Inferring Quality in Point Cloud-based 3D Printed Objects using Topological Data Analysis

Paul Rosen[1], Mustafa Hajij[2], Junyi Tu[3], Tanvirul Arafin[4], Les Piegl[5]

[1]University of South Florida, prosen@usf.edu
[2]University of South Florida, mhajij@usf.edu
[3]University of South Florida, junyi@mail.usf.edu
[4]University of South Florida, tanvirul@mail.usf.edu
[5]University of South Florida, lespiegl@usf.edu

Corresponding author: Paul Rosen, prosen@usf.edu

Abstract. Assessing the quality of 3D printed models before they are printed remains a challenging problem, particularly when considering point cloud-based models. This paper introduces an approach to quality assessment, which uses techniques from the field of Topological Data Analysis (TDA) to compute a topological abstraction of the eventual printed model. Two main tools of TDA, Mapper and persistent homology, are used to analyze both the printed space and empty space created by the model. This abstraction enables investigating certain qualities of the model, with respect to print quality, and identifies potential anomalies that may appear in the final product.

Keywords: 3D Printing, Point Cloud, Topological Data Analysis
DOI: https://doi.org/10.14733/cadaps.2019.519-527

1 Introduction

3D printing is gaining incredible popularity in low-yield manufacturing for customized or specialized parts. However, assessing the quality of models before they are printed remains a challenging problem [5], particularly when you consider point cloud-based models [3], such as those that come from 3D scanners. This paper introduces an approach to quality assessment, which uses techniques from the field of Topological Data Analysis (TDA) to compute a topological abstraction of the eventual printed model and the empty space around and contained within it. This abstraction enables investigating certain properties of the model, with respect to print quality, and identifies potential anomalies that may appear in the final product.

2 Mapper and Persistent Homology

This approach uses 2 of the fundamental tools of TDA, namely Mapper [4] and persistent homology [1], to provide users with feedback about their models (see Figure 1). Mapper is used in 2 ways. First, it is used to extract information about the layer-by-layer connectivity of the model to be printed, providing an abstraction of the overall shape of the object. Second, it is used to determine the topology of the empty space contained within and surrounding the printed model. Persistent homology on the other hand is a tool that normally is used to provide a multiscale view of connected components, holes/tunnels, and voids in data of any dimension. Our approach uses persistent homology for the detection of connected components and holes within a printer layer.

The inner workings and associated details of both Mapper and persistent homology are quite complicated, and so we refer the reader to prior work for a better understanding [1, 4]. We will instead provide an intuition about the types of structures captured by each of these tools.

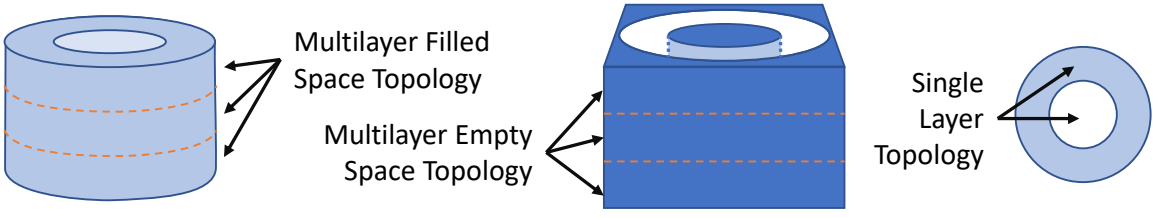


Figure 1: Our approach uses Mapper to look at the filled space topology of multiple layers (left) and empty space topology of multiple layers (middle). It uses persistent homology to understand the topology of a single layer (right).

2.1 Mapper

Mapper is a TDA tool that provides a graph-based abstraction of the topology of a mesh or point-based data. Mapper construction starts by first parameterizing and slicing the data. In our case the parameterization is vertical.

The graph vertices are created from connected components identified within each layer. In other words, the connected components of the layer are "collapsed" into graph vertices. There are many variations on identifying connected components from points. We use the persistent homology approach, introduced in the next subsection.

Finally, graph edges are added between components that touch on neighboring layers. This connection is made by adding a small amount of overlap to each layer. If one or more points in the overlap region are contained within connected components from 2 different layers, those component vertices receive a graph edge. The resulting graph can describe the overall topology of the connected components of a printed object.

