www.cadanda.com
Computer-Aided Design AND APPLICATIONS

Direct 3D Printing System: from Point Cloud to Additive Manufacturing

Tianyun Yuan[1], Xiaobo Peng[2], Dongdong Zhang[3] and Lin Li[4]

[1]Prairie View A&M University, tyuan@student.pvamu.edu
[2]Prairie View A&M University, xipeng@pvamu.edu
[3]peterzdd_2002@hotmail.com
[4]Prairie View A&M University, lilin@pvamu.edu

Corresponding author: Xiaobo Peng, xipeng@pvamu.edu

Abstract. Prototyping technology plays an irreplaceable role in the manufacturing industry. Rapid prototyping and reverse engineering are two major technologies that meet the demands of the development. The existing approaches for directly prototyping a physical object involve complex processing steps, including CAD model reconstruction from the scanned point data, and/or stereolithography (STL) model generation. Such processes require professional knowledge and skills and thus are far from automatic processes. This paper introduces a direct 3D printing system that enables automatic 3D printing from the scanned point cloud. Neither a CAD model nor an STL model is required. In the proposed system, the two-dimensional (2D) contours in each printing plane are generated by using the moving least square (MLS) method. An improved clustering method was developed to solve the topology problem of the multiple contours in each slicing plane. Moreover, a filling algorithm was implemented to support fill each contour during the layer-by-layer process. The proposed system simplifies the whole workflow by integrating the point-cloud projecting process, the printing path generating process, and the 3D printing process.

Keywords: 3D printing, MLS method, DBSCAN, direct manufacturing.
DOI: https://doi.org/10.14733/cadaps.2020.825-835

1 INTRODUCTION

3D printing is well-known for its effectiveness on material and time consumption, manufacturing cost, and its ability to produce complex geometry designs for rapid prototyping [12]. Reverse engineering technology [8] refers to the process of copying or creating an existing physical object or surface in the computer environment, in the case that the CAD model or the engineering drawing is not available. These technologies have been widely applied in many fields, such as aerospace, medical care research, education, fashion design, architecture, and the food industry.

In the manufacturing industry, Computer-aided design is playing an increasingly important role. For example, when a complicated surface/geometry is involved, the designed surface is first

replicated on a clay model, and then the model is rebuilt in the computer. The conventional workflow of this process includes several steps, as shown in Figure 1. First, the cloud point data is scanned from the surface. Next, the CAD model of the object is reconstructed using professional software. Afterwards, the 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 manufacturing. Most of these processes require professional knowledge and skills.

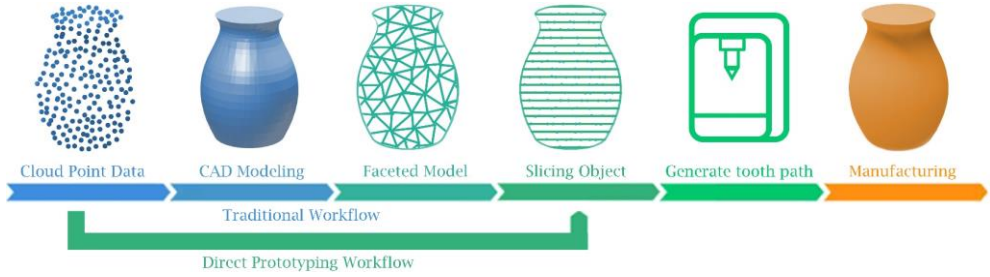


Figure 1: The comparison between traditional and proposed prototyping methods.

A sliced model is essential to manufacture the object with 3D printing technology. Currently, a sliced model is generated from an STL model, which represents the surface with a triangle mesh. Many algorithms have been proposed for the surface reconstruction to represent a more authentic surface with less calculation time. However, the topology problem in the complex geometries remains a challenge in the study of reconstruction and slicing methods. In addition, both processes of reconstructing the facet model and the slicing the model might reduce the accuracy. Since sliced models are the final format for printing, directly generating the sliced contours of from cloud point data will be a better solution for directly 3D printing. Moreover, to manufacture a solid object, the filling pattern and printing routine are also issues needed to be considered.

