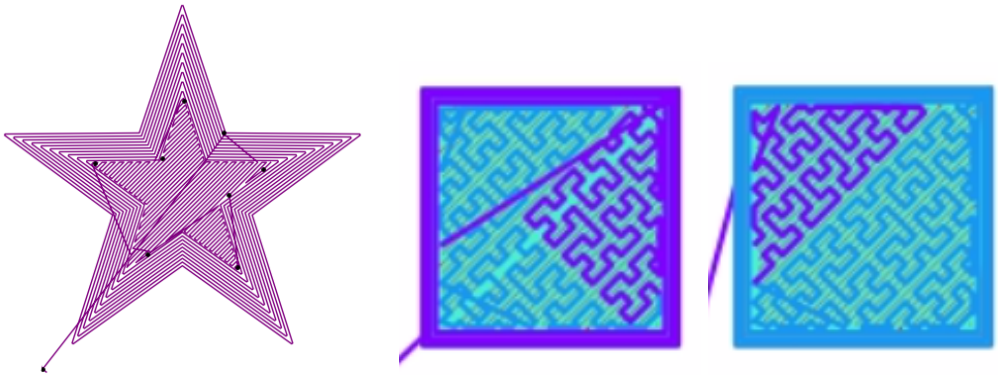
School of Mechanical, Industrial, and Manufacturing Engineering at Oregon State University
2000 SW Monroe Ave, 204 Rogers Hall, Corvallis, OR 97331
flemichl | walkers | branyanc | nicolaia | hollinge | mengucy @oregonstate.edu

Abstract

Recent work in additive manufacturing has introduced a new class of 3D printers that operate by extruding slurries and viscous mixtures such as silicone, glass, epoxy, and concrete, but because of the fluid flow properties of these materials it is difficult to stop extrusion once the print has begun. Conventional toolpath generation for 3D printing is based on the assumption that the flow of material can be controlled precisely and the resulting path includes instructions to disable extrusion and move the print head to another portion of the model. A continuous extrusion printer cannot disable material flow, and so these toolpaths produce low quality prints with wasted material. This paper outlines a greedy algorithm for post-processing toolpath instructions that employs a Traveling Salesperson Problem (TSP) solver to reduce the distance traveled between subsequent space-filling curves and layers, which reduces unnecessary extrusion by at least 20% for simple object models on an open-source 3D printer.

1 Introduction

Advances in additive manufacturing technology make 3D printing a viable fabrication method for a variety of materials beyond thermoplastics, including elastomeric materials such as silicone [1, 2], glass, epoxies, and concrete. When 3D printing viscous materials, as opposed to traditional thermoplastic filaments, it is difficult to stop and start the flow of material. Most print path generation software makes use of the printer's option to enable or disable material flow when constructing the toolpath. These toolpaths tend to produce low-quality prints when used with materials that must be extruded continuously. In this paper, we propose an algorithm for tuning existing toolpaths to reduce extruded material so that the resulting path is better suited to 3D printing fluids.



(a) *First layer of a rectilinear infill toolpath. Extrusion should stop before travel moves, indicated with black dots.*

(b) *Two Hilbert infill toolpaths. The purple path is an in-progress layer. Left: Original toolpath that crosses print. Right: Reordered toolpath that avoids crossing print.*

Figure 1: *Visualizations of toolpath curves and discontinuities.*

The mechanisms for additive manufacturing of viscous fluids differ significantly from the operation of a thermoplastic printer. Pumps push the fluid through tubing and a nozzle to print, and pressure must accumulate in order to get viscous formulations to flow. The nature of the fluid as well as the pressure built up in the pump makes it difficult to retract the fluid once it starts to flow. In contrast, thermoplastic material extrusion relies on a motor to pull the filament down into a heated extruder head so it flows and prints, and filament extrusion pauses when the motor stops.

Generating a toolpath for a print head to fill the boundary of an arbitrary model shape in one continuous motion is a nontrivial problem. Conventional slicing algorithms take advantage of the fact that thermoplastic-type systems can precisely control filament flow, so the resulting extruder paths need not be continuous as the material flow can pause while the print head moves to another region of the print. Because fluid extrusion is continuous, these paths that pass over the middle of the print (as shown in Figure 1a and 1b) or cut across concave outlines drop excess material onto the printed object which detracts considerably from print quality.