Figure 2 shows an example of Mapper on a simple domain. First, (a) the input model is (b) sliced with layer thickness being set to equal the 3D printer's layer resolution. Next, (c) the connected components are found and edges added when they touch. (d) Finally, the illustration of the printed object is shown for comparison. The nodes of the Mapper graphs do not provide any insight into the size or shape of a given connected component. Instead they provide insight into which components touch and how those components may or may not form holes in the output model.

Calculating the Mapper graph on the empty space is a similar process. However, to calculate the graph, the empty space first needs to be filled. This is done by populating the empty space with points. Then, Mapper construction proceeds identically on the empty space points. The approach is illustrated in Figure 3.

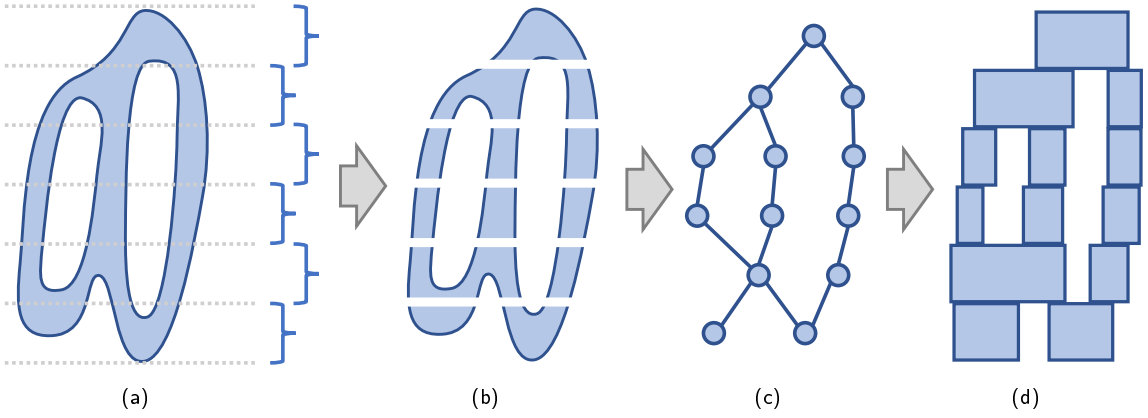


Figure 2: Example of Mapper on a mesh. The (a) model is (b) sliced. (c) Connected components are collapsed to vertices and edges added for components that touch. (d) Finally, an illustration of the printed object is shown.

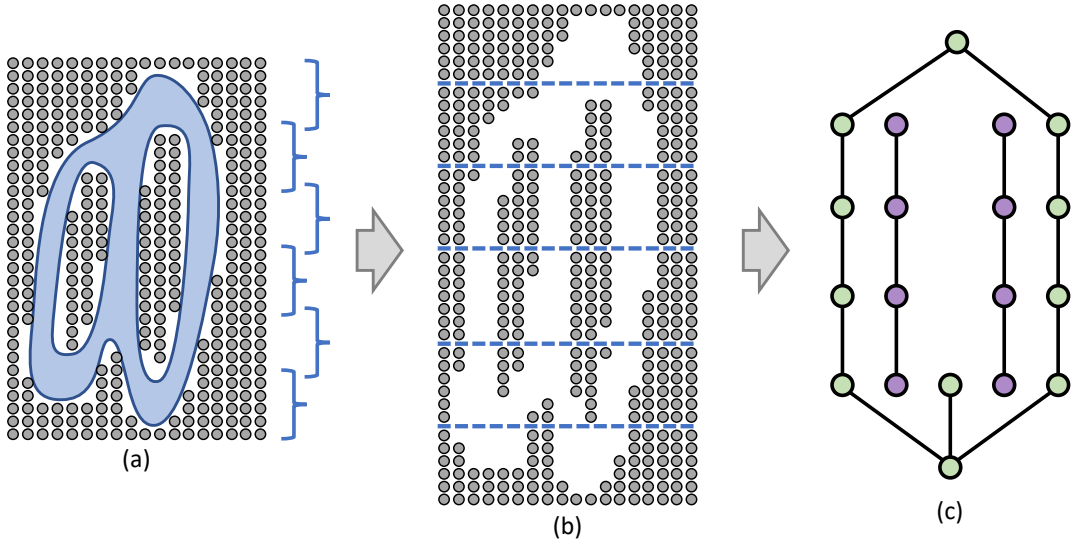


Figure 3: Example of Mapper on the empty space of a mesh. The (a) model has its empty space filled with points and is (b) sliced. (c) The connected components are collapsed to vertices and edges added for components that overlap. For illustration purposes, the vertices here are colored green for outside and purple for inside the model.

