A RANSAC Based CAD Mesh Reconstruction Method Using Point Clustering for Mesh Connectivity

Sørensen, Thor A.; Mark, Natalie; Møgelmose, Andreas

Published in:
International Conference on Machine Vision and Applications (ICMVA 2021)

DOI (link to publication from Publisher):
10.1145/3459066.3459076

Publication date:
2021

Document Version
Accepted author manuscript, peer reviewed version

A RANSAC Based CAD Mesh Reconstruction Method Using Point Clustering for Mesh Connectivity

Thor A. Sørensen
Aalborg University
Department of Electronic Systems
Aalborg, Denmark
tasa17@student.aau.dk

Natalie Mark
Aalborg University
Department of Electronic Systems
Aalborg, Denmark
nmark15@student.aau.dk

Andreas Møgelmose
Aalborg University
Department of Media Technology
Aalborg, Denmark
anmo@create.aau.dk

ABSTRACT
Mesh reconstruction is widely used in industrial reverse engineering and other 3D scanning applications. Many of the methods used can accurately and swiftly reconstruct a model from a given input scan dataset – however, in some applications, simplification of the output mesh may be needed to render further computation tractable. This paper presents a method based on a recursive RANSAC point removal approach, where an input point cloud is deconstructed into a set of convex point clusters, each corresponding to a mesh face of the final output model. The combined use of cluster analysis and RANSAC model extraction allows the method to operate on a relatively small number of possible vertex candidates for the final mesh, leading to an output mesh with relatively few faces compared to other approaches.

CCS Concepts
• Computing methodologies → Artificial intelligence → Computer vision → Computer vision problems → Reconstruction
• Computing methodologies → Machine learning → Learning paradigms → Unsupervised learning → Cluster analysis
• Applied computing → Physical sciences and engineering → Engineering → Computer-aided design

Keywords
Mesh reconstruction; RANSAC; Industrial reverse engineering; Point cloud data; PCL; 3D Scanning;

1. INTRODUCTION
For two decades, real-time or near-real-time 3D model acquisition technology has been available for research purposes [31]. In the last decade with the release of relatively inexpensive 3D acquisition tools such as the Kinect and similar commercial devices, this technology has become available for wide use in industrial and commercial applications. A common use case of 3D scanning technology is construction of a 3D model or mesh based on point cloud data from a scanned object or scene. This task is commonly referred to as the mesh reconstruction problem. For example, mesh reconstruction is applied in scanning machines in the healthcare sector [32], scene reconstruction from LiDAR systems in robotics applications [33], and industrial reverse engineering [34]. Many different techniques exist to perform mesh reconstruction on 3D scans. The motivation for the development of this method was a use case of robot path planning on surfaces of steel beams (as an example see Figure 1), specifically for the task of robotic painting of steel constructions. For painting (as well as other similar scenarios) the following three criteria are desired:

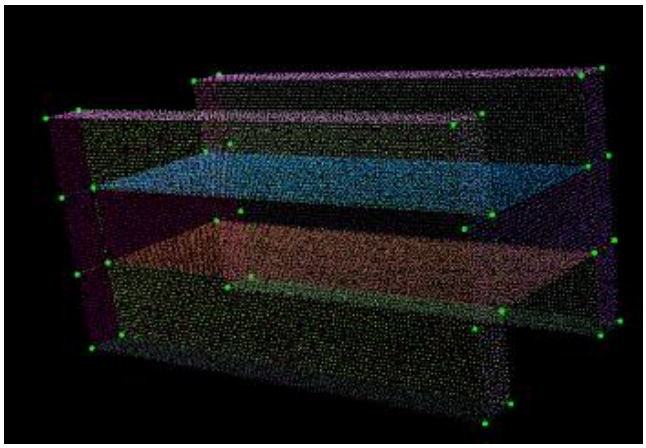

Figure 1. Example of a point cloud scan of an I-beam.

- Minimize number of polygons in the output mesh
- No smoothing of object geometry allowed
- Preserve planar surfaces whenever possible

The above characteristics help in ensuring an even layer of paint can be applied.

These criteria naturally led to the development of a method considering the case of objects constructed from planes, which is presented in this paper. While this constraint does prohibit the method from work on some objects, in particular those with curved surfaces, there are certainly also many real-world objects which do fulfill this criterion, steel beams being an obvious example.

This paper is structured as follows. In section 2, the two main concepts of the method will be explained and some of their previous uses will be listed. Section 3 provides the reader with a short review of related topics, focusing especially on other mesh reconstruction algorithms. In section 4, the proposed new method is explained using a simple test model, and in section 5 the performance of the algorithm on various other objects is shown.

2. BACKGROUND
The proposed algorithm consists of two main parts: The surfaces of the object are found using a RANSAC-based approach, and the connectivity of the mesh is derived from clustering of the point cloud.

2.1 RANSAC

RANSAC, or RANdom SAmple Consensus, first proposed by Fischler and Bolles [3], is a family of methods for stochastic regression fitting of geometric primitives to point data sets, which rejects outliers from the data set by only considering the immediate neighborhood of the shape being fitted [4] (Figure 2).

