

**Title:**

Data Clustering and Slice Generation of Point Cloud Data for Direct 3D Printing System

Authors:

Tianyun Yuan, tyuan@student.pvamu.edu, Prairie View A&M University

Xiaobo Peng, xipeng@pvamu.edu, Prairie View A&M University

Dongdong Zhang, peterzdd_2002@hotmail.com

Lin Li, lilin@pvamu.edu, Prairie View A&M University

Keywords:

Rapid Prototyping, 3D Printing, Direct Manufacturing, Point Cloud Data

DOI: 10.14733/cadconfP.2019.106-110

Introduction:

There is a demanding need for rapid replication or reproduction of physical objects. In many industries such as turbine manufacturing, aerospace, biomedicine, culture heritage etc., the CAD files of parts of interest are usually unavailable or inaccessible. In some situations, such as automotive styling, the designed parts are created by the designer using clay, wood, or foam rubber. The conventional method of reproducing parts includes two main processes, i.e., Reverse Engineering (RE) and manufacturing. The conventional procedure of reproducing a part includes several steps, as shown in Fig. 1. First, the cloud point data is collected from the surface of an existing part by 3D scanning. Next, the CAD model of the object is reconstructed using professional software. Afterward, the CAD model is converted into a facet model. The facet model is then sliced into layers by the slicing software. A G-code file is generated and imported to the 3D printer for final manufacturing. The whole process is far from being automatic. The steps result in rather expensive remodeling computations, large file transmissions, and highly accumulated approximation errors [2]. Laborious work and professional knowledge are required for all steps. Users have to be familiar with various modeling software, which usually requires years of training for them to be efficient and productive.

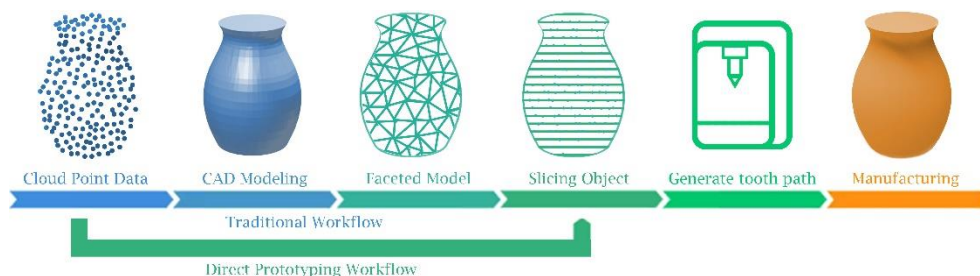


Fig. 1: The comparison between traditional prototyping and proposed direct prototyping methods.

In our previous research, an experimental direct rapid prototyping system was developed, which automatically prints the object from point cloud data using Moving Least Square (MLS) method [3]. This paper presents the development of a direct 3D printing system to automate the process to rebuild or duplicate a physical object. The physical object can be manufactured or duplicated from cloud point data by avoiding the CAD model or STL file reconstruction. A data clustering algorithm is developed to

handle the 3D datasets and solve the issues of multiple contours or each slicing plane. The developed system integrates the processes of model reconstruction from point cloud data, model slicing, printing path generation, and 3D printing. Laborious work and processing time can be saved using the system.

Methods:

The workflow of our direct 3D printing system is shown in Fig. 2. The processes involving user's interventions are shown in orange, whereas the computational processes shown in green are executed automatically. The users are only involved in the first and last step, i.e., inputting the cloud point data and starting the printing process. In the algorithm, the calculation loop scans through the input data along the printing orientation layer by layer. A bounding box with a preset height is applied to each layer. Points in the bounding box are divided into groups by the Improved Density-Based Spatial Clustering of Applications with Noise (DBSCAN) method to solve the multiple contours problem. Afterwards, MLS method is applied to generate the contour for each group. The proposed system also has the ability to print solid parts by filling the contours. The parallel filling pattern method was developed and implemented in the system.

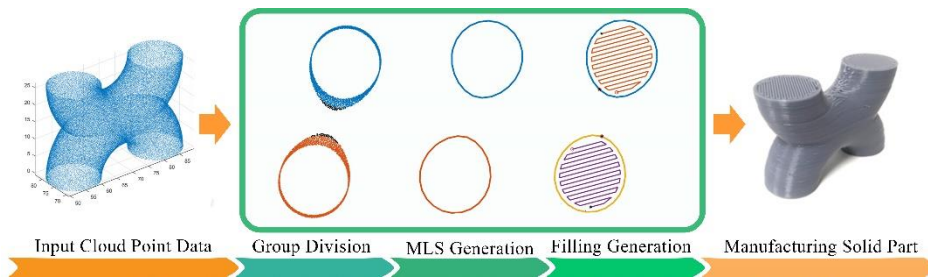


Fig. 2: The workflow of the direct 3D printing system.