In our previous research, an experimental direct rapid prototyping system was developed, which automatically prints the object from a scanned point cloud using Moving Least Square (MLS) method [20]. This paper presents the development of a direct 3D printing system to automate the process to rebuild or duplicate a physical object from cloud point data directly by avoiding the CAD model or STL file reconstruction. In the system, the MLS method is used to generate the 2D contours in each slicing plane. Data clustering algorithm is improved to handle the 3D datasets and solve the issues of multiple contours for each slicing plane. In addition, the filling path is implemented to fill each contour to print solid objects. The proposed 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 proposed system.

2 LITERATURE REVIEW

Slicing process in 3D printing involves generation of the contour and the printing pattern for each layer. The input model is sliced into layers, and then the printing path for each layer is generated and translated as machine G-code for the printer to manufacture the parts [15]. Many studies on slicing are based on STL format or a CAD model [13]. However, with the development of 3D scanning technology, many researchers are working on recreating or slicing the point cloud models. The point cloud data is a big dataset that depicts the position of the point on the surface in a Cartesian system. To create the CAD model, specialists select part of the points and recover the surface one region after another by using a four-side surface or other feature commands in the modeling software [3]. Fabio summarized and introduced the process of recreating the surface from cloud points [6]. The Moving least-square (MLS) method has been well implemented in many research studies to reconstruct the surface. A series of surface points is calculated from the massive input dataset. Levin

[7] named such a point-set surface as the MLS surface. Amenta and Kil [2] further studied this method and gave a more explicit definition of the MLS surface with an energy function and vector field. Subsequently, local feature size in the formulation was proposed to guarantee the reconstruction quality from a non-uniform sampling density [4]. Wu et al. [17] applied the correlation concept to determine the neighborhood radius adaptively in the process of curve construction. In Yang and Zhang's research [19],[21], curvature calculation and adaptive slicing were implemented when generating the 2D contours and determining the layer thickness.

Another challenge in the reconstruction process is the topology problem. The scanned model might have several branches or multiple contours on one layer, which will cause problems, such as broken or incomplete results, when generating the contours. One solution is to cluster the input data before the reconstruction. K-means, hierarchical clustering, and density-based spatial clustering are the three main clustering methods [11],[14],[16]. K-means is considered as an effective approach for clustering large data sets. However, the result might change according to the initial center point selected and the cluster number defined initially [9]. The hierarchical method reorganizes the input data in a tree structure, which forms a hierarchical relationship. This method is mainly applied to a categorical database. Density-based spatial clustering [5] can automatically discover clusters of arbitrary spatial input. This method performs very well on a large dataset; however, it is sensitive to higher dimension space. In Yang et al. [18], the topology problem is solved by finding the critical points by using Morse theory and Lagrangian multiplier formulation.

This research applied the MLS method to reconstruct the surface and generate the slicing contours from the point cloud data. The topology problem was solved by improving the Density-Based Spatial Clustering of Applications with Noise [5] (DBSCAN) method, named as Improved DBSCAN, to cluster points in 3-dimention. Filling pattern was also implemented to support solid object printing.

3 METHODS

The workflow of our system is shown in Figure 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 from the bottom to the top, 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 DBSCAN method to solve the multiple contours problem. Afterwards, MLS method is applied to generate the contour for each group. The system also can print solid parts by filling the contours. The parallel filling pattern method was developed and implemented in the system.

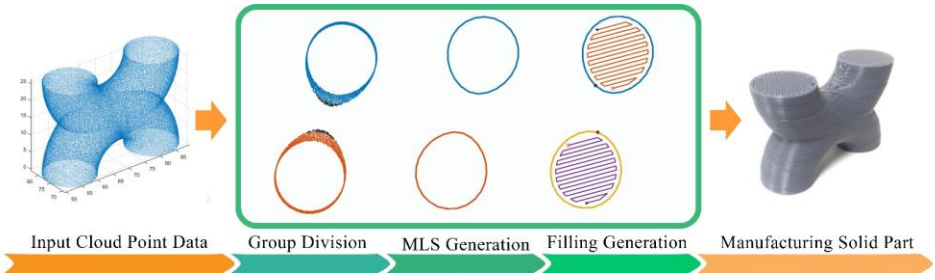


Figure 2: The workflow of the direct 3D printing system.