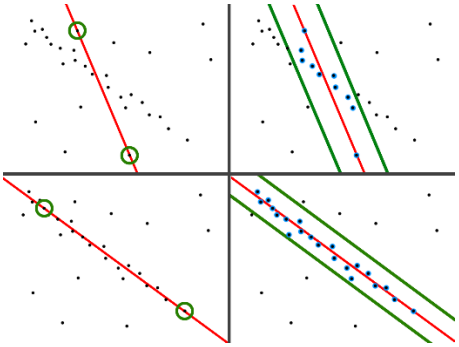

Figure 2. RANSAC fitting of a line.

RANSAC is commonly used in computer vision applications involving cameras or other imaging sensors for its ability to reject large numbers of outliers, finding uses such as visual odometry [3] and shape detection [4]. Because of this robustness, many different alterations and additions to the original method exists, including parallel and distributed versions [5-6], Machine-Learning enhanced fitting [7-8], and adaptive RANSAC methods [9]. Choi et al [10] provide a useful review of various RANSAC methods and their characteristics.

2.2 kNN Radius Filter & ECE

kNN (k-Nearest-Neighbor) is a category of classification methods where data is classified by only looking at the *k* nearest neighbors in the dataset [11]. This method uses a kNN based neighborhood density filter to remove outliers from the data set (Figure 3).

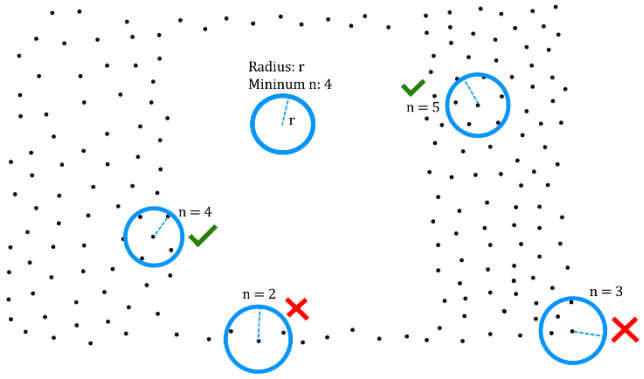

Figure 3. Sample points in a kNN density filter minimum k=4

Generally, classifying outliers based on neighborhood density is known as density clustering [13], and this specific method is sometimes known as the Radial Outlier Removal [12]. Many different schemes exist for density-based clustering [13]. The radius-based kNN filter has the useful property of being able to remove the least dense parts of a surface when used on a set of near-planar points.

ECE (Euclidean Cluster Extraction) uses a similar radius-based approach to classify point data into clusters, but instead of density uses connectivity – a point is in a cluster if it shares at least one neighbor within the search radius r. [14]

When combined, these methods segment point cloud data by density and connectivity: A point is considered a meaningful part of a cluster when it is connected to other points (ECE), but only if it is in a locally dense area of the point cloud (ROR).

3. RELATED WORK

These two steps of regression and segmentation will be exploited to reconstruct the scanned model – however, many other methods for reconstructing meshes from scan data exist.

Two of the broad categories of this point cloud to 3D mesh conversion are "iso-surface extraction" methods, which attempt to recreate the surface of the scanned object based on various local and global characteristics of the input scan data [15]; and "model fitting" methods, which attempt to fit multiple geometric primitives on the input data set until a reasonable approximation of the original object or scene geometry has been constructed [4].

3.1 Surface Extraction

One of the earliest surface extraction methods *Marchine Cubes* (MC), proposed by Lorensen and Cline [16], splits the input point cloud into cells that are classified as one of several cubes. Shu et al [17] proposed an adaptive version (AMC) that allows resizing of mesh faces depending on local shape, reducing output size. Nilson [18] solved a common problem in MC of mesh smoothness and degeneration of the mesh using a dual surface approach, while Pöthkow et al [19] used a Monte-Carlo method for resolving this same problem, avoiding the complicated logic often needed to classify the surface into one of the cube types by using a probabilistic approach.

Another type of method for surface extraction uses 3D alpha shapes, which rely on a generalization of the convex hull to non-convex surfaces (Edelsbrunner and Mücke, [20]). Several methods have been developed that use alpha shapes to reconstruct surfaces [21-23].

Common to both types is that they make no attempt at simplifying the output meshes and prefer a smooth mesh representation of the output – while a desired result may be reached with post-processing of the mesh, they are not ideal for the motivating task of this paper.

Jenke et al [24] proposed a novel Bayesian probabilistic method for surface reconstruction that redefines the reconstruction problem in terms of finding a maximum likelihood surface. Recently, a similar redefinition of the problem has also been attacked with a dictionary learning algorithm by Xiong et al [25], and another machine learning approach using a deep learning method was developed by Williams et al [26]. These statistical optimization methods are well-suited for reconstructing sharp-edged objects without smoothing the geometry, but still require processing to reduce mesh size.

