2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Learning Continuous Object Representations from Point Cloud Data

Henry J. Nelson and Nikolaos Papanikolopoulos
{nels8279, papan001}@umn.edu
Department of Computer Science and Engineering, University of Minnesota

***Abstract*— Continuous representations of objects have always been used in robotics in the form of geometric primitives and surface models. Recently, learning techniques have emerged which allow more complex continuous representations to be learned from data, but these learning techniques require training data in the form of watertight meshes which restricts their application as meshes of this form are difficult to obtain from real data. This paper proposes a modification to existing methods that allows real world point cloud data to be used for training these surface representations allowing the techniques to be used in broader applications. The modification is evaluated on ModelNet10 to quantify the difference between the existing and the proposed methods as well as on a novel precision agriculture dataset that has been released publicly to show the modification's applicability to new areas. The proposed method enables obtaining training data from real world sensors that produce point clouds rather than requiring an expensive meshing step which may not be possible for some applications. This opens the possibility of using techniques like this for complex shapes in areas like grasping and agricultural data collection.**

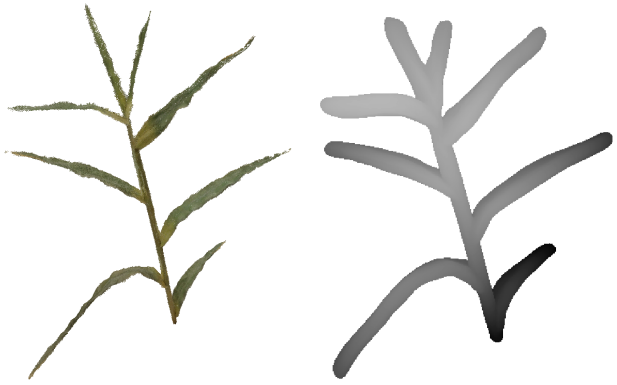

Fig. 1: Left: Point cloud of a corn stalk. **Right:** Depth map of learned continuous model of the same corn stalk.

I. Introduction

In order for robots to participate in our three dimensional world, they must have a reliable three dimensional representation of objects. These representations are commonly used in robotics for grasp planning [1]–[3] as well as for measuring surface properties like the surface area of plant leaves [4], [5]. While representing objects with simple geometric primitives [1], [2] works for simple manufactured objects, more complex or natural objects such as plants cannot be represented in the same way. Instead, complex surface models such as an elastic surface [3] or self-organizing map [4], [6] are commonly used but these representations cannot robustly deal with partial data.

In recent years however, deep learning has proven effective at producing representations of complex objects that are able to generalize well to unseen and even partial data at inference time [7]–[10]. But accuracy and generalization are not the only factors influencing the choice of representation in robotic perception systems, the chosen representation must also be compatible with the robot's sensors and useful for completing the robot's tasks. Currently, most sensors for collecting three dimensional data produce point clouds so any useful continuous object representation should be compatible with point cloud data. The downside of point clouds though, is that they are a sparse sampling of continuous objects making activities like navigation and grasp planning more difficult. In these situations a continuous representation, typically geometric primitives [1], [2] or an elastic surface model [3], are fit to the point cloud and used in place of the point cloud to complete the task. But fitting a model to a point cloud is difficult when the objects have a more complex shape or there is only partial data which has led to the development of more complex continuous models, the most recent of which have been deep learning based [9], [10]. While these methods can be trained to represent a wide variety of objects, they are defined to train from continuous data in the form of watertight polygonal meshes which restricts them from being trained on point cloud data. While creating meshes from point clouds is a well researched area [11]–[13], it is still difficult to produce watertight meshes for complex structures, especially those with extremely thin components such as the leaves of a plant. This leads to the question of whether we really need to produce a watertight mesh of the data, or whether we can train these representations from point clouds directly.