3.1 Point Cloud Clustering

The Improved DBSCAN method was developed for group division. 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 as the situations shown in Figure 3. Such a problem occurs when the slicing plane is close to the critical point whose gradient is zero. When reaching to critical points, the surface approaches the local extrema or the geometry starts to divide into several branches. To solve this problem and to avoid the interference from the mass data, the points in the bounding box are divided into groups before generating the contours.

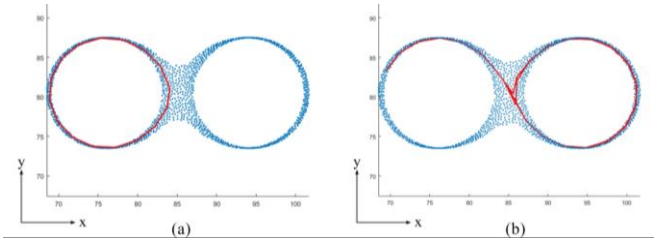


Figure 3: The incomplete or broken contours generated without clustering step.

In the system, the Improved DBSCAN method was implemented to realize the clustering process. Points are classified regarding to the density of their neighbor area. Four types of points were introduced in the Improved DBSCAN, as following:

A) Core point: the point with high density;

B) Quasi-critical point: the point with high density but located at or close to the critical point of the geometry;

C) Border point: the point close to or at the border of the geometry and lead to a less density of its neighborhood;

D) Outlier point: the point with low density.

Core point, border point, and outlier point are originally introduced in classic DBSCAN method. A new type of point, Quasi-critical point, was newly introduced in our method. Quasi-critical point can help to examine if there is a critical point in the slicing layer. The definition and functions of each point type are summarized in Table 1. Eps is defined as the maximum radius of the neighborhood of a point p. The neighbor points of p are the points q_i in the bounding box and meet the requirement $\mathrm{P} = \{\mathrm{q}|\mathrm{dist}(\mathrm{p},\mathrm{q}) \leq \mathrm{Eps}\}$. $\mathrm{N_{Eps}(p)}$ is the number of neighbor point in the neighbor area Eps. $MinPts$ is the minimum number of points in a neighborhood of the checking point p. A point q is directly density-reachable from a point p regarding 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, …, q_n, $q_1 = p$, $q_n = q$, such that q_{i+1} is directly density-reachable from q_i.

For core point, its neighbor point number is greater than or equal to $MinPts$. A core point can form a new cluster and include all the neighbor points q_i to the cluster, as well as recursively add their neighbors if the neighbor point q_i is also a core point. A border point is a point whose neighbor point number is less than $MinPts$ but is directly density-reachable from a core point. 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 quasi-critical point has neighbor points more than $MinPts$, but differs to core point because of the height range of its neighbor points. The definition is explained below. A quasi-critical point can neither form a new cluster nor be included in any cluster. A point is defined as an outlier point if it is not a core point, a quasi-critical point, or a border point. An outlier point can neither start a new cluster nor be included into any cluster. All point types are illustrated in an example shown in Fig 3.

	Point type	*Point condition*	*Start a group*	*Join the group*

A	Core point	a. $N_{Eps}(p) \geq \text{MinPts}$ b. $Z_{Eps}(p) \cap \text{Band}_{\text{z\_now}} \neq \emptyset$	√	√
B	Quasi-critical point	a. $\text{N}_{\text{Eps}}(\text{p}) \geq \text{MinPts}$ b. $\text{Z}_{\text{Eps}}(\text{p}) \cap \text{Band}_{\text{z\_now}} = \emptyset$	×	×
C	Border point	a. $1 < \text{N}_{\text{Eps}}(\text{p}) < \text{MinPts}$ b. Density reachable from a core point	×	√
D	Outlier point	a. The rest of the points	×	×

Table 1: Point types in the Improved DBSCAN method.

As the example in Figure 4, input points in the bounding box are marked as black dot in the blue area and shown in z-x coordinates. The points need to be clustered. Z_{now} is the slicing plane and $Band_{z\_now}$ is considered as a band with a tolerance width, where:

$$Z_{now} - tolerance \leq Band_{z\_now} \leq Z_{now} + tolerance.$$

Neighbor points of a point p are found in the neighbor area Eps shown with orange dash line in the figure. The z-range of neighborhood noted as $Z_{Eps}(p)$ covers from maximum z value to the minimum z value of the neighbor points. It is highlighted as an orange area with solid boundary.