3.2 Similar Methods

Specifically, for tasks where minimizing mesh faces is required such as CAD reconstruction, primitive-fitting methods are more commonly used as they render the final mesh minimal in size.

Arven et al. proposed a method for fitting CAD models to a scene using a convolutional neural network, and presented a large data set for future research into the topic of learned CAD model fitting [29].

Shabayek et al. developed a cross-section based method for CAD reconstruction, slicing a point cloud along its principal axis and reconstructing a model using these cross-sections to preserve connectedness [27].

A similar hybrid approach to the one presented here, using point segmenting and primitive fitting was developed by Friedrich et al [28], using cylinders, boxes and planes as primitives, however no

attempt was made to assert the connectivity of the output mesh, only to correctly deconstruct a model into primitives.

4. METHOD

Like all mesh reconstruction algorithms, the aim of the method is to take as input a point cloud acquired from a data acquisition tool such as a 3D scanner, and output an accurate mesh representing the scanned object. In addition to that, our method reproduces the mesh with as few surfaces as possible.

A diagram of the proposed method can be seen in Figure 4, with the colors of the diagram representing the main steps of the method: Deconstructing the input into planes (orange), finding convex point clusters corresponding to mesh faces in the output model (green), calculating the locations of possible vertices in the output model (purple), and assembling the mesh using the found faces and vertices (blue).

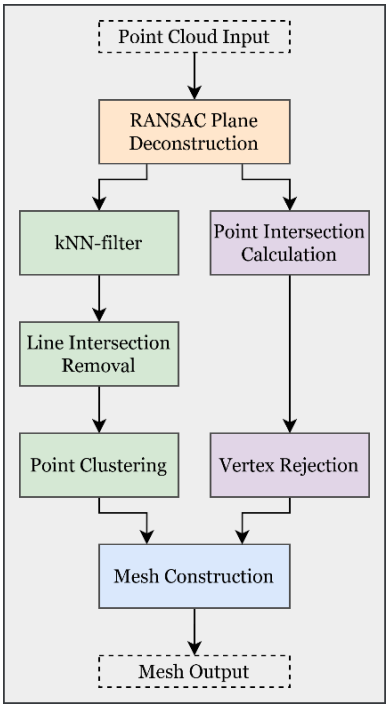

Figure 4. Method overview.

The rest of this section will explain each of these four steps in the above order, concluding with the assumptions that the method makes about the input data, and the implications this has for the usage of the algorithm on generic point cloud data.

4.1 RANSAC Plane Deconstruction

As configuration, this algorithm takes two radii: A RANSAC selection radius r, and feature size R where $r \ll R$. In the first step, the input point cloud is segmented into sets of co-planar points. RANSAC is used to determine the best-fit plane on the point cloud. The coefficients of the normal equation

$$ax + by + cz + d = 0$$

for the best-fit plane is saved to a list, and the RANSAC inlier points are removed from the input point cloud. This is then repeated recursively, ending when the RANSAC plane fitting fails, meaning the input point cloud contains less than 3 points (Figure 5).

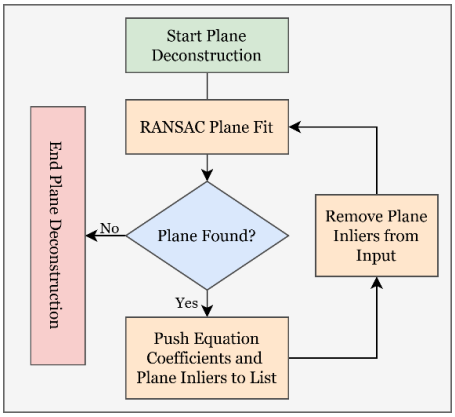

Figure 5. Plane deconstruction flowchart.

In the following steps, the plane equations are used to determine the locations of vertices of the final output mesh, while the plane inliers are used to determine the connectivity of the vertices into mesh faces of the output.

The effect of this deconstruction process on a simple test point cloud can be seen in Figure 6: In each iteration, the best-fit RANSAC plane (green) is found for the input data (red). Its inliers are then added to a list of planar points, and the inliers are removed from the input point cloud. This process continues until a plane can no longer be fitted to the input, and the point cloud has been emptied.

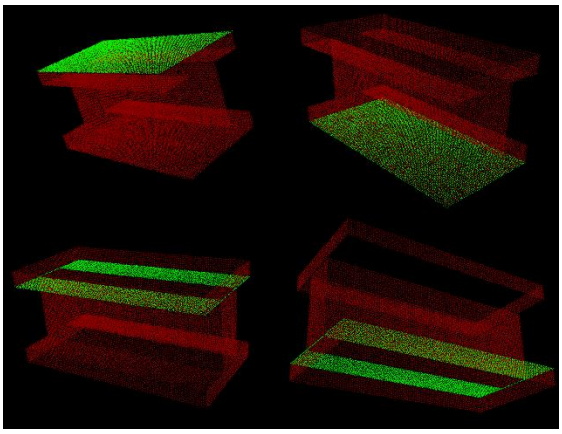

Figure 6. Example of four iterations of plane deconstruction.