In this paper, a method for approximating the signed distance function of an object's surface from its point cloud is given. This is done by computing the distance function of the subset of the points that lay on the object's surface to approximate the distance function of the object's surface and setting the sign based on whether the point is internal or external to the object. This works in a straightforward way since the point cloud is a sampling of the object's surface meaning that its distance function will approach the distance function of the surface as the sampling density increases. Other challenges like the problem of representing thin components of an object and the problem of determining whether a point is inside or outside the surface of an object based solely on a point cloud are similarly addressed by using a simple offset and an approximate visibility algorithm. A point cloud dataset for precision agriculture that exemplifies some of the

difficulties of geometric model fitting and meshing methods is provided to show that continuous representations can be obtained from real world point cloud data with the proposed method and the use of one of the aforementioned deep learning based object representation methods. The use of the method and precision agriculture data is an important step in agriculture automation and plant phenotype estimation as plant models are extremely variable and hard to fit traditional models to. Yet obtaining a continuous surface model for the plant is extremely important for estimating a plant's leaf area index [4], [5].

In the end the proposed method takes in a point cloud and produces a signed distance function from which one of the continuous object representation networks can be trained. This allows the user to produce generalizable and continuous representations of complex objects from point clouds, as can be seen in Fig. 1. The error of the signed distance function produced from a point cloud via the methods given in this paper is only about 1% of the object size when measured on a large dataset of meshes [14] and the representations learned from the data are both accurate and generalizable. The major contribution of this work is that it bridges the gap between the widely available data format of point clouds and the emerging field of learning-based continuous representation techniques. It is also a step towards developing general surface models that are complex, yet generalizable enough to be used in agricultural automation.

II. Related Work

In the image domain, using deep representation learning techniques such as GANs [15] and VAEs [16] is now commonplace. Recently, there has been a lot of work extending the successes of these techniques from the image domain to three dimensional data. This includes work on voxel grids, point clouds, and more recently continuous representations. In this section, various three dimensional object representation approaches are discussed. The main focus is kept on works dealing with point cloud data and those learning continuous object representations.

While works like PointNet [17] and PointNet++ [18] have paved the way for using point clouds directly in deep learning formulations, they are restricted by the discrete representations which they were built to process. As has been shown in [7], [8], these architectures can be used to learn object representations from point clouds but they also produce point cloud models making them inadequate for circumstances such as navigation or grasp planning where a continuous model would be preferred. This is because they may be able to denoise and increase the number of points representing the object, but in the end a continuous model would still have to be fit to the points in order to complete the task.

On the other end of the spectrum, works like DeepSDF [9] and SRNs [10] produce a continuous 3D representation. The continuous representation is produced by encoding the signed distance function for the surface of the object, a function where every point in space is assigned the value of the distance from that point in space to the surface of the object and the sign corresponds to whether the point is inside (<0) or outside (>0) the object surface as the surface must be a completely closed boundary. This inherently continuous representation is just a simple extension of the level-sets concept which has been extensively used in the image domain for active contours [19] and even in the three dimensional image domain for medical image segmentation [20]. The object boundary is thus defined to be the zero level-set of the signed distance function. This representation will only work for objects with a completely closed boundary, like those of a watertight mesh. This restricts the method from being used in instances where both sides of a thin face are external, like the object shown in Fig. 2, because the network approximated signed distance function will not have a zero crossing at this boundary as it is positive on both sides of the boundary. In order to avoid this problem, [9] sidesteps the issue and throws out any shapes that fit this criterion as they do not have a completely closed boundary.

Although it has some restrictions on shape, this work heavily utilizes DeepSDF [9] as the representation learning framework. In fact, the work of this paper can be viewed as a simple modification of the DeepSDF methodology as after the visibility of faces and normal information is established, each face is densely sampled into a set of points. The work of this paper thus removes the requirement of mesh training data and the restriction to closed surface (watertight) object shapes from the DeepSDF methodology.