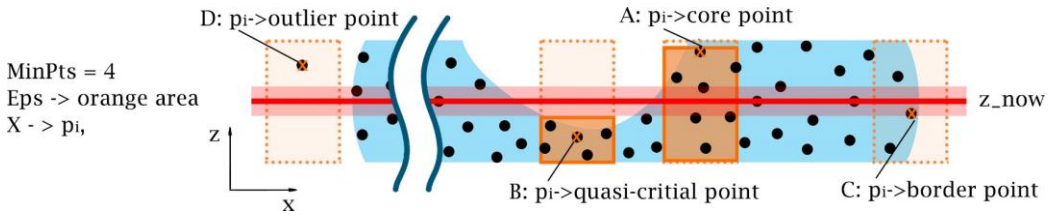


Figure 4: Examples of the core point and the quasi-critical point.

A quasi-critical point, shown as point B in Figure 4, has enough neighbor points (greater than or equal to $MinPts$), and also the z-range of its neighborhood does not cross the slicing plane band $Band_{z\_now}$, i.e., $Z_{Eps}(p) \cap Band_{z\_now} = \emptyset$. In contrast, a core point, shown as point A in Figure 4, is a point with high density, and the z value range of its neighbor crosses the slicing plane, i.e., $Z_{Eps}(p) \cap Band_{z\_now} \neq \emptyset$.

In the Improved DDBSCAN method, all input points will be classified. Figure 4 shows examples of all four different types of points defined in the Improved DBSCAN method. In Case A, the point is classified as a core point because its neighborhood is intersected with the slicing plane z_now. In Case B, a quasi-critical point is defined because there is no intersection between the neighborhood z-range and the slicing plane. Case C shows a border point whose neighbor point number is less than $MinPts$. Case D shows an outlier point, which is not directly density-reachable from a core point. The pseudocode of the Improved DBSCAN method is shown in Figure 5. Points in the bounding box are initially imported to unvisited-list, and the first point in the list will be visited. A visit-list, which is a queue, is created when a cluster is formed and becomes an empty queue when all the points are visited and no more points are added. A new cluster is formed after the visit-list is empty.

3.2 Contour Generation Using MLS

After the points are grouped, one contour will be generated for each cluster. The main idea of generating the slicing contour is to calculate the surface point to represent the input point data. This section introduces the process to generate the MLS contour. The tracing process is explained in Figure 6, where the blue dots are the sample-points p_i; the yellow triangle dots y represents a

surface point calculated from p_i; the red square dot x is the initial guess-point, and x' is the next guess-point for the following iteration to calculate the next surface point.

```
creat unvisited-list with all points in the bounding box
define Eps, MinPts, and tolerance tt
initialize: i_group = 1
WHILE unvisited-list is not empty
    visit first point p in the unvisited-list
    find points in p's Eps area and get neighborPts
    remove point p from unvisited-list
    IF  N_Eps > MinPts
        IF zmax_neighborPts <= z_now+tt OR zmin_neighborPts >= z_now-tt
            -> Quasi-critical_point
        ELSE -> Core_point
            put point p in the group(i_group)
            add neighborPts in visit-list
            WHILE visit-list is not empty
                visit first point p in the visit-list
                remove point p from both unvisited-list and visit-list
                IF is Core_Point
                    add point p in the group(i_group)
                    add neighbor points in visit-list
                ELSEIF is Border_point
                    put point p in the group(i_group)
    ELSEIF  1 < N_Eps <= MinPts -> Border_point
    ELSE -> Outlier_point
i_group = i_group + 1
```

Figure 5: The pseudocode of the Improve DBSCAN method.

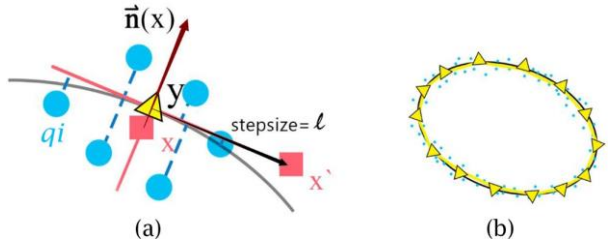


Figure 6: MLS surface representation and the closed contour generation.