4.2 kNN-filtering & Clustering

The list of planar points forms a one-to-one relationship with the faces of the scanned model, if the model is strictly convex. Since this is rarely the case, processing is needed such that the planar points can be converted into a set of "point polygons" corresponding to mesh faces of the output mesh. This is achieved by processing each plane of points in the following way:

- A kNN density filter is used to remove less dense parts of the point cloud. This removes edges of the point clusters and outliers from each plane.
- Points that lie on the interior of a plane but close to an intersection of two planes are removed. This ensures that every cluster is convex in the plane.
- The remaining points are then clustered

- The next section will show that such a clustering of the points leads to a simple way to triangulate and reconstruct the faces of the scanned model.

4.2.1 kNN Filtering

A kNN density histogram for the points of each plane is built, using the feature radius R. This iterates through each point, counting the k nearest neighbors of that point, until the next nearest neighbor is outside the radius R. A typical result of this can be seen in Figure 7. Points on the interior of a surface have many neighbors ($k \approx 190$), points near the edge of a surface have progressively fewer as the edge is approached ($k \approx [90; 180]$), while points not on a surface have few neighbors ($k \approx 20$).

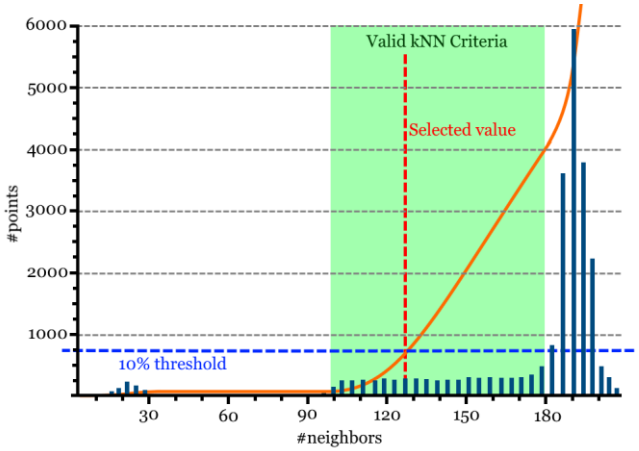

Figure 7. Neighborhood density histogram (blue) and cumulative distribution (orange)

Valid values of k for the kNN filter are values that filter outliers and part of the edges, without removing many points on the interior of the planes. On the histogram, this is the "tail" area of the curve, below the main lobe (shaded green on Figure 7). Selection of the kNN filter criterion is done using cumulative thresholding. When the accumulated neighborhood density exceeds a threshold, the reached value is selected for k, and points with fewer than k neighbors are discarded. An example of the filter being applied to a plane can be seen in Figure 8, removed points marked in orange. The two faces shown are the top of the lower plate of the I in the I-beam. They are mistakenly connected via the ends of the I-beam which necessarily must contain a few points in the same plane as this plate.

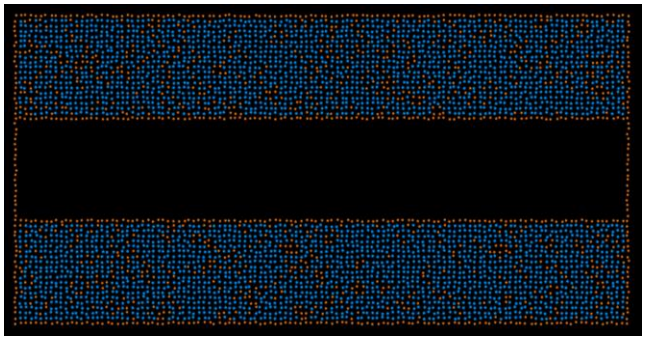

Figure 8. The points removed by the kNN filter.

Note that some points are removed from the interior of the surface. While this is an unintended consequence of the filtering, is does not impact the performance of the algorithm, as there are still many points available on each face. The advantage of this filtering is clear, though: The two narrow lines which connect disjoint faces on the figure due to the 3D nature of the object are removed, leaving us with two separate faces.

4.2.2 Line Intersection Removal

Next, the list of plane equation is traversed, and a line equation for each of the intersections of two planes in the object is obtained. The distance from each point in each plane to this intersection is calculated and points closer than the RANSAC selection radius r are removed.

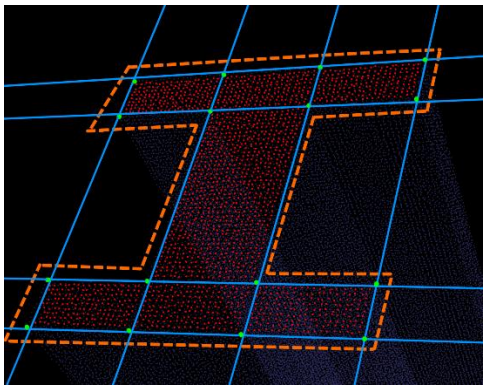

Figure 9. Line intersections shown in blue

For each plane, this creates a number of convex-shaped point sets (Figure 9). Each of these point sets will become a polygon in the final mesh.

