COMPUTER-AIDED DESIGN & APPLICATIONS, 2018
VOL. 15, NO. 2, 227–237
https://doi.org/10.1080/16864360.2017.1375673

Surface detection and modeling of an arbitrary point cloud from 3D sketching

Ariel Schwartz [a], Ronit Schneor [a], Gila Molcho [a] and Miri Weiss Cohen [b]

[a]Technion-Israel Institute of Technology, Israel; [b]Braude College of Engineering, Israel

ABSTRACT
This work describes a method for converting unorganized point cloud data from 3D sketching into an explicit 3D model, contributing to closing the knowledge gap in transferring 3D sketches from a conceptual design to the detailed design. The main challenge in the work is processing an unorganized point cloud where the points fill the volume of the object and do not only lie on the surface, as they would in most scanned data. The scope of the work is a single object, genus 0. The 3D model is constructed by building a loft feature through curves generated by slicing the point cloud. Firstly, the point cloud is sliced and points worked onto slicing planes; secondly, the external points that describe the contour of each slice are isolated using alpha shapes; lastly, smooth curves are created through the external points, and a loft is generated.

KEYWORDS
3D sketching; point cloud; 3D surface modeling

1. Introduction

This work aims to advance 3D sketching-to-production capabilities. Multiple methodologies and tools exist to sketch products at their conceptual phase. Amongst them are CAD systems and various 3D virtual sketching capabilities, which have emerged in recent years [2,15,17,25,27–30]. However, there still exists a significant knowledge gap in transferring these initial sketches for detailed design or realizing products based on these abstract sketches. In this work an algorithm is presented that contributes to closing this gap.

A designer may use 3D sketching as a tool in the early stages of work, but the technology does not yet allow the user to bring sketches directly into CAD models. Whether a part is to be machined or 3D printed, it is generally represented as a 3D model prior to manufacturing. The motivation for this work is enabling the transfer of the output from the 3D sketcher to the designer for detailed design.

The aim of this work is to develop a solution for converting a 3D point cloud, created by an augmented reality 3D sketching system, into an explicit 3D model, while incorporating as much design intent as possible. The work presented aims to suffice CAD design features, focusing on extrude and revolve, therefore our examples adhere to this content.

While several algorithms exist that convert a point cloud created by a 3D scanner to a 3D model, the challenges of converting a 3D sketch output has yet to be addressed. Unlike the 3D sketcher, modern 3D scanners sample points from the objects boundary, largely providing an ordered point cloud.

In the 3D sketching process, users sketch the object in space. They do not necessarily only draw the external surface of the desired object, rather, they might fill the object with internal points as well (filling in the object). However, there is no indication of whether the points lie within the object, outside the object, or on the surface. The main challenge of this work is, therefore, to construct a 3D model from an unordered point cloud that includes not only points from the boundary but internal and external (noise) points as well.

3D sketching systems allow users to design bodies and assemblies of unlimited complexity. This work focuses on understanding and overcoming the main challenges involved with conversion of data from the augmented reality to a 3D model. Therefore, the scope of this work is a single object, with genus-0 surfaces.

Much research has been done regarding mesh reconstruction from an unorganized point cloud, where the points are created from a scanner. Schnabel et al. [20] developed an algorithm to detect common shapes in an unorganized point cloud. Using the detected shapes, they build the structure into a concise CAD model. Kyriazis et al. [11] divide their point cloud data into thin slices, and work the points within each slice onto a plane. Then,

CONTACT Ariel Schwartz arielgilliandayan@gmail.com; Ronit Schneor schneor@technion.ac.il; Gila Molcho gila@technion.ac.il; Miri Weiss Cohen miri@braude.ac.il

they find the convex hull on small regions of the worked point cloud, and combine the results to obtain an interpolation of the object's shape. Woo, Kang, Wang, and Lee [26] propose an algorithm using the octree-based 3D grid method to handle large numbers of unordered sets of points data. Their method enables them to extract the edge neighborhood points, while considering the geometric shape of the part. Tang et al. [23]and survey the state-of-the-art methods for Building information models from laser scanned point clouds and discuss their potential application to automated as-built BIM creations, moreover, Bosche [2], details a method for construction purposes.

One of the existing methods to extract the surface points in a 3D point cloud where the points do not necessarily lie on the surface is the alpha-shape algorithm. Edelsbrunner and Mucke [5] derived the use of alpha shapes to isolate the boundary points in a point cloud and reconstruct the shape of the object. Xu and Harada [27] and Guo et al [6] use alpha shape for surface reconstruction. Another option for construction of the CAD model is the use of mesh triangulation. Lin, Tai, and Wang [12] present an algorithm for constructing a triangle mesh using the sampling uniformity degree at each sampling point as an intrinsic property of the point cloud. This way, they avoid the use of a user-specified parameter to control the mesh, minimizing the error in reconstruction.

Different approaches can be found in Sam et al. [19], which propose a method for representing a curve skeleton from point cloud by estimating centers of antipodes inside the shape, then filtered and then a one- dimensional Moving Least Squares (MLS) is implemented to build a thin point cloud and then a smooth curve skeleton. Zhao et al [29], use machine learning classifiers with three different shape descriptors, Light-Field Descriptors, Angular Radial Transform (ART), and Zernike Descriptors (ZD). They evaluated the classification performance of different classifiers and combined them with the different shape descriptors on point cloud examples. Masuda et al [15,16], claim that Standard Deviation of residuals in surface fitting has various errors according to the sizes, distances, and materials of the scanned objects. Their work investigates these distributions, resulting a prediction functions, which are used for surface extraction. A moving parabolic approximation (MPA) to reconstruct a triangular mesh was done by [28], and an automatic method for outlier detection based on the principle of majority voting is found in [24].