For each cluster, the initial guess-point is selected as the point most close to the last surface point of previous cluster. Amount of points, here named as sample-points, are found as the points that closest to the guess-point x. In the example, it is defined as the closest 6 points q_i. Blue dots in Figure 6(a) are the sample-points found around the guess-point x.

A weighing function is applied to these sample-points based on the distribution of them. In this system, it is defined by a Gaussian function as Eq. (1).

$$w(x, q_i) = e^{\frac{-\|\vec{v}_i\|^2}{h^2}} \tag{1}$$

where $\vec{v}_i = \vec{q_i} - \vec{x}$. h is the scale factor in the weighting function.

The surface point is found along the moving direction, which is calculated from the normal vector of the sample points with Eq. (2), as the vector $\vec{n}(x)$ shown in Figure 6(a). The $\vec{n}_{pi}$ is the normal vector of the input points or the calculated normal of each input point.

$$\vec{n}(x) = \frac{\sum_{pi \in P} w(x, q_i) \times \vec{n}_{\text{qi}}}{\left\| \sum_{pi \in P} w(x, q_i) \times \vec{n}_{\text{qi}} \right\|} \tag{2}$$

An energy function was introduced to locate the surface point. An implicit plane, which is perpendicular to the moving direction, is introduced. The distance of each sample point to this plane contributes to the energy. The total energy is the sum of the distance from each neighbor point to the implicit plane, as described by Eq. (3).

$$e(y,\vec{n}(x)) = \sum_{q_i \in Q} ((q_i - y)\cdot \vec{n}(x))^2 \cdot w(y,q_i) \tag{3}$$

where y is the potential surface point, also shown by the triangular dot in Figure 6. The surface point is found along the moving direction $\vec{n}(x)$; thus, it could be depicted as $y = x - \vec{n}\cdot t$, in which, the t is the moving distance from the guess point to the surface point. The energy equation could be restated as a function of t given by Eq. (4).

$$e(t) = \sum_{q_i \in Q} (((x + t\cdot \vec{n}(x)) - q)\cdot \vec{n}(x))^2 \cdot w(y,q_i) \tag{4}$$

The point with the minimum energy is the final surface point used to describe the MLS contour. Also, the normal vector of the final surface point is recalculated by using Eq. (2).

After this, the following guess-point is calculated by adding a step size l in the direction tangent to the normal $\vec{n}(y)$ of the previous surface point. The calculation is iterated until a closed contour is formed. The contour generated, shown as Figure 6(b), is the linear connection of all the surface points in sequence.

4 RESULTS AND DISCUSSION

The algorithms were implemented using MATLAB. The results of group division, contour generation, and filling path are discussed in this section.

4.1 Point Cloud Group Division

The input data of the grouping process are the points selected in the bounding box with a certain height. The input data are in three-dimensional space and grouped using the Improved DBSCAN method. Table 2 shows the results while slicing at different height of the two sample geometries. The left column shows the input data, i.e., the points in the bounding box. The right column shows the result of group division. In these examples, the neighbor point size is set as 30 points in the initial. 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.

The result of sample (d) are obviously in one group, and the results of sample (c) and (f) are in multiple groups. In samples (a), (b) and (e), the slicing plane is close to the critical point. These inputs would be 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 can recognize the quasi-critical points and divide the inputs into multiple groups. Thus, the MLS contours for each group are generated as desired.

4.2 Filling Path for Each Layer

To support solid part printing, a parallel filling pattern [1],[10] was generated inside each contour. In addition, to improve the strength of the solid parts in all directions, a 60 degrees difference between two adjacent layers was considered. The results are shown in Figure 7. The figures in the left three columns present three layers of the printing direction in 2D view, and figures in the right two columns are the 3D view and the top view.

4.3 System Integration and Printing Results

An interface was designed for the direct 3D printing system and implemented by Matlab as shown in Figure 8. The interface had two areas. The left portion of the interface showed two sets of control

parameters, input files, the printing status, and command buttons. The right portion of the interface had three display areas in which the tracing works were graphically rendered in real-time. Figure 8(a) shows the results of group division and the surface points generated on the MLS contour. Figure 8(b) shows the filling pattern and the contour of the slicing layer.