4.2.3 Point Clustering

Because of the previous steps, all point sets separated by planes are now separated by a distance of at least $2r$. The final step of the clustering process classifies the point sets into clusters by Euclidean nearest-neighbor distance, using a threshold value slightly below $2r$, which separates them (Figure 10).

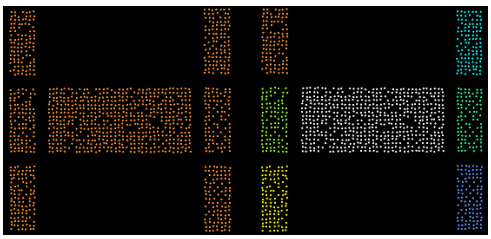

Figure 10. A plane before and after clustering.

After processing every plane, the point cloud is now split apart into several clusters that each correspond to a convex part of the final mesh (Figure 11).

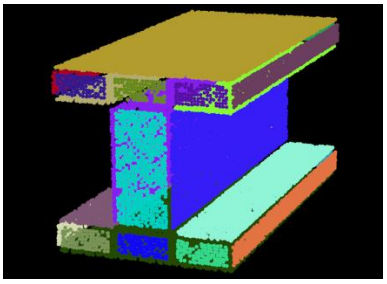

Figure 11. The point cloud after clustering.

4.3 Model Vertex Selection

Next, the mesh vertices are found by intersecting triplets of plane equations

$$ax + by + cz + d = 0$$

using the formula for the intersection point P[30]

$$P_{i,j,k} = \frac{-d_i(n_j \times n_k) - d_j(n_k \times \mathrm{n_i}) - d_k(n_i \times \mathrm{n_j})}{\det(A)}$$

where

$$A = [n_i, n_j, n_k] = \begin{bmatrix} a_i & a_j & a_k \\ b_i & b_j & b_k \\ c_i & c_j & c_k \end{bmatrix}$$

Since every triplet of planes is considered, this produces many unneeded intersection points. Selecting the subset corresponding to the vertices of the original scanned model is first done by bounding box check (near-parallel planes intersect far away from the object) or distance search to the previously found point clusters:

- A list of mesh face vertices is created for that cluster.
- The distance d from each intersection point to the cluster is checked. If $d < R$, a copy of the point is added to the list of face vertices.
- Intersection points that are not part of a list of face vertices are discarded

This leaves only the intersection points that are close enough to a cluster that it could be part of a mesh face (Figure 12).

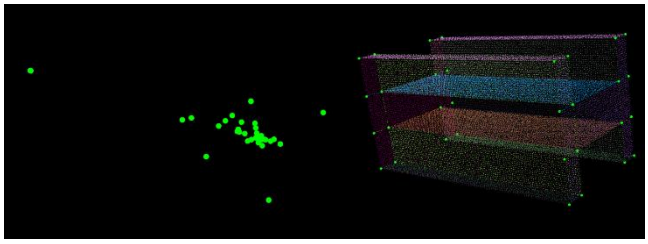

Figure 12. The intersections before (left) and after (right) selection.

4.4 Mesh Construction

Each of the clusters now contains a small number of points, corresponding to a vertex in that cluster's part of the mesh. These mesh vertices are near the corners of the cluster (Figure 13)

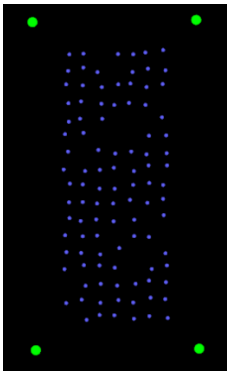

Figure 13. A cluster and its vertices.

Since the cluster is guaranteed convex, the mesh vertices can be joined into a mesh with a radial sweep algorithm; Now the final mesh can be constructed by iterating through each of the unconnected mesh polygons, joining two meshes if they contain coinciding points. In this way, the final model reconstruction is assembled (Figure 14).

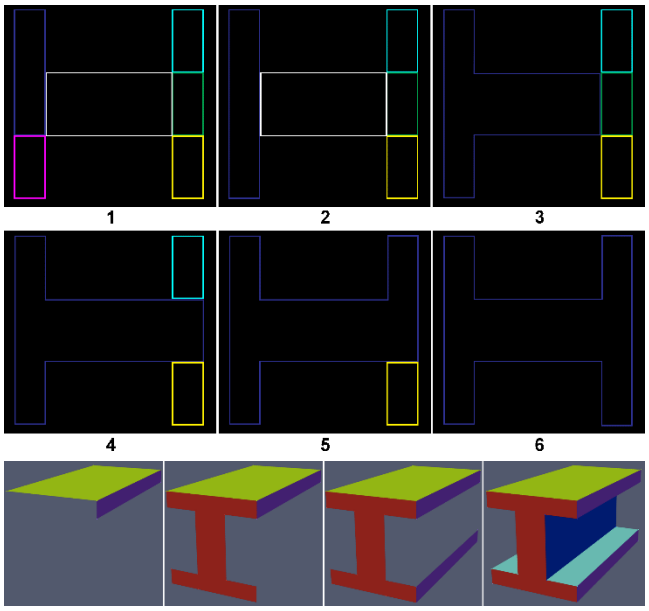

Figure 14. All the meshes of a plane are connected (top) and assembled (bottom).