The calculation of Mapper is relatively inexpensive. The slicing operation is linear in the number of points. The connected component detection is naively quadratic in the number of points per layer, but this can be improved with spatial partitioning. The overall performance can be improved by using a parallelized algorithm [2].

2.2 Persistent Homology

Given a topological space $\mathbb{X}$, the homology groups $H_0(\mathbb{X})$, $H_1(\mathbb{X})$, and $H_2(\mathbb{X})$, describe the connected components, holes/tunnels, and voids of the space, respectively. For example, consider the annulus in Figure 4(a). It has a single connected component. It also has a single hole/tunnel through the middle. Finally, it contains no void.

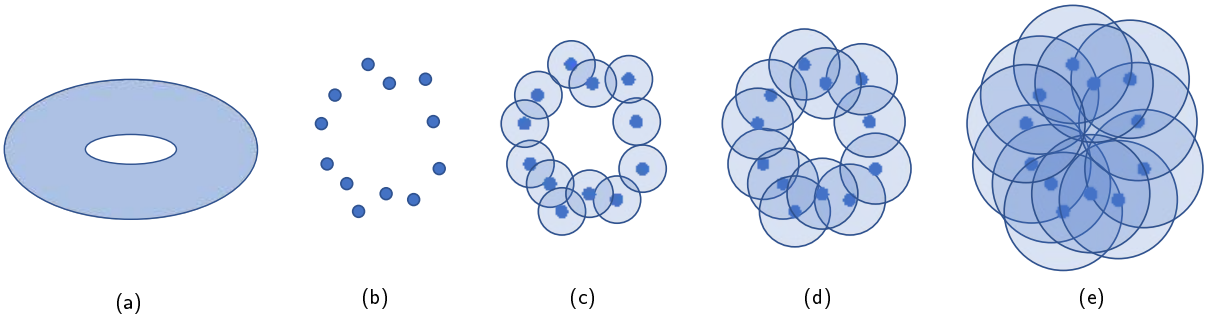


Figure 4: (a) An annulus. (b-e) Example of persistent homology as it relates to a point-based annulus. As points are thickened, from (b) to (e), a hole/tunnel forms in (c) and closes in (e).

The multiscale notion of homology, called persistent homology, extracts the homology groups of a set of points considering different resolutions. A topological feature therefore has a minimum resolution where it first appears, known as the birth time, and a maximum resolution it is still visible, known as its death time. This can be intuitively thought of as the thickening of points. Figure 4(b-e) shows an example. Starting with (b) 12 points, the points are thickened, until (c) they form a single connected component with a hole. As the points continue to thicken (d) the hole remains visible, until (e) the thickness of the points closes it.

The performance calculating H_0 connected components is the same process per layer as with Mapper, naively quadratic. Finding the H_1 homology groups (i.e. holes/tunnels) in persistent homology is quite expensive. This calculation builds a simplicial complex on the data in the form of a boundary matrix and performs a reduction, similar to Gaussian elimination, which leads to a worst case performance that is cubic in the number of points. The average run time is linear with a large time constant. We mitigate this by pre-extracting per-layer connected components and running this calculation only on those components.

2.3 Link Between Mapper and Persistent Homology

The most direct link between Mapper and persistent homology is to use the persistent homology approach in the calculation of $H_0(\mathbb{X})$ homology groups (i.e. connected components) for the individual slices of the Mapper algorithm. However, we augment the conventional Mapper implementation by further attaching the $H_1(\mathbb{X})$ homology groups (i.e. holes/tunnels) to the individual nodes of the Mapper graph. By doing this, the number of holes in each connected component is retained for further analysis.

3 The Topology of 3D Printing

It turns out that both Mapper and persistent homology have direct applications to 3D printing anomaly detection. For Mapper, the slicing operation has a direct corollary in the layers of a 3D printer. Therefore, the slice thickness, known as the cover, can be set to the same value as the thickness of a single layer on the 3D printer (i.e. the z resolution). For persistent homology, the calculation of connected components is the same as a physically connected components within a single layer. The holes within each layer represent

the holes within the model. These can be determined by targeting the xy resolution of 3D printer of interest. Furthermore, using the empty space, Mapper can provide information about the watertightness of the model.