III. Methods

The work of this paper centers around estimating the signed distance function from a given point cloud. To this end some notion of whether a given point is inside or outside the surface of the object must be produced. Since this internal versus external concept is ill-defined on a point cloud, we borrow techniques from the graphics community for estimating the surface and thus the boundaries of objects. Specifically we heavily utilize the techniques proposed in [21] which defines an algorithm for testing the visibility of points in a point cloud and estimating the surface they are sampled from. The algorithm uses the convex hull on a set of points after a spherical reflection centered at the viewpoint. This produces a set of visible points and triangular faces which can be used to estimate the surfaces of the object. A simple solution called ε-correction is also proposed to remove the non-watertight surface restriction that was discussed earlier. Altogether, the final approximation for a signed distance function of a given surface is:

$$f(\vec{x}) = vis(\vec{x}) \cdot dist(\vec{x}) - \varepsilon. \tag{1}$$

In the following sections each of the parts of this function are explained in detail.

The input point cloud, $P = \{\vec{x_1}, \vec{x_2}, \dots\}$, to this methodology should be segmented from its surroundings so that only the points of the object remain. Then any noise points in the point cloud should be removed. In practice even simple noise filters like the statistical outlier removal algorithm from

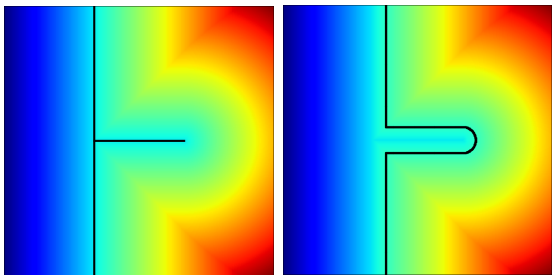

Fig. 2: A surface (black line) and its distance field (color) where the left edge of each image is inside the object and the right edge is outside. **Left:** Original surface with infinitely thin component. **Right:** Surface after ε-correction.

PCL [22] were sufficient. The major constituent functions of equation (1), $vis(\vec{x})$ and $dist(\vec{x})$, both require the subset $P_s \subseteq P$ that lies on the surface of the object. While this may be the entire point cloud, in practice there is usually some range that the points spread through the surface due to noise, meaning that we must remove them in order to get a definite surface. To find P_s the point cloud is translated to the origin and normalized to fit within the unit sphere. Then, k viewpoints are uniformly selected from the surface of a sphere of radius 2 centered at the origin. For each of the k viewpoints we compute the triangles of the visible faces from this viewpoint with the technique proposed in [21] and select the points of P that are a vertex of, or lie on, one of these triangles. The final set of surface points P_s is thus the union of the k subsets of P produced from each viewpoint. During this step we save the spherical reflection radius and convex hull computed during the visibility determination for use in a later step. The number k is chosen so that each exterior point on the object should be visible by at least one viewpoint. This will need to increase for shapes with more concavities. In practice $k = 50$ was found to be adequate for all tests.

Now that we have the subset of points corresponding to the surface of the object P_s, the constituent functions of equation (1) can be defined. Computing the function $vis(\vec{x})$ is very much like the the process of finding the surface points of the point cloud. For a given point $\vec{x}$, the spherical reflection is computed for each viewpoint and compared with the convex hull found for that viewpoint. If the reflected point is found to lie within a convex hull, then it is visible. If the point lies outside of all k convex hulls then the point is not visible. Thus we can define $vis(\vec{x})$ as:

$$vis(\vec{x}) = \begin{cases} 1 & \text{if visible} \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

The function $dist(\vec{x})$ is similarly simple as it is just the distance function of P_s. The definition of $dist(\vec{x})$ is thus:

$$dist(\vec{x}) = \min_{\vec{p} \in P_s} \{\|\vec{x} - \vec{p}\|\}. \tag{3}$$

So when taking the product $vis(\vec{x}) \cdot dist(\vec{x})$ we get a signed distance function where $vis(\vec{x})$ supplies the sign information and $dist(\vec{x})$ supplies the distance information.