4.5 Input Assumptions & Method Limitations

The method makes two main assumptions about the data. First, that the input point cloud has constant point density on the sampled model (example in Figure 15). Second, that the model to be reconstructed is well-approximated by a set of planar surfaces (example cross-section in Figure 16).

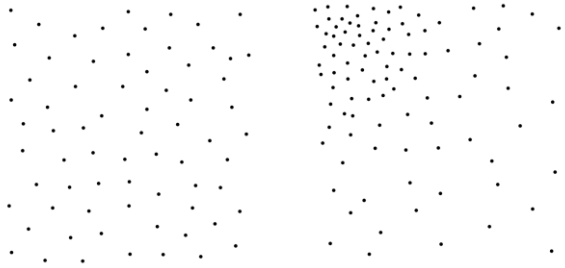

Figure 15. Constant (left) versus non-constant density (right).

The constant-density assumption is used during the clustering process, where a non-constant density will make the kNN method used to filter the point cloud invalid. A point cloud that does not have constant density will need to be resampled before it can be processed.

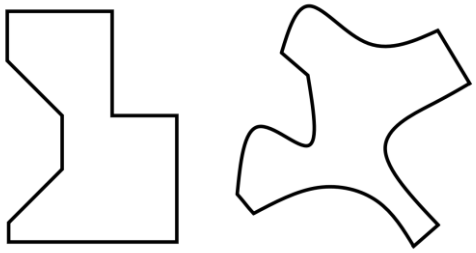

Figure 16. Planar (left) versus non-planar (right) cross-section

The assumption of being well-approximated by planes is introduced by the choice of planes as the RANSAC model that are fit to the point cloud. This could be expanded to include other models, if determining intersections of the various models chosen remain feasible.

5. RESULTS

This section shows the results of applying the method to four different models – three representative of the motivating task, the Wedge Beam, Diamond Profile, and Z-beam, and one input model showing when the method tends to stop working, the Icosahedron (Figure 17).

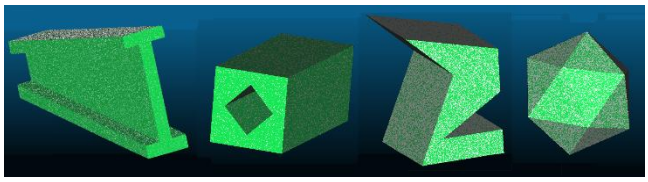

Figure 17. Left to right: Wedge Beam, Diamond Profile, Z-beam, Icosahedron

Points have been sampled evenly on the models with a point density of approximately 4 points per square unit. The algorithm was run with $r = 0.5$ and $R \approx [2.5; 8]$ depending on the model in question. The results can be seen in table 1 and Figure 18.

Table 1: Test results

Model	Wedge	Diamond	Z-beam	Icosa
Points	107,311	119,466	430,869	51,080
R	2.4	4.0	7.9	4.0
Success?	Yes	Yes	Yes	No
Orig. Size	44 faces	32 faces	28 faces	20 faces
New Size	56 faces	40 faces	30 faces	-
Increase	27.3%	25.0%	7.14%	-

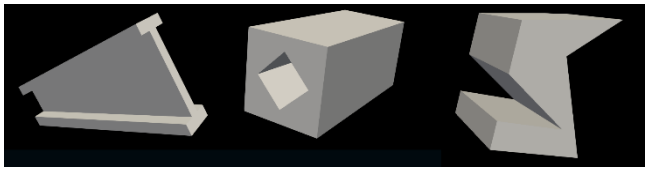

Figure 18. The three successfully reconstructed models.

The method successfully reconstructed three out of the four models, yielding an average increase of only 19.81% in size of the reconstructed model, compared to the original CAD files that the point clouds have been sampled from.This is an acceptable increase compared to the several orders of magnitude difference between the point clouds and the model size.

The final model, the icosahedron, was successfully segmented and clustered by the algorithm, but could not be assembled, instead resulting in the model seen in Figure 19.

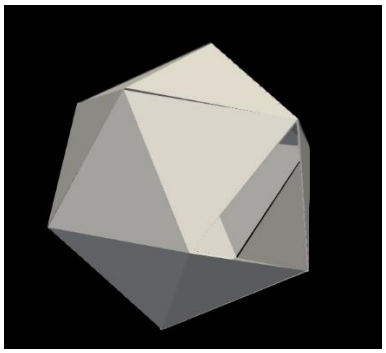

Figure 19. The icosahedron reconstruction. Note the missing face.