3.1 Visualization

Once the topology of the point cloud has been calculated, we provide a visualization for inspecting the data. The visualization contains 4 components. The first, and most important, is the Mapper graph of the printed model, as seen in Figure 5(a). The Mapper graph nodes shows the individual connected components of the model. In addition, each tunnel going through the connected component is represented by a red point in the node visualization. The next visualization, as seen in Figure 5(d) is the Mapper graph calculated on the empty space of the model, instead of the filled space. The last 2 visualizations are: the 3D point cloud (Figure 5(b)), with regions highlighted based upon the selection of Mapper graph nodes, and a 2D slice visualization (Figure 5(c)), again based upon nodes selected in the Mapper graph.

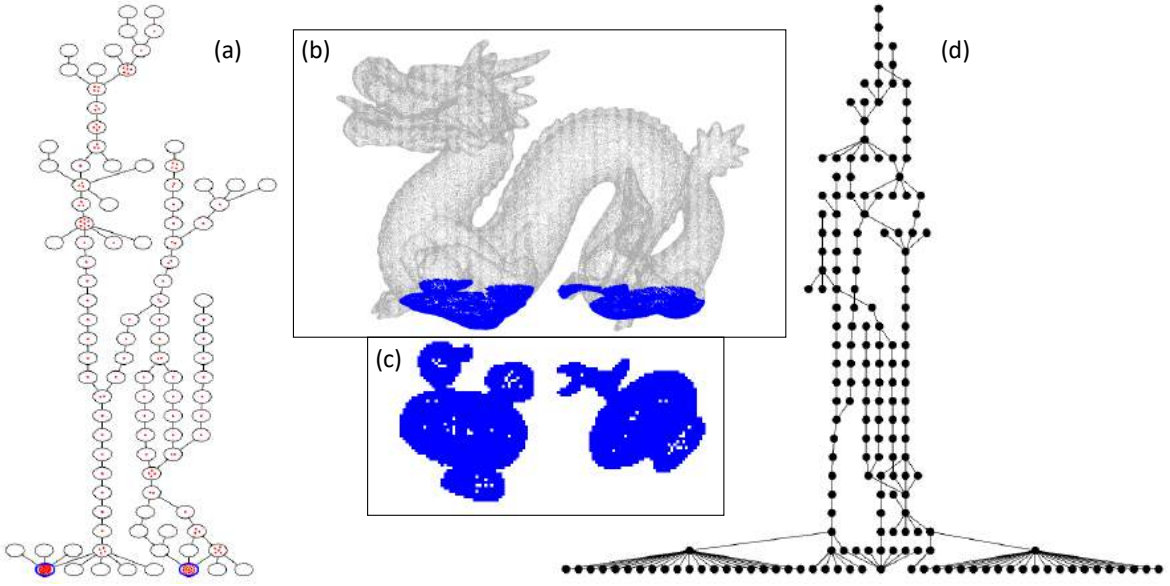


Figure 5: Our software with the Stanford Dragon dataset. (a) The filled space topology is shown as a Mapper graph with holes denoted as red dots. (b) A 3D view and (c) a single slice view are shown for detail. (d) The empty space topology is shown only as the Mapper graph.

4 Results

We implemented our approached using a number of tools. First, data is converted into a point by any method of choice, such as [3]. In our case, PLY or STL files had their vertices extract directly. Our Mapper implementation is in Java. The software loads a point cloud, slices it, detects connected components, and exports the Mapper graph and connected component points for both the filled space and empty space. Each filled space connected component is then fed into Ripser[1] for persistent homology detection of holes/tunnels.