Point Cloud Grouping

The improved DBSCAN method was developed for group division of the input cloud data. If the input cloud points (including x, y, z position and i, j, k, normal information) are not grouped properly, then incomplete or broken contours might occur. Such a problem occurs when the slicing plane is close to the saddle points, where the geometry starts to divide into several branches, or the critical points, where the surface reaches the local extrema. The gradient of these points was zero. To solve this problem and to avoid the interference from the mass data, the points in the bounding box were divided into groups before generating the contours.

The classic DBSCAN method [1] defines three types of points, i.e., core point, border point, and outlier point. As presented in Tab. 1. Eps is defined as the maximum radius of the neighborhood. The neighbor points of p are the points q_i in the bounding box and meet the requirement $P = \{q | dist(p, q) \leq Eps\}$. $N_{Eps}(p)$ is the number of neighbor points of the checking point p . $MinPts$ is the minimum number of points in a neighborhood of the checking point p . A core point is a point with high density, whose neighbor point number is greater than or equal to $MinPts$. A border point is a point whose neighbor point number is less than $MinPts$ but is directly density-reachable from a core point. A point q is directly density-reachable from a point p with regard to the Eps and $MinPts$ if q belongs to $N_{Eps}(p)$ and point p is a core point. A point q is density-reachable from a point p if there is a chain of points $q_1, q_2, \dots, q_n, q_1 = p, q_n = q$, such that q_{i+1} is directly density-reachable from q_i . A core point is able to form a new cluster and add all the neighbor points to the cluster and then recursively add their neighbors if they are core points. A border point itself is included in the cluster; however, it cannot start a new cluster, and its neighbor points cannot be directly added to the current cluster. A point is defined as an outlier point if it is neither a core point nor a border point. An outlier point can neither start a new cluster nor be included into any cluster.

	Point type	Point condition	Start a group	Join the group
1	Core point	1) $N_{Eps}(p) \geq \text{MinPts}$ 2) $Z_{Eps}(p) \cap \text{Band}_{z_now} \neq \emptyset$	✓	✓
2	Border point	1) $1 < N_{Eps}(p) < \text{MinPts}$ 2) Density reachable from a core point	×	✓
3	Outlier point	1) The rest of the points	×	×
4	Quasi-critical point	1) $N_{Eps}(p) \geq \text{MinPts}$ 2) $Z_{Eps}(p) \cap \text{Band}_{z_now} = \emptyset$	×	×

Tab. 1: Point types in the improved DBSCAN method.

In the improved DBSCAN algorithm, a new type of point, quasi-critical points, is introduced as a branch of core points. In Fig. 3., the slicing plane noted as Band_{z_now} is considered as band with a tolerance width, where: $Z_{now} - \text{tolerance} \leq \text{Band}_{z_now} \leq Z_{now} + \text{tolerance}$.

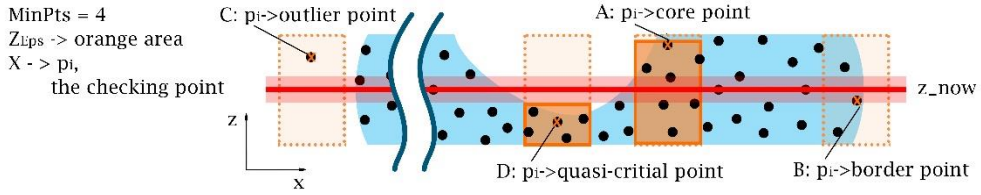


Fig. 3: Examples of the core point and the quasi-critical point.

The z value range of the neighborhood, which is shown as orange area, is noted as $Z_{Eps}(p)$. A quasi-critical point has enough neighbor points (greater than or equal to MinPts), and the slicing plane Band_{z_now} does not cross the z value range of its neighbors ($Z_{Eps}(p) \cap \text{Band}_{z_now} = \emptyset$), as shown in the case D in Fig. 3. Correspondingly, a core point is a point with high density, and the z value range of its neighbor crosses the slicing plane ($Z_{Eps}(p) \cap \text{Band}_{z_now} \neq \emptyset$), as shown in the case A in Fig. 3. The quasi-critical points can neither form a group nor be included in a group. For example, case A in Fig. 3, the neighborhood of a core point p_i crosses the slicing plane z_now . Case D in Fig. 3, all the points found in the neighborhood were below the band of slicing plane; thus, the point p_i is classified as a quasi-critical point. Therefore, the points in the bounding box could be separated into two groups.