This model presents a challenge for the method, for the following reasons: Firstly, the angles that the faces make with each other are shallow, while the internal angles of the triangles are sharp. This makes it difficult to pick a feature size R such that all intersection points get correctly assigned to their cluster, without missing any faces. Secondly, many more than three faces meet in a single point, which means that multiple similar intersection points are found near each vertex of the icosahedron. This makes correctly connecting the faces challenging without a way to reject duplicates.

6. CONCLUSION & FUTURE WORK

A novel RANSAC-based method for reconstructing CAD meshes based on 3D scan data has been presented. The method successfully recreated sharp-edged models from 3D scans of CAD models, with an average increase in the size of the model of only 19.81%. Some challenges for using the method still appear. Choosing functioning parameters for the RANSAC rejection distance, feature size R, and threshold such that the method works can be difficult, and the method has issues recreating objects with wide angles and where many faces share a vertex.

Future work will first focus on testing the method in various other scenarios to confirm its robustness against noise; then on developing good metrics for picking the feature size of an object so the algorithm works. Finally, modifications so that the method can handle pathologic data such as the Icosahedron model presented could be done.

7. ACKNOWLEDGEMENTS

The method implementation was done using the Point Cloud Library (https://pointclouds.org/)[12], which provides fast implementations of RANSAC and the kNN-based radius searches. The sampling of points and images of the test models in section 5 were created using CloudCompare[29], a free open tool for handling and processing point clouds. The project was initially based on a proposal and dataset for surface reconstruction of beams provided by the Danish company Inropa to the authors of this paper – that data is not publicly available, so other test objects have been created.

8. REFERENCES

[1] M. A. Fischler and R. C. Bolles. 1981. “Random Sample Consensus: A paradigm for model fitting with applications to image analysis and automated cartography.” In *Communications of the ACM, 24(6):381–395.*

[2] D. Scaramuzza, F. Fraundorfer and R. Siegwart, "Real-time monocular visual odometry for on-road vehicles with 1-point RANSAC," *2009 IEEE International Conference on*

Robotics and Automation, Kobe, 2009, pp. 4293-4299, doi: 10.1109/ROBOT.2009.5152255.

[3] K. Derpanis, “Overview of the RANSAC Algorithm”, 2010, Image Rochester NY.

[4] Schnabel, R., Wahl, R. and Klein, R. (2007), Efficient RANSAC for Point-Cloud Shape Detection. Computer Graphics Forum, 26: 214-226. doi:10.1111/j.1467-8659.2007.01016.x

[5] Xu, M.; Lu, J.: 'Distributed RANSAC for the robust estimation of three-dimensional reconstruction', IET Computer Vision, 2012, 6, (4), p. 324-333, DOI: 10.1049/iet-cvi.2010.0223

[6] M. Alehdaghi, M. A. Esfahani and A. Harati, "Parallel RANSAC: Speeding up plane extraction in RGBD image sequences using GPU," *2015 5th International Conference on Computer and Knowledge Engineering (ICCKE)*, Mashhad, 2015, pp. 295-300, doi: 10.1109/ICCKE.2015.7365845.

[7] Eric Brachmann, Carsten Rother; Neural-Guided RANSAC: Learning Where to Sample Model Hypotheses. *Proceedings of the IEEE/CVF International Conference on Computer Vision* (ICCV), 2019, pp. 4322-4331

[8] Volker Rodehorst and Olaf Hellwich. Genetic Algorithm SAmpleConsensus (GASAC) - a parallel strategy for robust parameter esti-mation. InProceedings of the IEEE Computer Society Conferenceon Computer Vision and Pattern Recognition Workshop (CVPRW),

[9] Raguram R., Frahm JM., Pollefeys M. (2008) A Comparative Analysis of RANSAC Techniques Leading to Adaptive Real-Time Random Sample Consensus. In: Forsyth D., Torr P., Zisserman A. (eds) Computer Vision – ECCV 2008. ECCV 2008. Lecture Notes in Computer Science, vol 5303. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-88688-4_37

[10] Choi, Sunglok & Kim, Taemin & Yu, Wonpil. (2009). Performance evaluation of RANSAC family. Proceedings of the British Machine Vision Conference 2009. 24. 10.5244/C.23.81.

[11] Altman, Naomi S. (1992). "An introduction to kernel and nearest-neighbor nonparametric regression” The American Statistician. 46 (3): 175–185. doi:10.1080/00031305.1992.10475879. hdl:1813/31637.

[12] Radu Bogdan Rusu and Steve Cousins, 2011 “3D is here: Point Cloud Library (PCL)”, IEEE International Conference on Robotics and Automation (ICRA),

[13] Kriegel, H.-P., Kröger, P., Sander, J. and Zimek, A. (2011), Density-based clustering. WIREs Data Mining Knowl Discov, 1: 231-240. doi:10.1002/widm.30

[14] Radu Bogdan Rusu (2009), *Semantic 3D Object Maps for Everyday Manipulation in Human Living Environments* Technische Universitaet Muenchen

[15] Charles D. Hansen; Chris R. Johnson (2004). Visualization Handbook. Academic Press. pp. 7–11. ISBN 978-0-12-387582-2.