In this paper, we formulate an algorithm for rearranging the set of space-filling curves in a toolpath to accommodate continuous extrusion and demonstrate the performance of this method on an open-source 3D printer. We provide a formal definition of a toolpath for additive manufacturing in terms of its component space-filling curves and layers, which we rearrange by combining a Traveling Salesperson Problem (TSP) solver with a greedy approach to planning. This algorithm is applied to several models of varying complexity and its performance is assessed in terms of the quantity of material extruded.

The rest of this paper is organized as follows: Section 2 discusses related work in model slicing, space-filling curves, and toolpath planning. In Section 3 we define a toolpath layer as a graph and formulate a minimization problem that we address with the greedy approach detailed in Section 4. Section 5 discusses the application of this algorithm to object models and analyzes results. Plans for future work are described in Section 6.

2 Related Work

Toolpath generation is performed by 3D model slicing programs that transform the triangular mesh encoded in an STL file into layers, and then compute numeric control commands for travel [3]. The slicer output is a toolpath that describes layer thickness and enumerates travel moves of the extruder [4] as G-code. An example slicing algorithm [5] determines the intersection of the model's triangular mesh facets with slicing planes in order to generate a set of perimeter curves for the model at each layer. Space-filling curves are generated over the area within this boundary curve based on a selected infill pattern type, and all curves are transformed into toolpath instructions.

There are two primary targets for optimization in toolpath generation: boundary filling and tool sequencing. Tool sequencing strategies are used to connect sub-paths in a particular order [6]. Approaches to tool sequencing optimization include connecting sub-sections of a layer to improve machining efficiency [7] or combining multiple layers into 3D regions that will be printed sequentially [8]. One recent work applied heuristic approaches to the Traveling Salesperson Problem (TSP) towards toolpath sequencing to increase 3D printing speed [9]. Boundary filling algorithms aim to generate elegant space-filling curves like Hilbert curves [10] and the Connected Fermat Spiral [11], both of which could enhance continuity within a given layer of a sliced model but require customization to work with complex (non-primitive) geometries [10].

In this work, we formulate an algorithm to improve tool sequencing to minimize extrusion, and do not consider boundary filling algorithms. Our approach reorders space-filling curves to reduce the distance traveled between their endpoints, including the entry point of the layer that is determined by the previous layer's path. The algorithm and implementation were designed for use with toolpaths generated by the free slicing program Slic3r [12] for straightforward compatibility with our open-source 3D printer [2].

3 Problem Statement

We propose an algorithm to post-process toolpaths generated by slicing software to be more suitable for continuous extrusion. The primary goal of optimization for continuous extrusion is to minimize the distance traveled along any curve that is not a space-filling curve for placing material, thus minimizing the misplaced material. We assume that the print head speed and extrusion rate are constant.

Rather than redrawing space-filling curves generated by the slicer, our approach identifies discontinuities in these curves and reorders the continuous regions to minimize total distance between their endpoints. See Figure 1a for a visualization of discontinuities within a layer, marked as black dots on the layer sketch. The algorithm should also consider transitions between layers as discontinuities and plan over the layers to select appropriate start and end points.

A toolpath for additive manufacturing can be described as an ordered set of n layers, $L = \{l_1, l_2, ...l_n\}$. Each layer l_i in L consists of a sequence of space-filling curves over which material is extruded. For each layer, define the set of its m_i space-filling curves $C_i = \{c_{i,1}, c_{i,2}, ...c_{i,m_i}\}$ such that every $c_{i,j} \in C_i$ is *continuous*, meaning that there are no instructions to disable extrusion and move without placing material along the trajectory of $c_{i,j}$.

Between any two sequential curves $c_{i,j}$ and $c_{i,k}$ there is a travel cost between the end of the jth curve and the start of the kth. There is no intrinsic direction associated with a curve, and so either end could be selected as the start point. The distance between the end point of l_i and the start point of the layer l_{i+1} is equal to the distance between c_{i,m_i} and $c_{i+1,1}$. All travel costs are measured in terms of the Euclidean distance between the endpoints, since this quantity is directly proportional to the amount of misplaced material at a constant extrusion rate. We use d to denote the Euclidean distance between the end point of one curve and the start point of another, and define an objective function to determine the optimal ordering of every curve set C_i that minimizes the total travel cost between subsequent curves and layers:

$$\text{minimize } J(C_i) = \sum_{i=1}^{n}[\sum_{j=1}^{m_i-1} d(c_{i,j}, c_{i,j+1})] + d(c_{i,m_i}, c_{i+1,1}) \tag{1}$$

where the layer ordering $i = 1...n$ is fixed with respect to the original layer order and the curve orderings $j = 1...m_i$ for all i are permutations of the sets C_i.