Contour Generation Using MLS

The Moving Least Square (MLS) method is briefly described here and illustrated in Fig. 4. The details can be found in the previous work [3]. The input data is a set of points, which include the information of their position and normal. x is the starting point. A set of neighbor points p_i is selected by calculating the distance between x and p_i . A vector \vec{v}_i is assigned as $\vec{v}_i = (p_i - x)$. The Gaussian weighting function is defined as:

$$w(x, p_i) = e^{-\|\vec{v}_i\|^2/h^2} \quad (1)$$

in which h^2 is a scale factor that determines the width of the Gaussian kernel. Here it is taken as a fraction of the local feature size, $h^2 = \|\vec{v}_i\|_{\max}^2$. The moving direction $\vec{n}(x)$ is obtained by Eqn. (2). In Eqn. (2), \vec{n}_{p_i} is the normal at point p_i . The point on the MLS surface will be found in the moving direction.

$$\vec{n}(x) = \frac{\sum_{p_i \in P} w(x, p_i) \times \vec{n}_{p_i}}{\left\| \sum_{p_i \in P} w(x, p_i) \times \vec{n}_{p_i} \right\|} \quad (2)$$

An estimated point y , can be found along the moving direction $\vec{n(x)}$. t is the moving distance between guess point x and the estimated point y along $\vec{n(x)}$. The energy function Eqn. (3) is defined as a function of t :

$$e(t) = \sum_{p_i \in P} ((x + t \cdot \vec{n(x)}) - p_i) \cdot \vec{n(x)}^2 \cdot w(y, p_i) \quad (3)$$

The surface point is the estimated point, whose $e(t)$ reaches to the smallest energy. It means the final moving distance is determined by finding the minimum local energy along the moving direction $\vec{n(x)}$. Then the guess point x is projected to the surface point. The next guess point is calculated by adding a step size in the direction tangent to the normal of the previous surface point until a closed contour is formed. All the surface points found on the slicing plane will be connected as a closed contour.

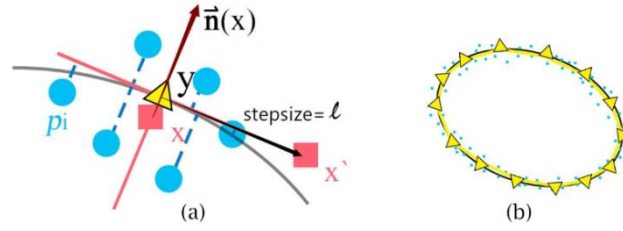
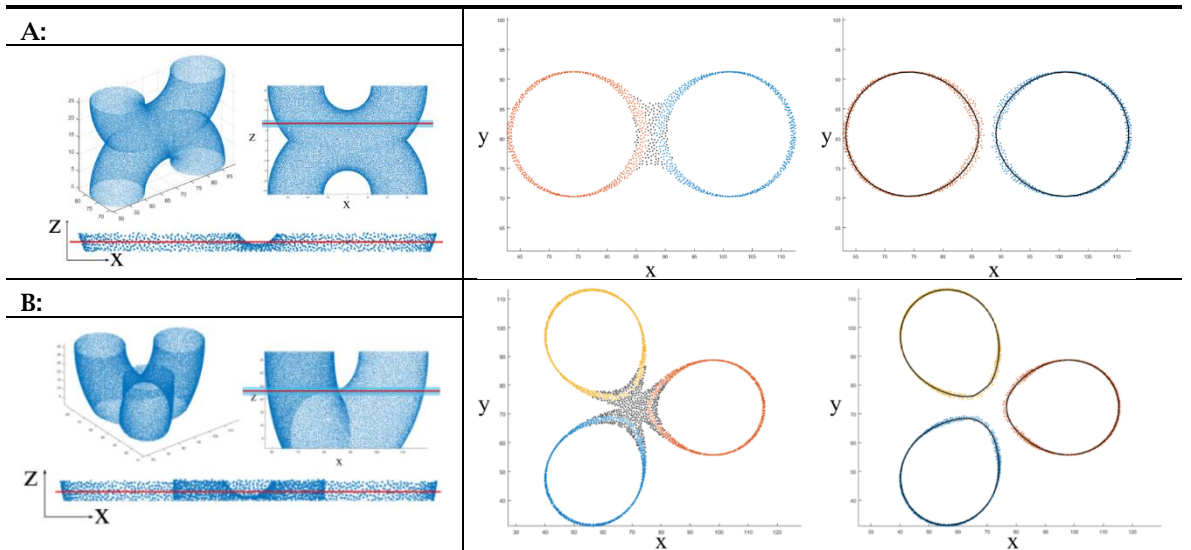