[1]Ripser: https://github.com/Ripser/ripser

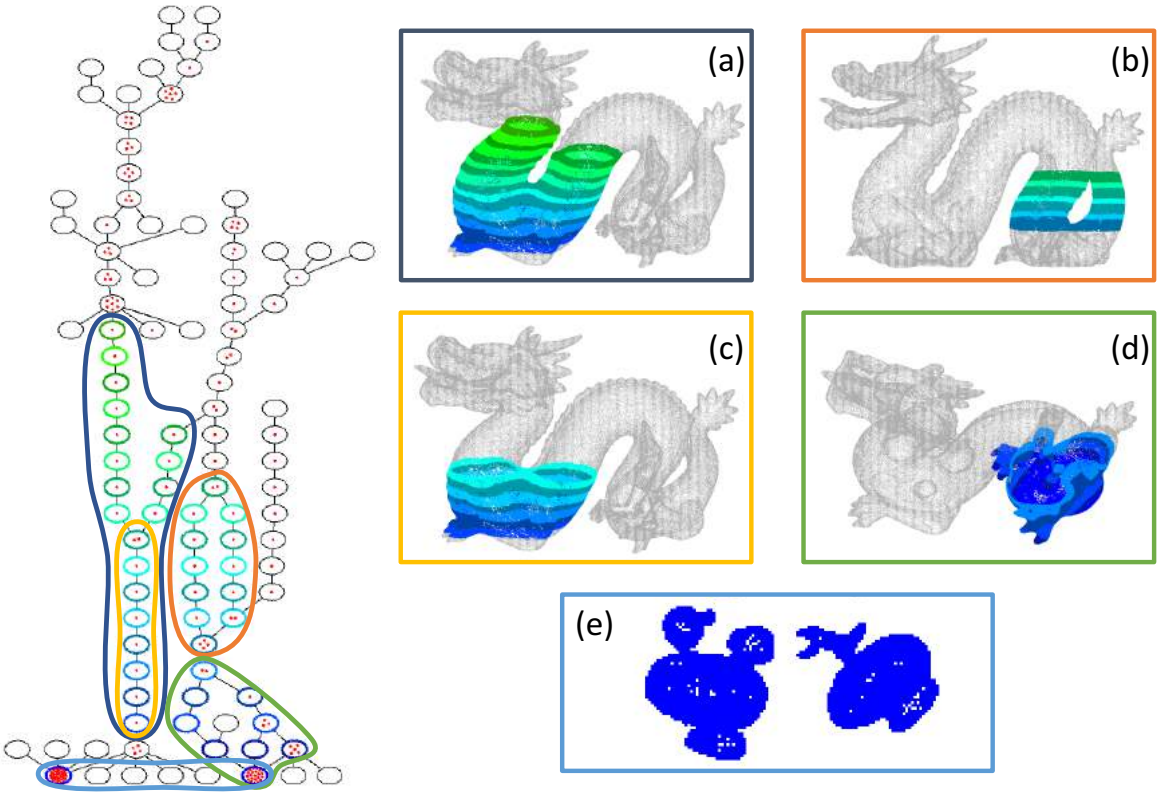


Figure 6: Results of Dragon dataset. The Mapper graph of the filled space (left) has 5 different portions (a-e) highlighted (right).

For the visualization of the Mapper graph, the layout was calculated using Graphviz[2]. The data was then fed into our visualization tool built using Processing[3].

We tested our approach on the Dragon dataset from the Stanford 3D Scanning Repository. We used the points from the reconstructed dataset, which contained approximately 437,000 points. The question we were after was, if someone was to try to rasterize these points directly for 3D printing (ignoring any mesh connectivity), what sort of anomalies would occur. We first scaled the model to a height of 10 cm. We then chose the z resolution to be 3.3 mm and xy resolution to be 1.0 mm.

4.1 Original Model

After running our pipeline, the results are displayed in Figure 5 and Figure 6. In Figure 6, the tree on the left overviews the entire structure of the graph. We will concentrate on the few circled regions.

First, starting with Figure 6(c) in yellow, notice that this region represents a portion of the body of the dragon. In this region, each ring forms a single connected component, each with a single hole through the middle. That is until the topmost ring, where a single connected component has 2 holes, beginning the bifurcation of the upper front and middle portions of the body, as seen in Figure 6(a) in dark blue. This

[2]Graphvis: https://www.graphviz.org/
[3]Processing: https://processing.org/

feature can be observed in the graph by looking at the top most node in the yellow circle. Notice 2 red dots, indicating 2 holes in that component.

Next, notice the region Figure 6(b) in orange. In this region, the model itself splits and comes back together leaving a hole between the torso and tail. This can be observed in the graph as well. Starting after the bottom node of orange region, the graph bifurcates, indicating a split in the connected components, and merges again at the top. This splitting and merging pattern is indicative of an exterior hole in the model. This same type of splitting and merging behavior can also be noticed in the graph region circled in green and associated with Figure 6(d). This hole is caused by the leg and body coming together. However, it is difficult to observe by looking at the 3D imagery of the point cloud. In fact, we could not find a good viewing angle that showed this hole directly.