Among many slicing methods found in literature, Zhong et al. [30], propose a series of algorithms including Inverse Distance Square method (IDS) for slicing, extrapolation method for vertical slicing. The use of NURBS as contour curve was used for reconstruction of profile data points. Liu[13], presents an algorithm which group all points belonging to the same functional surface, moreover, a B-spline surface is fitted to these points so as to generate an editable NURBS surface. Oropalloa, Piegl, Rosen and Rajab [17], use a different approach by using the original NURBS model and converting it into a point cloud, based on layer thickness and accuracy requirements, for direct slicing. Their work proves efficiency in computational requirement, which is done error free. Preprocessing of slicing surface generation with focusing on reliable normal estimation is done by Huang et al. [7], their method uses a local optimal projection operator to improve local PCA, furthermore, an iterative normal estimation and a priority-driven normal propagation scheme is used. Robust segmentation for multiple manifolds is presented in [10].

2. System overview

To establish the overview, a basic understanding of the 3D sketching system and its output is required. As a reference for this work, the 3D sketching system used, is the Immersive Freehand 3D Sketching on Air in Full Object Scale [20].

The system uses a number of Vicon cameras, set up around the room, which can capture the motion of reflective markers on the head and hand of the user (Fig. 1). The output from the system is a point cloud in Cartesian coordinates. For the purpose of this work, only sample point clouds were taken that fit within the scope: a single object, genus-0.

3. Related work

3.1. K-means clustering

One commonly used clustering algorithm is K-means clustering. K-means is a method to partition a set of

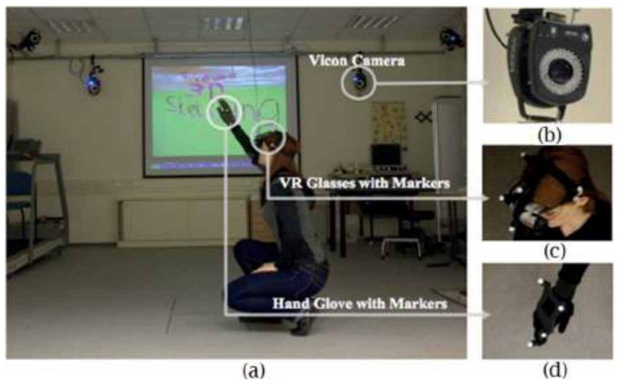


Figure 1. 3D sketching system, from Shain, [21]. (a) The room, (b) Vicon camera, (c) VR glasses with reflective markers, (d) Hand glove with reflective markers.

points into k number of clusters. The K-means algorithm consists of two separate phases: in the first phase it calculates the k centroids and in the second phase, measures the distance from each point to the cluster that has the nearest centroid from the respective data point.

There are different methods for defining the distance to the nearest centroid, one of which is Euclidean distance (see the detailed review in [3]). Once the grouping is found, the algorithm recalculates the new centroid of each cluster. Based on that centroid, a new Euclidean distance is calculated between each center, and each data point. The points in the cluster with minimum Euclidean distance are assigned [3,18]. Each cluster in the partition is defined by its member objects and by its centroid.

The centroid for each cluster is the point to which the sum of distances from all the objects in that cluster is minimized.

The algorithm for k-means clustering follows:

1. Initialize number of cluster k and center.
2. For each point, calculate the Euclidean distance d between the center and each point, using equation 1:

$$d = \|p(x, y) - C_k\| \tag{1}$$

3. Assign all the points to the nearest center, based on distance d.
4. After all points have been assigned, recalculate the new position of the center using equation 2:

$$C_k = \frac{1}{k} \sum_{y \in C_k} \sum_{x \in C_k} p(x, y) \tag{2}$$

5. Repeat the process until it satisfies a defined error value.
6. Reshape the cluster points.

This algorithm can also work for a large number of variables, but produces a different cluster result for different numbers of clusters. So it is necessary to initialize the proper number of clusters, k. Different values of initial k and centroids would result different clusters.

3.2. Alpha shapes

The concept of alpha shapes developed by Edelsbrunner and Mucke [5] formalizes the intuitive notion of "shape" for spatial point sets. The alpha shape is a mathematically well-defined generalization of the convex hull and is a subgraph of the Delaunay triangulation [4,9]. It is defined intuitively as: α-shapes are a generalization of the convex hull of a point set. Let S be a finite set in $\mathbb{R}^3$ and α a real number with $0 < \alpha \leq x$. For $\alpha =$ x, the α-shape is identical to the convex hull of S. However, as α decreases, the α-shape shrinks to cavities.

The formal detailed definition for alpha shapes is found in [1,6], summarized as follows:

For $0 < \alpha \leq x$, let an α-ball be an open ball with a radius α. An α-ball b is empty if $b \cap^{S} = 0$.

Any subset $T \subseteq S$ of size $|T| = k + 1$, with $0 \leq k \leq 3$, defines a k-simplex σ_T, which is the convex hull of T.

The alpha shapes bring spheres of a defined probe radius as close to the point cloud as possible without allowing any of the points to enter a sphere. All the points that come in contact with a sphere are defined as the external points in the point cloud.

3.3. Octree

An octree [8] is a tree data structure used to partition a three-dimensional space by recursively subdividing it into eight octants [8] unit is labeled as full, void, or mixed, based on the information in its space. Any unit with a mixed value is again subdivided into eight. This process continues until all unit cubes within an octant have the same labels associated with it: either full or void.