Fig. 4: MLS surface representation and the closed contour generation.

Results and Discussion:

The algorithm was implemented using MATLAB. The results of group division, contour generation, and printed examples are discussed in this section.

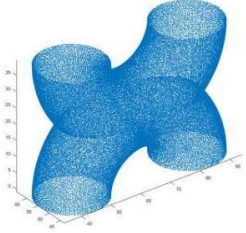
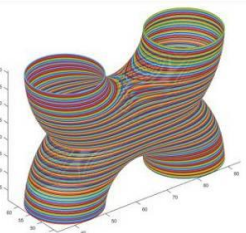

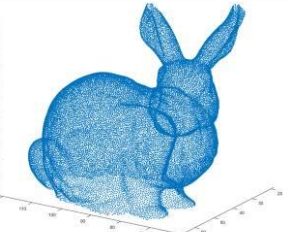
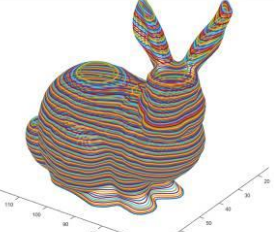

Table 2 shows the results of group division of the two sample geometries. The points in x and z coordinates are the input data, i.e., the points in the bounding box. The x and y coordinates on the right show the result of group division. The points in different groups are shown in different colors, and the quasi-critical points are presented as black dots. A closed contour is generated for each group shown in x and y coordinates on the right.



Tab. 2: Group division for two samples.

In Tab.2, the slicing planes of sample A and sample B are close to the saddle point. These inputs are considered as one group by the classic DBSCAN method because the points in the middle area connect the left and right parts in three-dimension space. However, the improved DBSCAN method is able to recognize the quasi-critical points and divide the inputs into multiple groups. Thus, the MLS contours for each group are generated as desired.

Two models were tested with the direct 3D printing system, as shown in Tab. 3. The printer used in the research work was controlled with open source software named Pronterface (www.pronterface.com). The main material used was PLA. The multiple branch problem was solved by generating multiple contours of the layer. As shown in sample (A), the two branches at the bottom and the top of the model were successfully divided. In sample (B), the two ears of the bunny are separated.

	<i>Point cloud model</i>	<i>Sliced model</i>	<i>Solid-printed</i>
A			
B			

Tab. 3: Examples of 3D printed objects.

Conclusions:

This paper presents a 3D printing system that directly manufactures or rebuilds an existing object. Neither STL nor a CAD model was recreated in this process. To solve the multiple contours problem on one slicing layer, we developed an improved DBSCAN algorithm to divide the points in the bounding box into groups. The geometries were successfully manufactured with the developed system. The sample point cloud data used in this paper were obtained from online public resources or generated from CAD software. Future work will test the system using the data directly scanned from real objects.

References:

- [1] Ester, M.; Kriegel, H. P.; Sander, J.; Xu, X.: A density-based algorithm for discovering clusters in large spatial databases with noise, Proceedings of 2nd International Conference on Knowledge Discovery and Data Mining, 96(34), 1996, 226-231, Portland, Oregon.
- [2] Liu, G.; Wong, Y. S.; Zhang, Y.; Loh, H. T.: Error-based segmentation of cloud data for direct rapid prototyping, Computer-Aided Design, 35(7), 2003, 633-645. [https://doi.org/10.1016/S0010-4485\(02\)00087-8](https://doi.org/10.1016/S0010-4485(02)00087-8).
- [3] Yuan, T.; Peng, X.; Zhang, D.: Direct rapid prototyping from Point Cloud Data without surface reconstruction, Computer-Aided Design and Applications, 15(3), 2018, 390-398. <https://doi.org/10.1080/16864360.2017.1397889>.