4 Algorithm

Our algorithm greedily reduces unnecessary material extrusion for each layer l_i in succession by selecting an ordering for the curve set C_i, as shown in Algorithm 1. We represent each layer as a fully-connected graph with a node for each curve $c_{i,j}$, and add edges to the graph with costs corresponding to the Euclidean distance between the endpoints of two curves. We also assume that any given curve with endpoints A and B can be traversed from A to B or B to A by the print head without any discernible impact on the resulting print quality. This assumption allows more flexibility in curve reordering with the option to traverse a curve backwards from the end point that is nearer than the original start point. The number of edges in the fully-connected graph increases by a factor of 4 to match the number of endpoint pair combinations.

Our approach first employs the Concorde solver package [13] to generate the shortest closed-loop path to visit all curves in the set. Because the number of curves in a simple object layer is fairly small (under 100), we chose the exact TSP solver implemented in Concorde that uses a cutting-plane method with iterative linear programming [14]. Unlike TSP, a print path should not return to the starting node after extruding all space-filling curves, and there are no particular constraints on which node is visited first. We select an open-loop path from the TSP solution by picking a first curve $c_{i,j}$ that results in the lowest observed cost for this layer. Layer cost is the distance between the first curve $c_{i,j}$ and the last curve in the previous layer,

plus the TSP solution path cost starting from $c_{i,j}$ without including the last edge to return to $c_{i,j}$. Optimal minimization is not guaranteed with this algorithm.

Algorithm 1 Minimize-Material-Greedy

for each layer l_i **do**
Construct-Complete-Graph(C_i) ▷ Add an edge for each pair of curve endpoints with Euclidean distance cost
  $shortest \leftarrow$ Tsp-Concorde(C_i, $c_{i,0}$) ▷ Get exact TSP solution path and cost from Concorde
  for each curve $c_{i,j}$ **do**
    $path \leftarrow$ One-Way-Subpath($shortest$, $c_{i,j}$) ▷ Select subpath from TSP cycle starting at $c_{i,j}$ without return edge to $c_{i,j}$
    $cost \leftarrow \sum^{path} d + d(minpath_{i-1}, c_{i,j})$ ▷ Calculate TSP subpath cost and add distance between $c_{i,j}$ and last layer
    if $cost < mincost_i$ **then**
      $mincost_i \leftarrow cost$ ▷ Save the best path and cost for each layer
      $minpath_i \leftarrow path$
    end if
  end for
  $totalcost \leftarrow totalcost + mincost_i$ ▷ Total toolpath cost accumulates over layers
end for

5 Experimental Results & Discussion

We implemented a set of Python scripts that parse an existing toolpath, construct layer graphs, execute the greedy reordering algorithm, and generate the new toolpath. We applied these scripts to toolpaths produced by Slic3r from four 3D models: a square prism, a cylinder, a star-shaped prism, and an octopus. With the exception of the octopus, all models were a similar size and contained 85 layers for comparable results. The space-filling curves were generated by Slic3r with various combinations of infill pattern and density parameters, which together determine the geometry of the toolpath curves that are reordered by our scripts.

Examples of two infill patterns selected for these experiments, rectilinear and Hilbert, are shown in Figures 1a and 1b. The rectilinear pattern is the most common infill used in 3D printing thermoplastics. The Hilbert infill pattern produced by Slic3r is only visually similar to the Hilbert curve, and not a true continuous space-filling curve. Each object was also sliced with a concentric pattern of perimeters for comparison with the other infill patterns. We varied the density of the infill which, in thermoplastic filament printing, relates to the overall strength of the print. In our experience working with soft materials, 100% density is used in order to avoid gaps, so this setting was used for most of our test cases. We also sliced the simpler objects at 50% infill density.

Solution quality metrics are summarized in Figures 2, with costs measured as percent improvement in travel move distances relative to the unmodified toolpath from Slic3r. Across all shapes and infill combinations, our greedy algorithm significantly reduced path cost, with the square prism with Hilbert infill demonstrating the most impressive percent cost improvement of 90%. The largest reductions were produced on the simplest model shapes. For the square prism and cylinder, the default toolpath cuts across the model at the start of every new layer to begin in the same place. Our algorithm produced considerable improvement by reversing the order of alternating layers to avoid this costly travel move. An example visualization of original and reordered toolpaths is shown in Figure 1b, and Figure 4 captures a comparison of printed layers.