After taking this product, equation (1) is complete except for the subtraction of the ε constant. Subtracting this constant where $\varepsilon > 0$, called the ε-correction, is the small fix to turn non-watertight models into watertight models. Subtracting a constant effectively enlarges the model by a small amount making thin components have at least a thickness of 2ε. As can be seen in Fig. 2, this makes a closed model out of even infinitely thin components ensuring that there will be a zero crossing in the signed distance function. This is important because infinitely thin objects have no volume and thus will not produce a surface. This subtraction enlarges the entire model slightly, but this is only significant in extremely thin components as long as ε is small. Since the models in this work were normalized to fit within the unit sphere, $\varepsilon = 0.02$ (or about 1% the object length) was found to be effective.

IV. Data

In order to test the methods proposed in this paper on real point cloud data, a novel dataset that exemplifies many of the challenges of representing point clouds as a continuous model was produced. Corn plants were selected as the subject of this dataset as measuring the surface area of leaves is informative and could benefit from a continuous representation [4]. Also, the shape of plants is highly variable and the leaves of the plant are extremely thin structures making it an excellent example of a point cloud for which a watertight mesh is prohibitively difficult to obtain. Each plant in the dataset is a point cloud produced via structure from motion using COLMAP [23], [24].

A set of 5 to 10 plants were imaged as a group, collecting roughly 100 images. Since each plant may or may not be visible in any one image it is difficult to say how many images were used to produce each model. The plants which were imaged were highly realistic synthetic corn plants which were used so that the position of leaves could be modified between each reconstruction so that multiple models of different shapes could be obtained from the same plant. This process was repeated until there were 50 different models in total.

The point cloud models were segmented from the environment based on color after which the statistical noise filter from [22] was applied. Then each plant was manually separated from the others and scaled to fit within the unit sphere. They were then manually oriented so that the up direction of the stem pointed in the z direction and the stem passed through the origin. Each model was then translated so that the centroid of the point cloud is in the xy-plane.

It is notable that since the position and orientation of leaves is so variable for plants, there is no way to completely register the models. Since DeepSDF [9], the learning framework used in this paper, requires registered models, the stems of the plants were registered and the rest of the plant is ignored for registration. The plants thus fall in a variety of rotations around the z-axis. In order to learn though this variation, each model was augmented into 32 different models rotated around the z-axis with $\frac{\pi}{16}$ radians between each consecutive rotation. The dataset is made publicly available online[1].

[1] https://github.com/hennels/Corn50

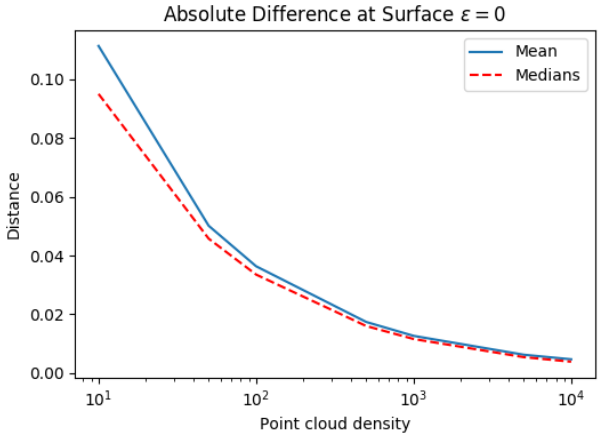

Fig. 3: The absolute difference (or Chamfer distance) between the original function used in DeepSDF [9] and the replacement function proposed in this paper. Each model was centered at the origin and scaled to the unit sphere. This is an important qualification as it gives some sense of the scale for the sampling densities and distances.

V. Results

To test the methods proposed in this paper we use both quantitative and qualitative tests. First, since equation (1) is being proposed as a drop in replacement for the signed distance function proposed in the DeepSDF paper [9], the error between the two functions is quantified. But in many ways, since the shape is only determined by the zero points of the function, the error between the functions at non-zero points does not matter. Thus, only the difference at zeros of the functions matter. Since each function is a distance function it is notable that the difference between the two functions at zero points is effectively the Chamfer distance between the two shapes represented by the signed distance fields, which is a common measure for differences between shapes.