3.4. Loft surfaces

A loft surface is a special case of homotopy, a straight-line homotopy. Shinagawa et al. [22] define a loft as the surface patch between adjacent contours that is generated by connecting corresponding points on each contour by homotopy. The homotopic model of a loft surface is the generalization of the following:

$$F(x, t) = (1 - t)f(x) + tg(x) \tag{3}$$

Let the lower and upper B-Spline curve be expressed by the maps $f, g : X \to R\hat{}3$, where $X = [x_0, x_1] \subset R$. For simplicity, let the lower contour line be on the plane z = 0 and the upper one on z = 1, and $X = I$. Thus, the two points $P(p_x, p_y)$ and $Q(q_x, q_y)$ are connected by homotopy F, meaning that there exists $x \in I$ such that $F(x, O) = f(x) = (p_x, p_y, O)$ and $F(x, 1) = g(x) = (q_x, q_y, 1)$. are defined. Loft surfaces are commonly used in CAD software.

4. Proposed approach

There are various approaches to processing point cloud data. It was established that in the processing stage, the surface points would need to be isolated from the point cloud. Two approaches were investigated to achieve this goal: octree representation and alpha shapes.

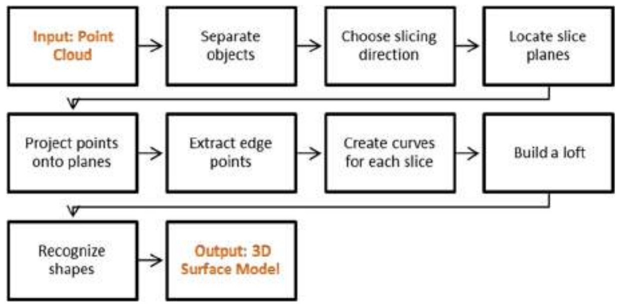


Figure 2. A scheme of the proposed approach.

By using an octree to organize the point cloud, the boundaries of the cloud could be found. The point cloud would be broken down into bins using an octree, such that if a bin contains points, it is defined as "full", and if it is empty, it is defined as "void". The boundaries of the cloud are then defined as full bins that have at least one void neighbor. One problem that arose with this approach was the varying density of the point cloud, as a result of the 3D sketching. Using this method, a void unit could be detected in a low-density area in the center of the point cloud. Therefore, "boundaries" of the point cloud would be detected in the center of the volume, resulting in incorrect processing of the data.

Using the alpha-shape algorithm, the point cloud is enveloped from the outside, which—in the case of a genus-0 object—is desired. We chose to use this methodology as the basis, and work towards improving the results. One of the major disadvantages observed in the alpha-shape method was that to obtain an accurate depiction of the point cloud, a small probe radius was required, resulting in a very uneven surface. We determined that one way to improve the results is to break the problem down into a multi-stage solution incorporation of both 2D and 3D analysis. The selected approach to improve precision was to slice the point cloud and analyze each slice (2D), and then to reconstruct the 3D model from the analyzed slices. This approach allowed for improved preservation of the overall shape of the object when extracting the boundary information.

Fig. 2 is a block diagram depicting the basic idea of our approach, in stages, with the relevant methodology used for performing the task defined in each stage.

4.1. Input

The input is a set of non-ordered points in space. The points can sit anywhere within the object, or on the surface. There is no predetermined surface defining the shape of the object. For the purpose of testing, a sample point cloud was produced by generating random points within three overlapping spheres of varying radii (Fig. 3).

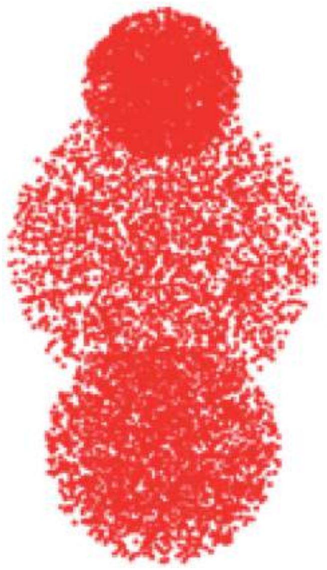


Figure 3. Sample point cloud.

This point cloud represents an object within the scope of the work (single object, genus-0 surface) as an example of what may be generated by a user of the augmented reality.

4.2. Separate bodies

As mentioned above, the scope of this work is single object, genus-0. Therefore, the preliminary stage of separating distinct bodies is a prerequisite for using the slice and loft algorithm. This stage was completed using k-means clustering, this function detects the clusters in a point cloud, given the number of clusters as input. Assuming the number of objects in the point cloud is ten or less, the k-means function runs iteratively, up to ten times, increasing the assumed number of clusters by one at each iteration. The solution with minimum error can determine how many bodies are in the point cloud, and then the clusters can be separated.

4.3. Find the slicing direction

The direction in which the object is sliced can greatly affect the resulting model. Fig. 4 demonstrates this by slicing an L-shaped point cloud in three different directions and applying the loft algorithm.

Several options exist for finding the optimal slicing direction. The first option is that the user manually chooses the direction by interactively aligning an axis by which they would like the object to be sliced. The second option is to automatically seek a transformation of the object that minimizes a target function. Another option is to choose from the *x*, *y*, and *z* directions. Since the point

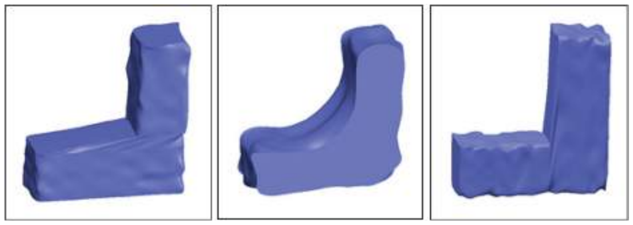


Figure 4. L-shape sliced in three directions.

cloud at hand comes from a 3D sketch, the user generally draws the object in the Cartesian coordinate system. Thus, one of the three directions is likely the optimal slicing direction. The system would then evaluate which of the three directions gives the best depiction of the object. This process proves to be a challenging one, seeing as there is no given information about the object.

For this work, the slicing direction was manually chosen as *x, y, or z* based on an interactive decision of the designer for each sample. In the example depicted in Fig. 4, the *z* direction was chosen, such that all the slices' contours would be circular in shape and concentric to one another.