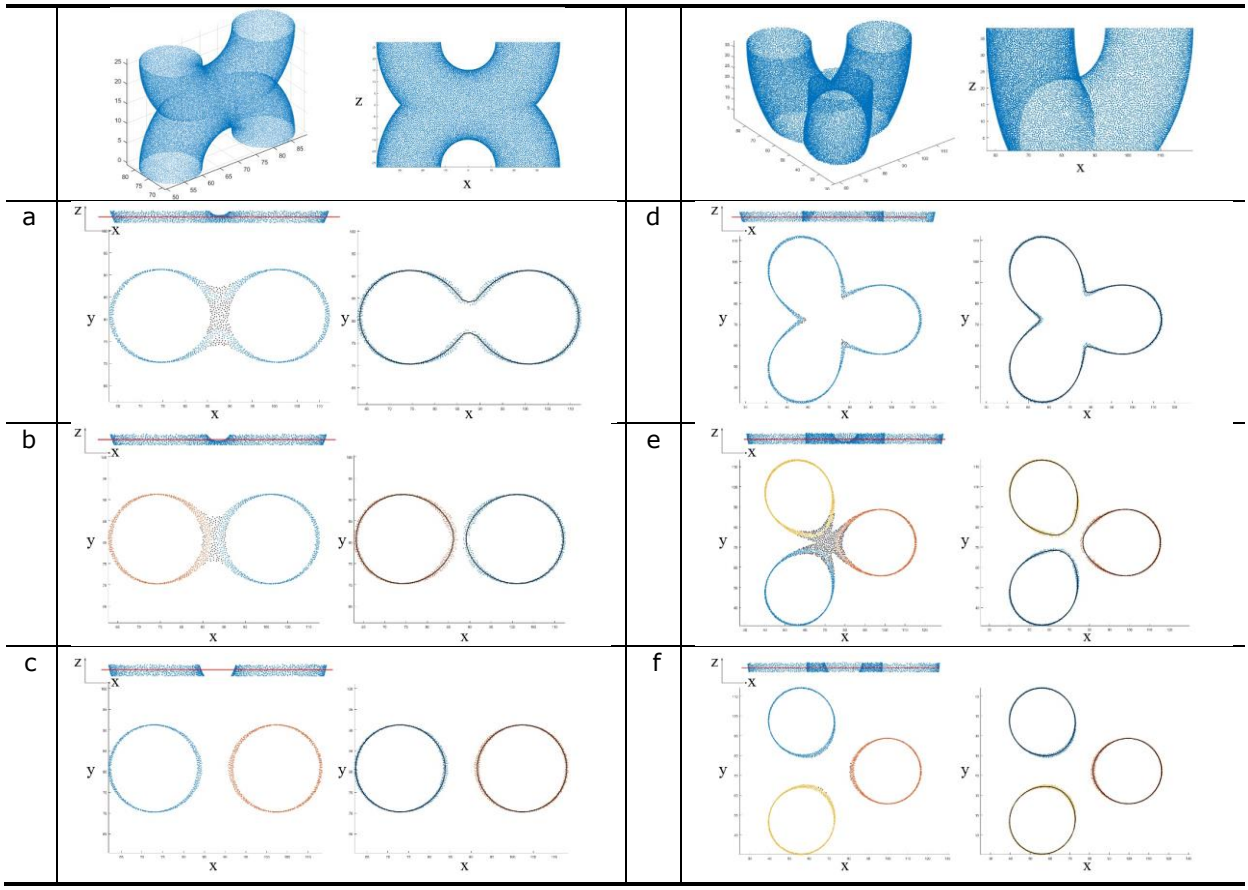


Table 2: Group division for two samples.