Since we need both a mesh and point cloud to evaluate the difference between the two functions, we evaluate the functions on meshes which we then sample at a variety of densities to produce point clouds. For this we use the ModelNet10 dataset [14] as it provides numerous meshes of complex shaped objects that were constructed manually. For tests, 100 watertight models from the training sets of each of the 10 categories were randomly selected to ensure even weighting between object categories. To produce point clouds of different sample densities, the meshes were uniformly sampled at a variety of densities. Since one of the steps in the DeepSDF function requires a dense sampling of the mesh, we sample the mesh at 10000 points per unit area and run tests between this function and the functions produced by the point clouds which are sampled at the same and much lower densities. Since the ε-correction will necessarily make the surfaces occur at different positions because it slightly inflates the model, this test was performed with both $\varepsilon = 0$ and $\varepsilon = 0.02$ and the results are shown in Figs. 3 and Fig. 4 respectively.

From these tests it is clear that the difference between the DeepSDF function that operates on meshes and the one proposed in this paper that operates on point clouds is small. Since the objects are centered on the origin and scaled to the unit sphere, the difference between the functions can roughly be stated as a fraction of the object size. This means that one unit in distance is about half the longest dimension of object (assuming symmetry of the object). Thus, most of the average deviations in Fig. 3 are about 1% to 2.5% of the size of the object. This is very small considering that the function uses point clouds which are orders of magnitude more sparse than the sampling of the mesh used in the original function. Also in Fig. 4, which shows the absolute difference between the functions when using an ε value of 0.02, it can be seen that the there is a sampling density at which there is a minimum error. This contrasts with the case when ε was 0 where the error seemed to drop monotonically as the point cloud density increased. In fact, the ε-correction mitigates some of the error between the functions for all but the highest sampling densities. This is because in sparse point clouds the distance function is always an overestimate of the true distance function. So the ε-correction reduces this error. While the ε-correction was proposed as a work around for non-watertight models or models with extremely thin components, it seems it is beneficial in more than just these instances if the point cloud is sparse. But if the point cloud is very dense or there are no infinitely thin components then there is no reason to use the ε-correction as it will always change the learned shapes. Furthermore, it is obvious in a ray-cast depth rendering, like Fig. 1, that the surface properties of of the resulting models are altered by the ε parameter due to the inflation effect, so this relation should be quantified if surface properties are important for the application.

While quantifying the difference between the two functions tells us that they are similar, the real test of this work is whether or not the new function can be used to train an object representation network. To test this, models produced by a network trained using the function proposed in this work are inspected qualitatively. In order to be adequate for use, a network trained with the function proposed in

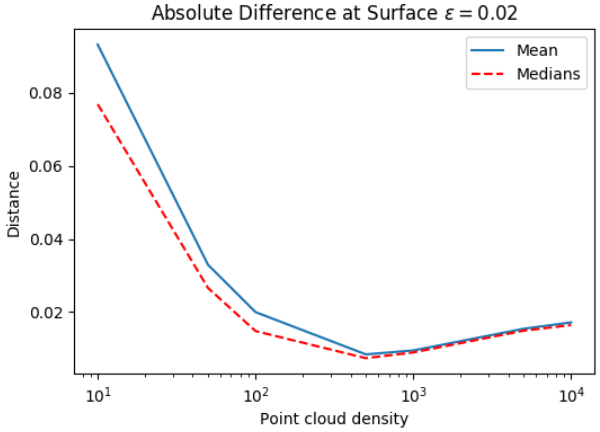

Fig. 4: The absolute difference (or Chamfer distance) between the original function used in DeepSDF [9] and the replacement function proposed in this paper. Each model was centered at the origin and scaled to the unit sphere. This is an important qualification as it gives some sense of the scale for the sampling densities and distances.