The star prism and octopus models generally showed more modest path cost reductions, although the smallest improvement was still about 20%. For more complex concave geometries like the star and the octopus, curve endpoints are often placed at far ends of convex subregions in the cross-sectional shape. Our algorithm is not able to completely eliminate travel paths that cross the model or cut across concave outlines (as shown by the example octopus printed in Figure 4). The concentric infill objects also showed less improvement than other test cases, but these original toolpath costs were significantly lower than other infill patterns.

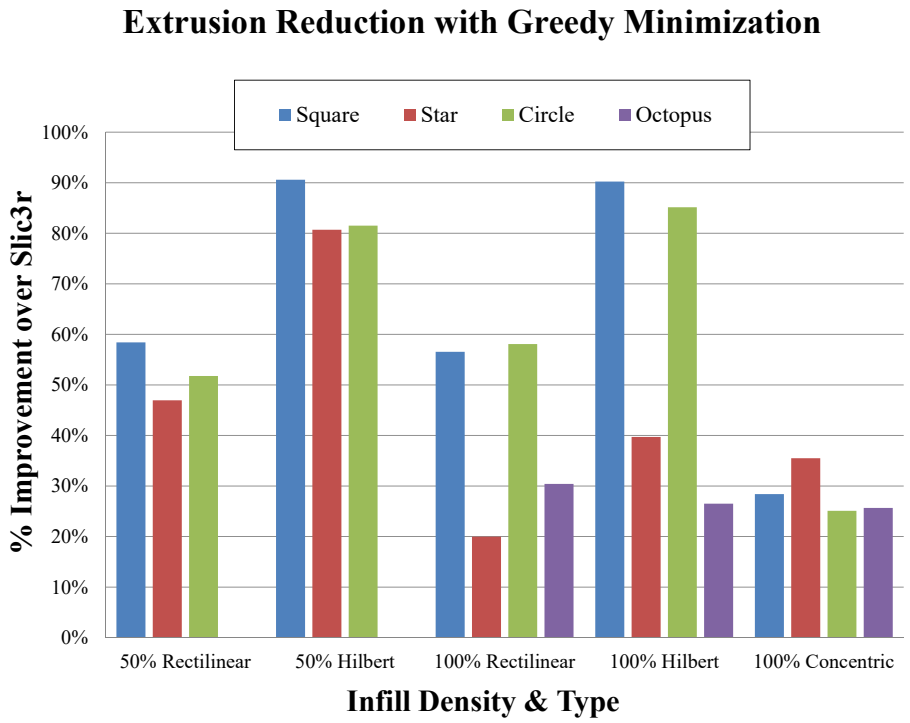


Figure 2: *Graph of percent improvement relative to original toolpath cost in several combinations of slicing parameters for simple prism object models with circular, square, and star cross-sectional shapes.*

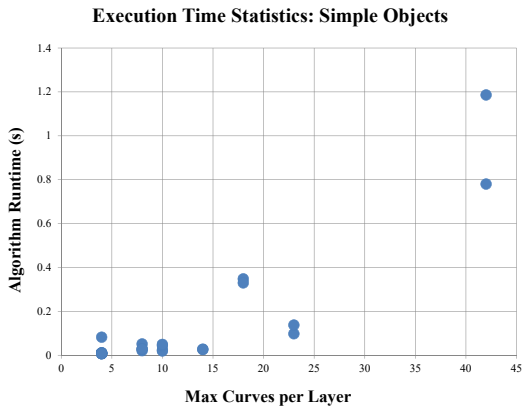


Figure 3: *Algorithm execution time recorded for various prism object models with the same total layer count plotted against the maximum number of curves in any given layer of the model.*

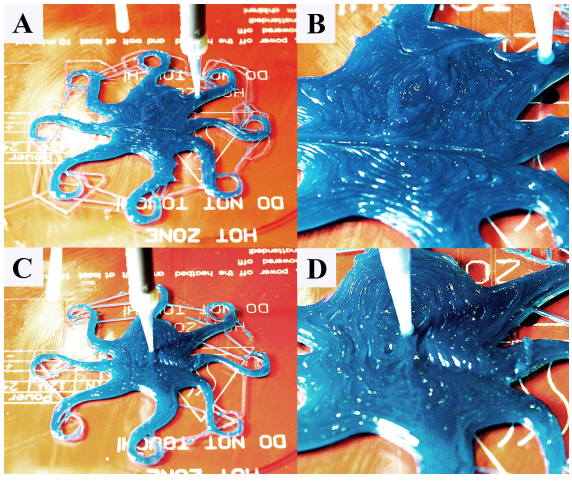


Figure 4: *An example of print quality comparison for original (a,b) and reordered octopus models (c,d). Notice the streaky infill material placement in (b).*