4.4. Locate slice planes

The locations of the slice planes has a large effect on the final result, as missing changes in object curvature or extreme/key points can easily occur, significantly altering the results. Thus, it is crucial that the slices are taken at strategic points. Many modern 3D interest point detection algorithms define functions based on local extrema points [4]. In this work we take a similar approach.

Step 1: The cloud of points is transformed such that the slicing direction coincides with the z-axis. This makes the set of points easier to work with in the subsequent stages.

Step 2: A 3D alpha-shape algorithm (with a small probe radius) is applied to the object to get a general understanding of the object's surface shape. Fig. 5 shows the original set of data points (red), and the surface points, which were extracted using alpha shapes (blue). From this point on, for the purpose of finding the slice planes, only the surface points (blue) are used.

Step 3: A central axis in the z direction (or slicing direction) is found, such that the object surrounds the axis (Fig. 5(a)). This can be accomplished using a least squares method [14], or by simply taking an average of the x and y points. We selected to implement the average x and y methodology.

Step 4: The absolute value of the distance from each point to the central axis is calculated. At this stage, the data can be represented by a graph of the absolute distance to the central axis, versus the z location of each point (Fig. 5(b)).

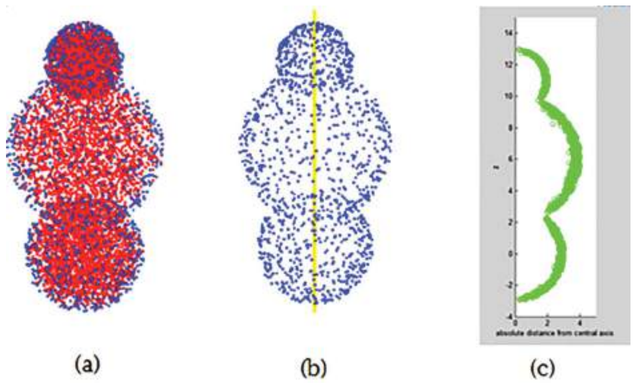


Figure 5. (a) Surface points (blue) extracted using alpha shapes, (b) Surface points and central axis, (c) Absolute distance of the surface points to the central axis.

Step 5: The local maxima and minima are found to ensure that the object is sliced at all the key/extreme locations. A number of methodologies can be used to find these points, such as curve approximation. Since the data cannot always be approximated to a curve, the following iterative method for slicing the object was derived and depicted in Fig. 6.

When performing the iterative process, the approximate size of the slices must be determined. This is done according to the number of points per slice using k-means. In this example, slice size was chosen based on an average of 500 points per slice.

This process guarantees that all local maxima and minima are located, and the spaces between them are divided into slices. In the event of an object whose cross-section does not change much along the central axis (such

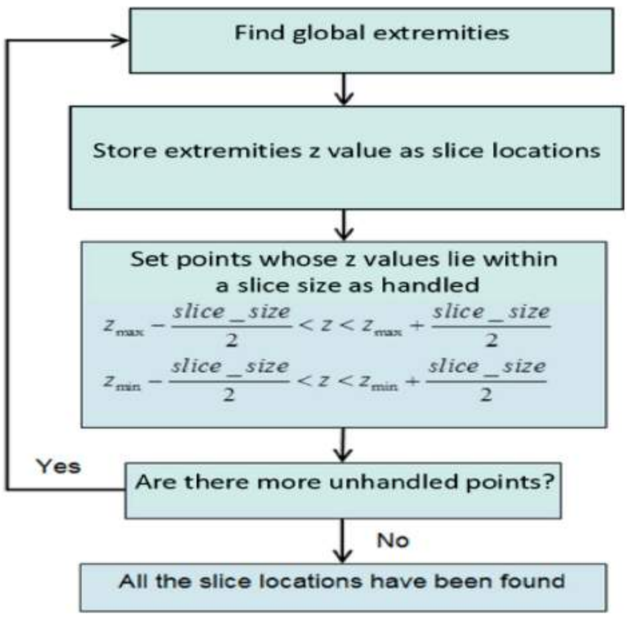

Find global extremities
Store extremities z value as slice locations
Set points whose z values lie within a slice size as handled
$z_{max} - \frac{slice\_size}{2} < z < z_{max} + \frac{slice\_size}{2}$
$z_{min} - \frac{slice\_size}{2} < z < z_{min} + \frac{slice\_size}{2}$
Are there more unhandled points?
Yes
No
All the slice locations have been found

Figure 6. Iterative process for finding the slice locations.

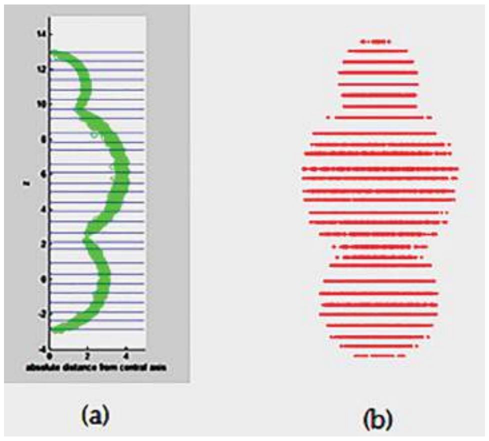


Figure 7. (a) Location of slice planes, (b) Side view of the worked point cloud onto slice planes.

as an extruded object), this algorithm results in uniform slicing. Fig. 7(a) shows the resulting slice locations after executing the iterative slicing process on the example object.

4.5. Work points onto planes

After the locations of the slices are determined, the points are allocated to their slices. At this stage, the points used are not those extracted from the 3D alpha shapes, but the original (red) input points. The points allocated to a slice are those surrounding it on both sides, halfway to the next slice plane. Parallel working in the slicing direction is used to work the points onto their respective planes (Fig. 7(b)).