We now look at the bottom slice of the model in Figure 6(e) in light blue. Looking at the graph, one may observe 2 nodes on the bottom layer that have many red points in the visualization. Each point representing a hole in the layer. This may represent a problem for watertightness, particularly given that this is the bottom layer. Observing the connected components represented by those 2 node in Figure 6(e), many holes are visible

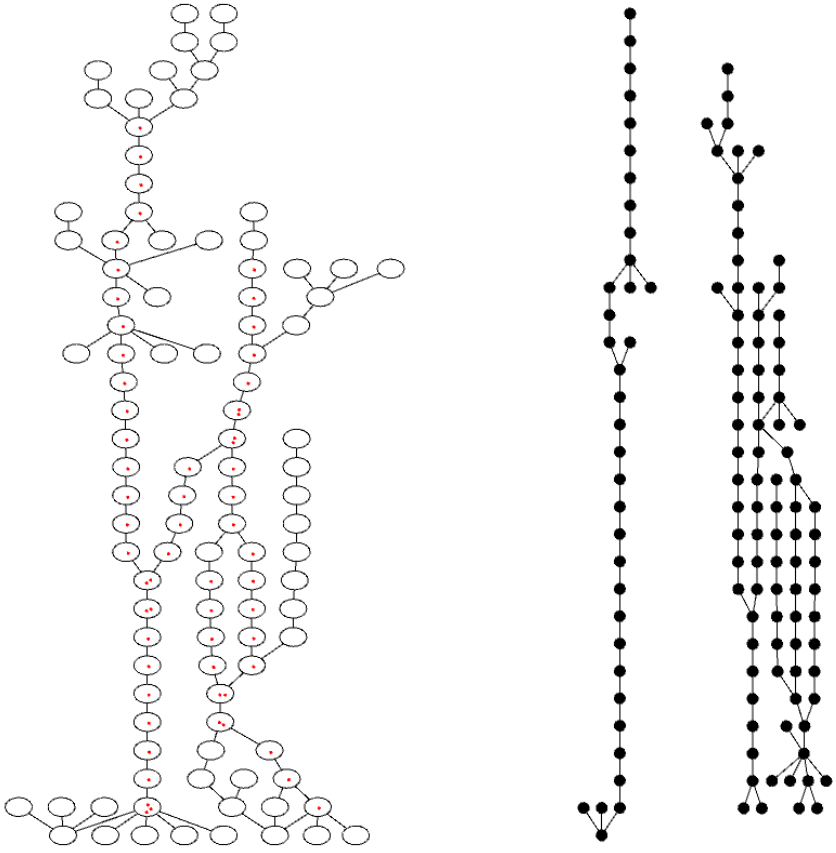


Figure 7: Error Corrected Results of Dragon dataset. (a) The filled space shows a single connected component and holes only on the interior. (b) The empty space has 2 connected components, (left) the outside of the model and (right) the inside of the model. This indicates that the model is now watertight.

in the layer due to inadequate resolution of the points. The initial concern about watertightness remains, given that these holes are not covered by a subsequent layer. Finally, the lack of watertightness can be confirmed by looking at the empty space graph in Figure 5(d). In this graph, there is a single component representing all empty space. If the model were watertight, at least 2 empty space components would form, one outside the model and one or more inside.

4.2 Error Corrected Model

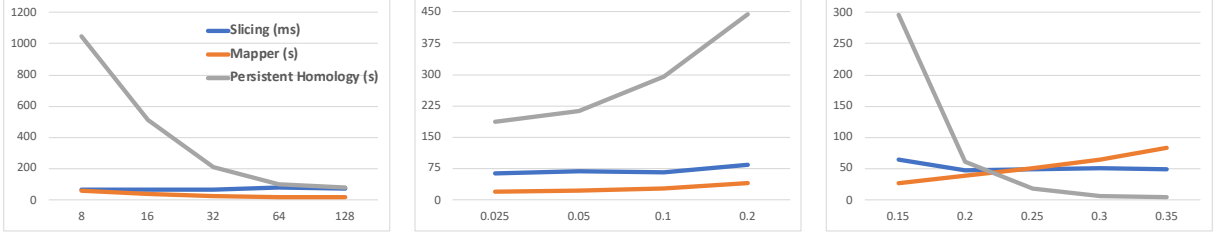
As a comparison, we have computed an error free version of the dragon model. To do this, the triangle mesh provided with the model was subdivided to calculate additional vertices until the point model became watertight. The result of the Mapper and persistent homology calculations can be seen in Figure 7. This new model contained 441,713 points (less than 1% increase from the original), making it visually indistinguishable from the original.