To express algorithm running time with respect to model complexity, in Figure 3 we plot execution time versus the maximum curve count per layer for the prism object test cases. We chose this metric as an indicator of model complexity because solving the TSP is the most computationally intensive portion of our algorithm, and is a function of the number of nodes (curves) contained in the model layers. The highly optimized Concorde TSP solver made our approach computationally tractable for the number of toolpath curves in simple printable objects, although there is still no guarantee of polynomial runtime complexity. All test cases completed in under 2 seconds.

6 Conclusion

We have designed and implemented a greedy algorithm that reorders toolpath instructions to reduce the amount of extrusion outside of space-filling curves, and demonstrated that this approach significantly reduces unnecessary extrusion for three simple object models and a more complex one.

In the future we plan to explore algorithms for generating custom space-filling curves to further improve the quality of complex prints. There may also be potential gains from tuning slicer settings to better fit the continuous extrusion problem, and to this end we will conduct similar experiments with other slicing software and parameters. Another possible trajectory for this work is to characterize the relationship between the geometry of the material extrusions and the mechanical properties of the resulting objects, so that this insight can be applied to guide toolpath planning.

References

[1] Thomas J. Ober, Daniele Foresti, and Jennifer A. Lewis. Active mixing of complex fluids at the microscale. *Proceedings of the National Academy of Sciences*, 112(40):12293–12298, 2015.

[2] J. Morrow, S. Hemleben, and Y. Menguc. Directly fabricating soft robotic actuators with an open-source 3D printer. *IEEE Robotics and Automation Letters*, 2(1):277–281, Jan 2017.

[3] John C. Steuben, Athanasios P. Iliopoulos, and John G. Michopoulos. Implicit slicing for functionally tailored additive manufacturing. *Computer-Aided Design*, 77:107–119, August 2016.

[4] Debasish Dutta, Fritz B. Prinz, David Rosen, and Lee Weiss. Layered Manufacturing: Current Status and Future Trends. *Journal of Computing and Information Science in Engineering*, 1(1):60, 2001.

[5] A. C. Brown and D. de Beer. Development of a stereolithography (STL) slicing and G-code generation algorithm for an entry level 3D printer. In *2013 Africon*, pages 1–5, September 2013.

[6] Yu-an Jin, Yong He, Jian-zhong Fu, Wen-feng Gan, and Zhi-wei Lin. Optimization of tool-path generation for material extrusion-based additive manufacturing technology. *Additive Manufacturing*, 1:32–47, 2014.

[7] Yu-an Jin, Yong He, Guang-huai Xue, and Jian-zhong Fu. A parallel-based path generation method for fused deposition modeling. *International Journal of Advanced Manufacturing Technology*, 77(5-8):927–937, March 2015.

[8] S. Lensgraf and R. R. Mettu. Beyond layers: A 3D-aware toolpath algorithm for fused filament fabrication. In *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3625–3631, May 2016.

[9] Nuwan Ganganath, Chi-Tsun Cheng, Kai-Yin Fok, and K Tse Chi. Trajectory planning for 3D printing: a revisit to traveling salesman problem. In *Control, Automation and Robotics (ICCAR), 2016 2nd International Conference on*, pages 287–290. IEEE, 2016.

[10] Ibrahim Ozbolat and Hemanth Gudapati. A review on design for bioprinting. *Bioprinting*, 3–4:1–14, September 2016.

[11] Haisen Zhao, Baoquan Chen, Fanglin Gu, Qi-Xing Huang, Jorge Garcia, Yong Chen, Changhe Tu, Bedrich Benes, Hao Zhang, and Daniel Cohen-Or. Connected fermat spirals for layered fabrication. *ACM Transactions on Graphics*, 35(4):1–10, July 2016.

[12] Slic3r Manual - Overview. Available at: http://manual.slic3r.org/intro/overview.

[13] Concorde TSP solver. Available at: http://www.math.uwaterloo.ca/tsp/concorde/index.html.

[14] David L Applegate, Robert E Bixby, Vasek Chvatal, and William J Cook. *The traveling salesman problem: a computational study*. Princeton university press, 2011.