[16] William E. Lorensen and Harvey E. Cline. 1987. Marching cubes: A high resolution 3D surface construction algorithm. *SIGGRAPH Comput. Graph.* 21, 4 (July 1987), 163–169. DOI:https://doi.org/10.1145/37402.37422

[17] Shu, R., Zhou, C. & Kankanhalli, M.S. Adaptive marching cubes. *The Visual Computer* **11,** 202–217 (1995). https://doi.org/10.1007/BF01901516

[18] G. M. Nielson, "Dual marching cubes," *IEEE Visualization 2004*, Austin, TX, USA, 2004, pp. 489-496, doi: 10.1109/VISUAL.2004.28.

[19] Pöthkow, K., Weber, B. and Hege, H.-C. (2011), Probabilistic Marching Cubes. Computer Graphics Forum, 30: 931-940. doi:10.1111/j.1467-8659.2011.01942.x

[20] Herbert Edelsbrunner and Ernst P. Mücke. 1994. Three-dimensional alpha shapes. *ACM Trans. Graph.* 13, 1 (Jan. 1994), 43–72. DOI:https://doi.org/10.1145/174462.156635

[21] Guo, B., Menon, J. and Willette, B. (1997), Surface Reconstruction Using Alpha Shapes. Computer Graphics Forum, 16: 177-190. doi:10.1111/1467-8659.00178

[22] Chandrajit L. Bajaj, Fausto Bernardini, and Guoliang Xu. 1995. Automatic reconstruction of surfaces and scalar fields from 3D scans. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (*SIGGRAPH '95*). Association for Computing Machinery, New York, NY, USA, 109–118. DOI:https://doi.org/10.1145/218380.218424

[23] Maillot Y., Adam B., Melkemi M. (2010) Shape Reconstruction from Unorganized Set of Points. In: Campilho A., Kamel M. (eds) Image Analysis and Recognition. ICIAR 2010. Lecture Notes in Computer Science, vol 6111. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-13772-3_28

[24] Jenke, P., Wand, M., Bokeloh, M., Schilling, A. and Straßer, W. (2006), Bayesian Point Cloud Reconstruction. Computer Graphics Forum, 25: 379-388. doi:10.1111/j.1467-8659.2006.00957.x

[25] Shiyao Xiong, Juyong Zhang, Jianmin Zheng, Jianfei Cai, and Ligang Liu. 2014. Robust surface reconstruction via dictionary learning. *ACM Trans. Graph.* 33, 6, Article 201 (November 2014), 12 pages. DOI:https://doi.org/10.1145/2661229.2661263

[26] Francis Williams, Teseo Schneider, Claudio Silva, Denis Zorin, Joan Bruna, Daniele Panozzo; Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 10130-10139

[27] Shabayek, A.; Aouada, D.; Cherenkova, K. and Gusev, G. (2020). Towards Automatic CAD Modeling from 3D Scan Sketch based Representation.In *Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP*

[28] Friedrich, M.; Illium, S.; Fayolle, P. and Linnhoff-Popien, C. (2020 Friedrich, M.; Illium, S.; Fayolle, P. and Linnhoff-Popien, C. (2020). A Hybrid Approach for Segmenting and Fitting Solid Primitives to 3D Point Clouds.In Proceedings of the 15th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications - Volume 1: GRAPP

[29] http://cloudcompare.org/

[30] Weisstein, Eric W. "Plane-Plane Intersection." From *MathWorld*--A Wolfram Web Resource. https://mathworld.wolfram.com/Plane-PlaneIntersection.html

[31] Szymon Rusinkiewicz, Olaf Hall-Holt, and Marc Levoy. Real-time 3d modelacquisition. Association for Computing Machinery, 2002. https://dl.acm.org/doi/abs/10.1145/566654.566600.

[32] Z.L.Wanga J.C.M.Teoa C.K.Chuia S.H.Ongbc C.H.Yanb S.C.Wangd H.K.Wonge S.H.Teoh, Computational biomechanical modelling of the lumbar spine using marching-cubes surface smoothened finite element voxel meshing (2005), *Computer Methods and Programs in Biomedicine*

[33] Akira Kato L. Monika Moskal Peter Schiess Mark E.Swanson Donna Calhoun Werner Stuetzle, Capturing tree crown formation through implicit surface reconstruction using airborne lidar data (2009), *Remote Sensing of Environment*

[34] Reverse Engineering, Josef Hoschek Werner Dankwort (1996), ISBN 978-3-322-84819-2

Columns on Last Page Should Be Made As Close As Possible to Equal Length

Authors' background

Your Name	Title*	Research Field	Personal website
Thor A. Sørensen	Master's student	Computer Vision	
Natalie Mark	Master's student	Robotics	
Andreas Møgelmose	Assistant professor	Computer Vision	http://moegelmose.com

***This form helps us to understand your paper better, the form itself will not be published.**

***Title can be chosen from: master student, Phd candidate, assistant professor, lecture, senior lecture, associate professor, full professor**