4.6. Extract the edge points

On each slice, the 2D alpha-shape algorithm is applied to extract the edge points from the resulting worked points. At this stage, a probe radius must be chosen for the alpha-shape algorithm. This probe radius greatly affects the resulting contour. For this work the radius was chosen manually. For more complex bodies, such as those with planar, concave, and convex sections, it may be preferable to use a different probe radius on each slice depending on its shape. Fig. 8(a) shows a side view of the worked points (red) and the edge points that were extracted (blue). Fig. 8(b) shows only the final extracted edge points (blue) on their slice planes.

4.7. Construct the 3D Model

The input for this stage is the edge points of each slice. In the world of CAD, there are several standard modeling techniques to build the features of an object. They

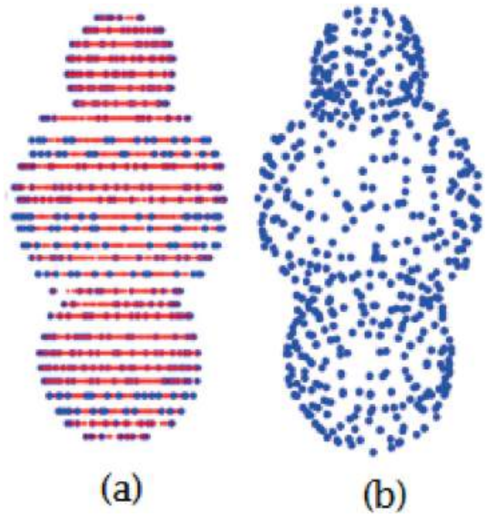


Figure 8. (a) Side view of surface points extracted using 2D alpha shapes, (b) Final set of points representing the surface.

include revolved, extruded, swept, and lofted features. A loft was chosen as the method for the construction of the CAD model. This method was selected because it is a natural transition from slicing the object to building a CAD model. It also allowed for the most flexible reconstruction of the object, so that the algorithm could be used on many different point clouds.

In the CAD system, each set of edge points is approximated to a closed spline curve representing the profile of the object on each given plane (Fig. 9(a)). By transitioning between profiles to create a non-analytic surface, the lofted object is created (Fig. 9(b)).

The model obtained using the slicing method gives an excellent depiction of the overall shape of the object,

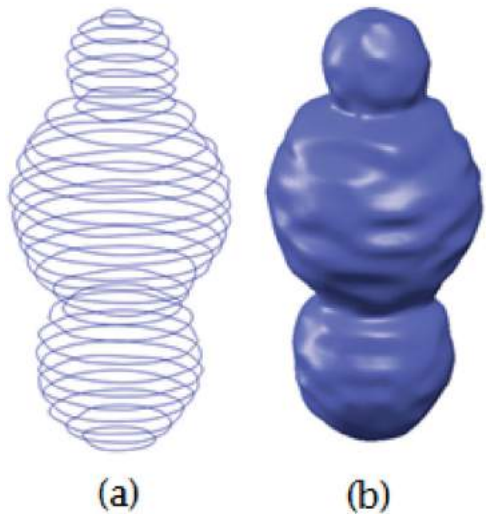


Figure 9. (a) Curves through edge points, (b) Final lofted CAD model shape recognition.

but has an uneven surface. This might be improved by using shape recognition techniques on the 2D slice planes.

In 2D, curve recognition techniques are applied to the edge points that were extracted on the slice planes. This is achieved by least square fitting of known curves such as circle, oval, or rectangle to the edge points. This reduces much of the roughness in the resulting 3D model. In the example point cloud built from three spheres, the desired shapes on the 2D planes are known to be circles. After the edge points were isolated, a circle was fitted to each of the slices. The loft was then built from the circular sections. Comparing this model to the model created from the same point cloud without fitting the circles, it can be seen that this greatly improves the surface quality of the 3D model (Fig. 10).

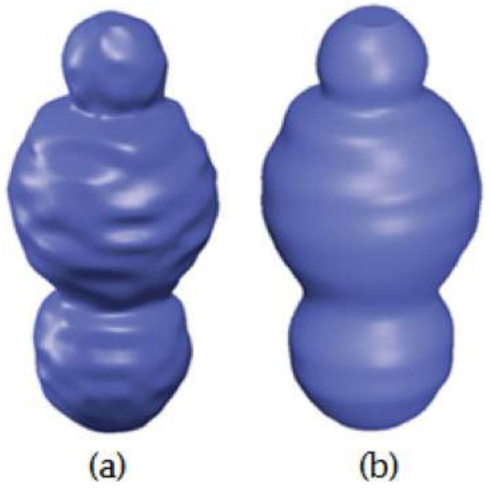


Figure 10. 3D model. (a) Without curve recognition, (b) With 2D curve recognition.

5. Results

The results of the methodology were tested in a number of ways.

5.1. Results quantification

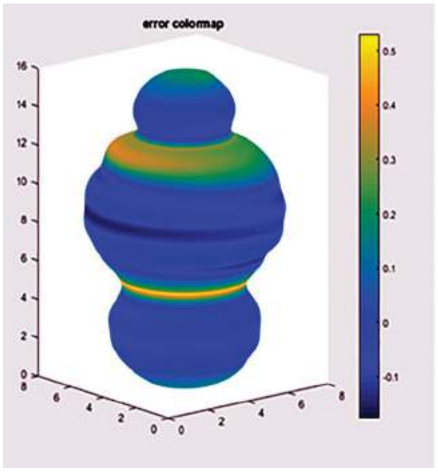
A method was developed to quantify the results. For the point clouds that were generated, the initial desired object was used as the goal surface. The error was measured as the normal distance, *d*, between the resulting surface and the goal surface. Positive and negative errors represent points lying outside and inside the desired surface, respectively.