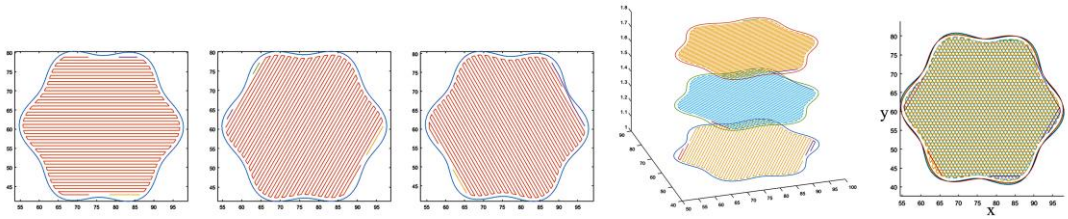
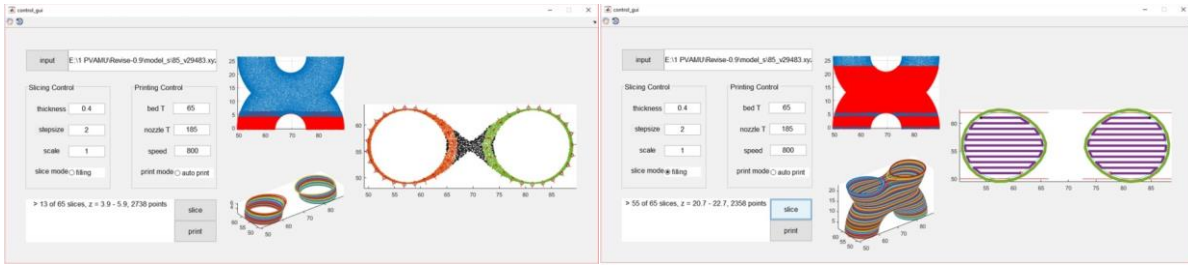


Figure 7: The printing direction of three adjacent layers.

Five geometries were tested with the direct 3D printing system, as shown in Table 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 samples were sliced and printed in two modes: contour-only or solid-printed with filling. The results showed that the direct 3D printing

system can successfully handle the multiple branch problem, as shown in Table 3. The fluffy body and details on the face of the bunny were represented as shown in sample (C). The sample (C) was only successfully printed with the filling because the contour-only mode cannot provide enough self-support. The detailed geometries and small curvature parts were successfully manufactured.



(a) (b)

Figure 8: Examples of the displays of interfaces.

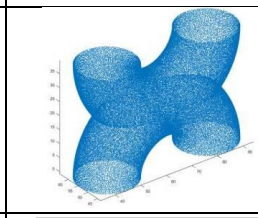
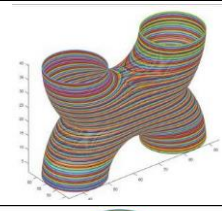
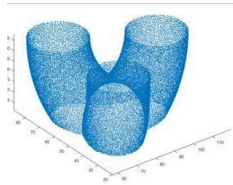
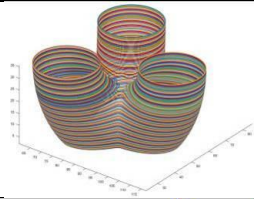
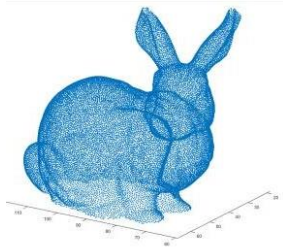
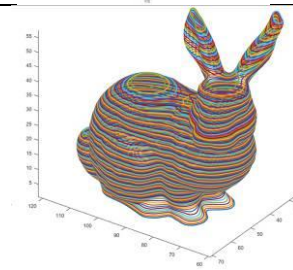
	Point Cloud Model	*Sliced Model*	*Contour-Only*	*Solid-Printed*
A				
B				
C			–	

Table 3: The printed results.

5 CONCLUSIONS

In this research work, a direct 3D printing system was implemented to directly manufacture or rebuild an existing object by integrating RE and RP process. Neither STL nor a CAD model was recreated in this process. The input cloud points were first divided into sections by layers. 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 surface points were calculated from the

input points cloud data by applying the MLS method. The tracing method was implemented to generated closed contours on each slicing layer. Several samples were tested with the direct 3D printing system, which were sliced and printed. The direct 3D printing system could well reconstruct the object from point cloud data. Moreover, the geometries were successfully manufactured with the 3D printing system developed in-house.

The system can be further improved in several aspects. Support structures have not been implemented in the current system. Currently, printing with filling pattern is a way to improve the support of some of the geometries. With the support structures, more complex geometries can be printed with our system. In addition, the optimization of the printing orientation is not available in the current system. Therefore, future work will be focused on design of the support structure and optimization of printing orientation.