In Figure 7(a), the Mapper graph of the filled space looks identical to the Mapper graph of the original in Figure 6(a). The persistent homology calculation however is quite different. The number of red dots (i.e. holes in the model) have reduced significantly. In fact, the only holes that remain are those representing the major empty cavities of the model's interior.

In Figure 7(b), the Mapper graph of the empty space is shown. The most important aspect of these new graphs is that there are now 2 connected components. Figure 7(b)(left) represents the connected component of the air surrounding the model. Figure 7(b)(right) represents the air inside the model. The lack of connection between these 2 components indicates that the model is now watertight.

4.3 Runtime Performance

We tested the runtime performance of our analysis on the Dragon data set by varying the 3 main parameters, the number of slices, slice overlap, and the xy grid resolution. The results can be seen in Figure 8. These results show that persistent homology is almost always the largest cost. This high cost can be attributed to regions that have large connected components.



(a) Variable Slice Count: 8-128; Fixed Slice Overlap: 0.05 cm; Fixed Grid Resolution: 0.15 cm

(b) Fixed Slice Count: 32; Variable Slice Overlap: 0.025-0.2 cm; Fixed Grid Resolution: 0.15 cm

(c) Fixed Slice Count: 32; Fixed Slice Overlap: 0.1 cm; Variable Grid Resolution: 0.15-0.35 cm

Figure 8: Performance result varying the 3 main parameters of the approach: (a) number of slices, (b) slice overlap, and (c) xy grid resolution. In all results, the time for slicing is presented in milliseconds, while Mapper and persistent homology are reported in seconds.

5 Conclusions

In conclusion, we have presented an approach for using Topological Data Analysis in the evaluation of the quality of 3D printed objects using point cloud-based models. We made some simplifying assumptions in this paper. For example, we assume that 3D printing resolution is uniform across the entire xy domain, which is not necessarily true. We also chose a naive rasterization procedure, though any other pre-rasterized model would be adequate for analysis in this pipeline.

It is also important to note that this approach, as presented, does not report specific problems, aside from watertightness. It instead enables a number of qualitative analyses that depend upon a user's expectation for the output of their model, including certain global or regional problems, such as issues with number of tunnels expected per component; whether the tunnels are connected; the number of connected components per slice; and which connected components make contact slice-to-slice. This essentially enables answering the question, 'does the printed model topology match my expectations?'

ACKNOWLEDGEMENTS

The dragon model was provided by the Stanford 3D Scanning Repository. This work was supported in part by the National Science Foundation (IIS-1513616).

ORCID

Paul Rosen http://orcid.org/0000-0002-0873-9518
Junyi Tu http://orcid.org/0000-0001-7026-7454
Les Piegl http://orcid.org/0000-0003-0629-8496

REFERENCES

[1] Edelsbrunner, H.; Letscher, D.; Zomorodian, A.: Topological persistence and simplification. In Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on, 454–463. IEEE, 2000. http://doi.org/10.1109/SFCS.2000.892133.

[2] Hajij, M.; Assiri, B.; Rosen, P.: Distributed mapper. arXiv preprint arXiv:1712.03660, 2017.

[3] Oropallo, W.; Piegl, L.A.; Rosen, P.; Rajab, K.: Point cloud slicing for 3-d printing. Computer-Aided Design and Applications, 15(1), 90–97, 2018. http://doi.org/10.1080/16864360.2017.1353732.

[4] Singh, G.; Memoli, F.; Carlsson, G.: Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition. In Eurographics Symposium on Point-Based Graphics, 2007. http://doi.org/10.2312/SPBG/SPBG07/091-100.

[5] Telea, A.; Jalba, A.: Voxel-based assessment of printability of 3d shapes. In International Symposium on Mathematical Morphology and Its Applications to Signal and Image Processing, 393–404. Springer, 2011.