Fig. 11 shows a color map and a histogram of the error, from the sample point cloud made from three spheres of radius 2, 3, and 4 (Fig. 11(b)). It can be seen that the maximum errors were obtained at the intersecting planes between the spheres. This result is due to the concave shape of the object in those areas. These intersection points are detected as key features when the slice locations are chosen. Then, the points from above and below the slice plane are worked onto the surface. Since the radius increases as we move away from these points, the resulting worked point cloud has a larger radius than the intersection plane. Thus, the resulting spline curves created on those planes are larger than the desired surface, contributing to the large error.

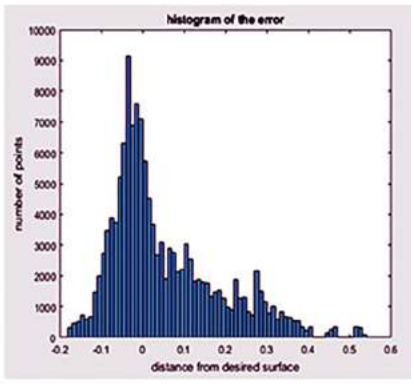
The histogram shows that the majority of the points are a relatively small distance from the goal surface, which is desired.

5.2. Check for distortion

The algorithm was tested on a set of data that lies on the surface of a known shape, similar to scanned objects. A



(a)



(b)

Figure 11. (a) Error color map, (b) Error histogram.

point cloud was generated where all the points lie on the surface of a basic shape, and the slice and loft algorithm was applied. In this manner, it is possible to quantify the distortion caused by the slice and loft algorithm.

First, a point cloud lying on the surface of a sphere of radius 3 was generated. The resulting shape, along with the error quantification data, can be seen in Fig. 12.

The largest error occurred at the top and bottom portions of the sphere. Additionally, a ripple effect is seen, which is due to fitted circles not being perfectly concentric.

5.3. Comparison to 3D alpha-shape method

Two point clouds were generated to compare the results to those obtained using alpha shapes. The first is a planar object in the form of a three-dimensional L shape (Fig. 13). The second is a curved object, made of three overlapping spheres of different sizes

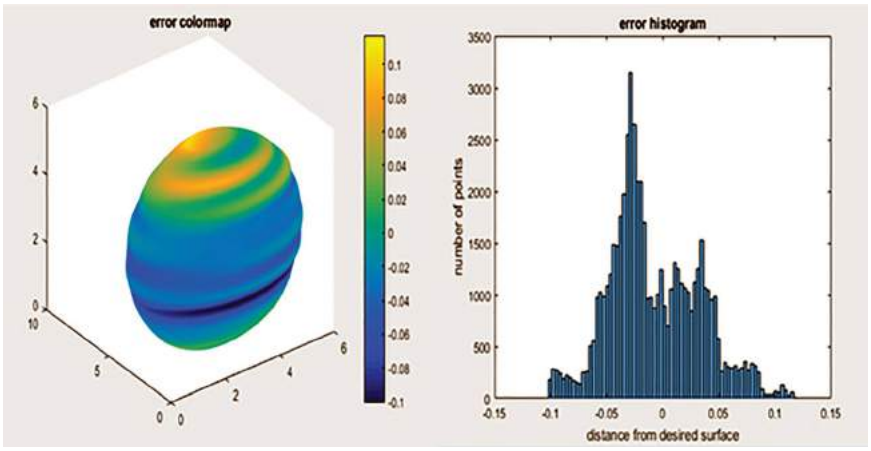


Figure 12. Lofted sphere from the scanned point cloud.

Planar object:

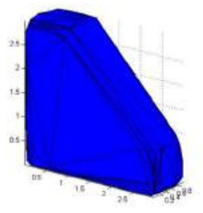
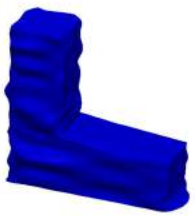
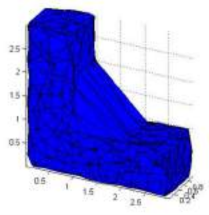
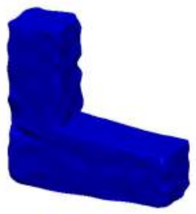
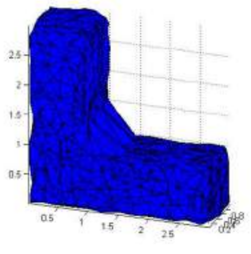
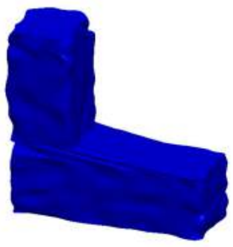
Probe Radius	3D Alpha Shapes	Slice and Loft
Infinity		
1		
0.5		

Figure 13. Planar object in the form of a three-dimensional L shape comparison.

(Fig. 14). The slice and loft algorithm and the alpha-shape algorithm were applied to each point cloud three times, each with a different probe radius. The 2D shape recognition was not performed here, to better understand the effect of the probe radius on the resulting object.

It can be seen immediately that both of the bodies' overall shapes are much better preserved using the slice and loft algorithm, especially when comparing the methods with a larger probe radius.

In the case of a smaller probe radius, the 3D alpha-shape algorithm may include some points from within the object as surface points, resulting in an uneven surface. The surface roughness is improved slightly using the derived model, because running the curve through xyz points and loft features results a smooth continuous solution. Additionally, when the points from within a slice are worked onto the plane, the points are more concentrated, which can improve the validity of the shape extracted using the alpha-shape algorithm.