Tianyun Yuan, http://orcid.org/0000-0001-6846-6550
Xiaobo Peng, http://ordcid.org/0000-0002-0498-7194
Dongdong Zhang, http://orcid.org/0000-0001-9471-1628
Lin Li, http://ordcid.org/ 0000-0002-9652-8111

REFERENCES

[1] Ahsan, A. N.; Habib, M. A.; Khoda, B.: Resource based process planning for additive manufacturing, Computer-Aided Design, 69, 2015, 112-125. http://doi.org/10.1016/j.cad.2015.03.006
[2] Amenta, N.; Kil, Y.-J.: Defining point-set surfaces, ACM Transactions on Graphics, 23(3), 2004, 26-270. http://doi.org/10.1145/1015706.1015713
[3] Chua, C.-K.; Leong, K.-F.; Lim, C.-S.: Rapid Prototyping: Principles and Applications, World Scientific Publishing Co Inc, Singapore, 2010.
[4] Dey, T. K.; Sun, J.: An adaptive MLS surface for reconstruction with guarantees, Symposium on Geometry processing, Vienna, Austria, 2005, 43-52.
[5] 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), Portland, Oregon, 1996, 226-231.
[6] Fabio, R.: From point cloud to surface: the modeling and visualization problem, International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 34(5)/W10, 2001, http://doi.org/10.3929/ethz-a-004655782
[7] Levin, D.: Mesh-independent surface interpolation, Geometric modeling for scientific visualization, Springer, Berlin, Heidelberg, 2004, 37-49. http://doi.org/10.1007/978-3-662-07443-5_3
[8] 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. http://doi.org/10.1016/S0010-4485(02)00087-8
[9] Jain, A.-K.: Data clustering: 50 years beyond K-means, Pattern Recognition Letters, 31(8), 2010, 651-666. http://doi.org/10.1016/j.patrec.2009.09.011
[10] Jin, Y.-A.; He, Y.; Xue, G.-H.; Fu, J.-Z.: A parallel-based path generation method for fused deposition modeling, The International Journal of Advanced Manufacturing Technology, 77(5-8), 2015, 927-937. http://doi.org/10.1007/s00170-014-6530-z
[11] Joshi, A.; Kaur, R.: A review: comparative study of various clustering techniques in data mining, International Journal of Advanced Research in Computer Science and Software Engineering, 3(3), 2013.
[12] Manyika, J.; Chui, M.; Bughin, J.; Dobbs, R.; Bisson, P.; Marrs, A.: Disruptive technologies: Advances that will transform life, business, and the global economy, McKinsey Global Institute, Washington, D.C., 2013.

[13] Pulak, M. P.; Reddy, N. V.; Dhande, S.-G.: Slicing procedures in layered manufacturing: a review, Rapid Prototyping Journal, 9(5), 2003, 274-288. http://doi.org/10.1108/13552540310502185
[14] Scikit-learn, Comparing different clustering algorithms on toy datasets, accessed May 1, 2017, http://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html.
[15] Venuvinod, P.-K.; Ma, W.: Rapid Prototyping: Laser-based and other Technologies, Springer Science and Business Media, New York, 2013.
[16] Verma, M.; Srivastava, M.; Chack, N.; Diswar, A.-K.; Gupta, N.: A comparative study of various clustering algorithms in data mining, International Journal of Engineering Research and Applications (IJERA), 2(3), 2012, 1379-1384.
[17] Wu, Y.-F.; Wong, Y.-S.; Loh, H.-T.; Zhang, Y.-F.: Modelling cloud data using an adaptive slicing approach, Computer-Aided Design, 36(3), 2004, 231-240. http://doi.org/10.1016/S0010-4485(03)00097-6
[18] Yang, P.; Li, K.; Qian, X.: Topologically enhanced slicing of MLS surfaces, Journal of Computing and Information Science in Engineering, 11(3), 2011. http://doi.org/10.1115/1.3615683
[19] Yang, P.; Qian, X.: Adaptive slicing of moving least squares surfaces: toward direct manufacturing of point set surfaces, Journal of Computing and Information Science in Engineering, 8(3), 2008. http://doi.org/10.1115/1.2955481
[20] 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
[21] Zhang, D.; Yang, P.; Qian, X.: Adaptive NC path generation from massive point data with bounded error, Journal of Manufacturing Science & Engineering, 131(1), 2009. http://doi.org/10.1115/1.3010710