Another point that should be noted is the effect of the probe radius on the shape. In every object, an optimal probe radius can be found to get results that are similar to the desired object yet not too uneven. In circular objects, a larger probe radius is generally desired. In planar objects, a smaller radius is desired, but not too small. In the L-shape, depicted in Figure 13, the best representation of the object is given with a probe radius of 1. A larger probe radius does not capture the corners of the rectangular sections well, and a smaller probe radius gives a more uneven result. In the curved object, an infinite probe radius gives the best representation, and with a radius of 0.5 the model is very uneven and poorly represents the object.

5.4. Example of methodology results

As an example, a point cloud was generated in the shape of a lamp. The base of the lamp is cylindrical, the stand rectangular, and the lampshade is cone-shaped (Fig. 15).

The slice and loft algorithm successfully constructed a 3D model from the point cloud. The loft effectively transitioned between the circular and rectangular slices. 2D shape recognition techniques were then manually applied to the slice curves, and the loft was re-created.

Curved object:

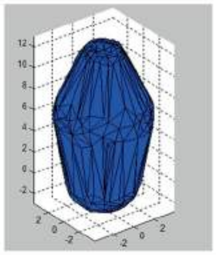
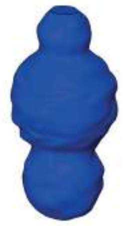
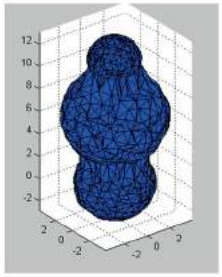
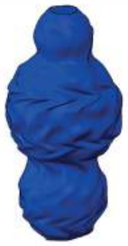
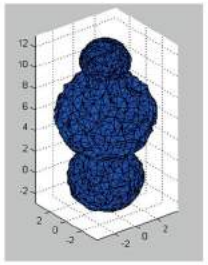
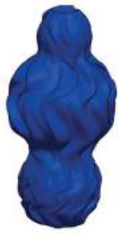
Probe Radius	3D Alpha Shapes	Slice and Loft
Infinity		
1		
0.5		

Figure 14. Three overlapping spheres comparison.

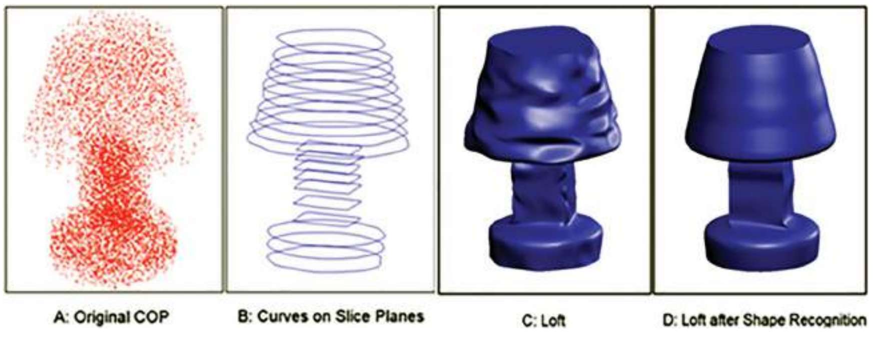


Figure 15. Lamp example.

6. Conclusions

In this work, the slice and loft algorithm was developed, which successfully converts a 3D point cloud into an explicit 3D model. The method begins by slicing the point cloud and working the points within a slice onto a plane. Then, it uses the 2D alpha-shape algorithm to extract the edge points. From there, a smooth spline curve through the edge points is created on each slice plane. Finally, a loft is produced from the resulting curves.

In some cases where surface quality of the resulting model was insufficient, we improved it by adjusting the probe radius in the alpha-shape stage, depending on the shape of the contour. Additionally, shape recognition could be implemented in the 2D or 3D stages to convert the object into a collection of known predefined shapes.

This algorithm may have great implications in the field of virtual reality and 3D printing. In the future, a user of 3D sketching may be able to quickly and easily print their designs with the click of a button.

ORCID

Ariel Schwartz http://orcid.org/0000-0001-6313-9171
Ronit Schneor http://orcid.org/0000-0001-5486-6529
Gila Molcho http://orcid.org/0000-0002-8878-4027
Miri Weiss Cohen http://orcid.org/0000-0001-5250-1016

References

[1] Bernardini, F.; Bajaj, C.: Sampling and Reconstructing Manifolds using Alpha-Shapes, *Computer Science Technical Reports*, 1997.

[2] Bosché, F.: Automated recognition of 3D CAD model objects in laser scans and calculation of as-built dimensions for dimensional compliance control in construction, *Advanced Engineering Informatics*, 24(1), 2010, 107–118. https://doi.org/10.1016/j.aei.2009.08.006

[3] Charu C.; Aggarwal, C.; Reddy, K.: Data Clustering: Algorithms and Applications, Chapman & Hall/CRC Data Mining and Knowledge Discovery Series, 2016.

[4] Dutagaci, H.; Cheung, C. P.; Godil, A.: Evaluation of 3D interest point detection techniques via human-generated ground truth, *The Visual Computer*, 28(9), 2012, 901–917.doi:10.1007/s00371-012-0746-4

[5] Edelsbrunner, H.; Mücke, E. P.: Three-dimensional alpha shapes, *ACM Trans. Graph*, 13(1), 1994, 43–72. https://doi.org/10.1145/174462.156635

[6] Guo, B.; Menon, J.; Willette, B.: Surface reconstruction using alpha shapes, *Computer Graphics Forum*, 16(4), 1997, 177–190. https://doi.org/10.1111/1467-8659.00178

[7] Huang, H.; Li, D.; Zhang, H.; Ascher, U.; Cohen-Or. D.: Consolidation of unorganized point clouds for surface reconstruction. *ACM Trans. Graph*. 28, 5, Article 176 (December 2009), DOI: https://doi.org/10.1145/1618452.1618522

[8] Jackins, C. L.; Tanimoto, S. L.: Oct-trees and their use in representing three-dimensional objects, *Computer Graphics and Image Processing*, 14(3), 1980, 249–270. https://doi.org/10.1016/0146-664X(80)90055-6

[9] Kuo, C.; Yau, H.: A Delaunay-based region-growing approach to surface reconstruction from unorganized points, *Computer Aided Design* 37(8), 2005, 825–835. http://doi.org/10.1016/j.cad.2004.09.011

[10] Kustra, J.; Jalba, A.; Telea, A.: Robust segmentation of multiple intersecting manifolds from unoriented noisy point clouds, *Computer Graphics Forum*, 33, 2014, 73–87. https://doi.org/10.1111/cgf.12255

[11] Kyriazis, I.; Fudos, I.; Palios, L.: Detecting Features from Sliced Point Clouds, Proceedings of the Second International Conference on Computer Graphics Theory and Applications, GRAPP, 2007.

[12] Lin, H.; Tai, C.; Wang, G.: A mesh reconstruction algorithm driven by an intrinsic property of a point cloud, *Computer-Aided Design*, 36(1), 2004, 1–9. https://doi.org/10.1016/S0010-4485(03)00064-2

[13] Liu, X.: Functional surface reconstruction from unorganized noisy point clouds, *Computer-Aided Design and Applications*, 12(3), 2015, 366–372. DOI:10.1080/16864360.2014.981467 https://doi.org/10.1080/16864360.2014.981467

[14] Lloyd, S. P.: Least squares quantization in PCM, *IEEE Transactions on Information Theory*, 28, 1982, 129–137. https://doi.org/10.1109/TIT.1982.1056489

[15] Masuda, H.; Niwa, T.; Tanaka, I.; Matsuoka, R.: Reconstruction of polygonal faces from large-scale point-clouds of engineering plants, *Computer-Aided Design and Applications*, 12(5), 2015, 555–563. DOI:10.1080/16864360.

2015.1014733 https://doi.org/10.1080/16864360.2015.1014733

[16] Masuda, H.; Tanaka, I,; Enomoto, M.: Reliable surface extraction from point-clouds using scanner-dependent parameters, *Computer-Aided Design and Applications*, 10(2), 2013, 265–277. https://doi.org/10.3722/cadaps.2013.265-277

[17] Oropallo, W.; Piegl, L. A.; Rosen, P.; Rajab, K.: Generating point clouds for slicing free-form objects for 3-D printing, *Computer-Aided Design and Applications*, 14(2), 2017, 242–249 https://doi.org/10.1080/16864360.2016.1223443

[18] Redmond, S. J.; Heneghan, C.: A method for initialising the K-means clustering algorithm using kd-trees, *Pattern Recognition Letters*, 28(8), 2007, 965–973. https://doi.org/10.1016/j.patrec.2007.01.001

[19] Sam, K.; Kawata, H.; Kanai, T.: A robust and centered curve skeleton extraction from 3D point cloud, *Computer-Aided Design and Applications*, 9(6), 2012, 869–879. https://doi.org/10.3722/cadaps.2012.869-879

[20] Schnabel, R.; Wahl, R.; Klein, R.: Efficient RANSAC for point-cloud shape detection, *Computer Graphics Forum*, 26(2), 2007, 214–226. https://doi.org/10.1111/j.1467-8659.2007.01016.x

[21] Shain, O.: Immersive Freehand 3D Sketching on Air in Full Object Scale: System Development and User Studies, 2014.

[22] Shinagawa, Y.; Kunii, T.L.: The homotopy model: A generalized model for smooth surface generation from cross sectional data, *The Visual Computer*, 7, 1991, 72–86. https://doi.org/10.1007/BF01901178

[23] Tang, P.; Huber, D.; Akinci, B.; Lipman, R.; Lytle, R.: Automatic reconstruction of as-built building information models from laser-scanned point clouds: A review of related techniques, *Automation in Construction*, 19, 2010, 829–843. https://doi.org/10.1016/j.autcon.2010.06.007

[24] Wang, Y.; Feng, H.-Y.: Outlier detection for scanned point clouds using majority voting, *CAD Computer Aided Design*, 62, 2015, 31–43. https://doi.org/10.1016/j.cad.2014.11.004

[25] Wang, Y.; Li, H.; Ning, X.; Shi, Z.: A new interpolation method in mesh reconstruction from 3D point cloud. In *Proceedings of the 10th International Conference on Virtual Reality Continuum and Its Applications in Industry*, VRCAI '11, ACM, New York, NY, USA, 2011, 235–242. https://doi.org/10.1145/2087756.2087790

[26] Woo, H.; Kang, E.; Wang, S.; Lee, K. H.: A new segmentation method for point cloud data, *International Journal of Machine Tools and Manufacture*, 42(2), 2002, 167–178. https://doi.org/10.1016/S0890-6955(01)00120-1

[27] Xu, X.; Harada, K.: Automatic surface reconstruction with alpha-shape method, *Vis. Comput.*, 19(7–8), 2003, 431–443. https://doi.org/10.1007/s00371-003-0207-1

[28] Yang, Z.; Seo, Y.-H.; Kim, T.-W.: Adaptive triangular-mesh reconstruction by mean-curvature-based refinement from point clouds using a moving parabolic approximation. *Computer-Aided Des.*, 42(1), 2010, 2–17. https://doi.org/10.1016/j.cad.2009.04.014

[29] Zhao, X.; Ilieş, H. T.: Learned 3D shape descriptors for classifying 3D point cloud models, *Computer-Aided Design and Applications*, 2016. https://doi.org/10.1080/16864360.2016.1257192

[30] Zhong, S.; Yang, Y.; Huang, Y.: Data slicing processing method for RE/RP system based on spatial point cloud data, *Computer-Aided Design and Applications*, 11(1), 2014, 20–31. https://doi.org/10.1080/16864360.2013.834133