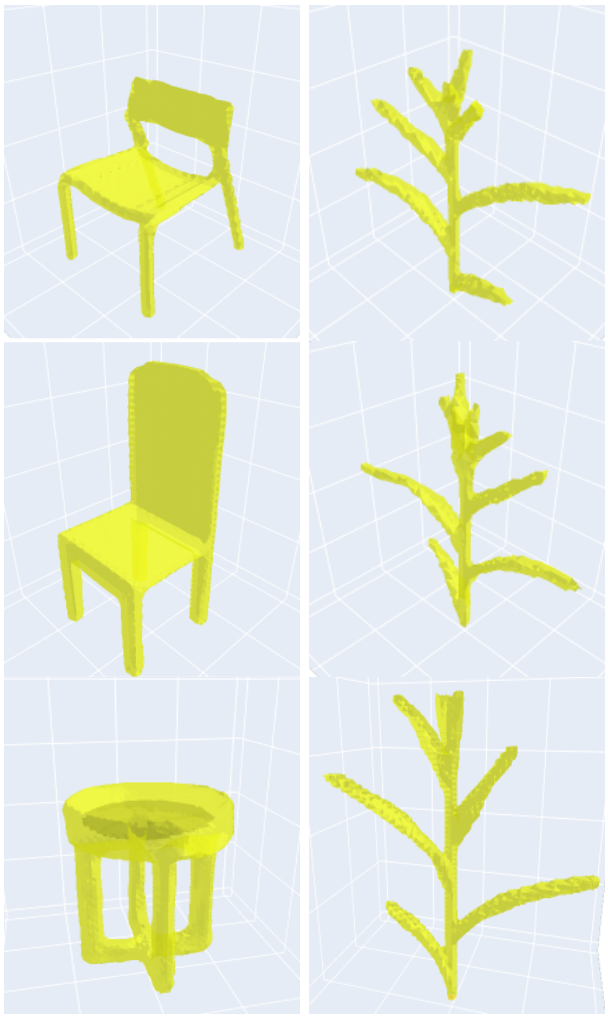

Fig. 5: The output of the DeepSDF network on **training** data after being trained on point cloud data using the method proposed in this paper. The continuous representation is discretized using marching cubes [25] to produce the visualization.

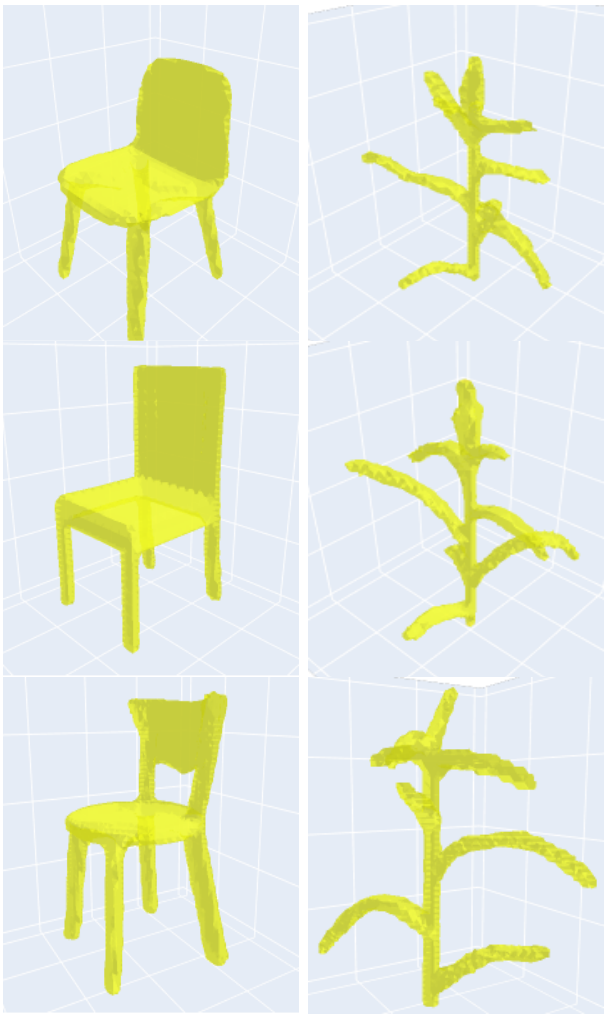

Fig. 6: The output of the DeepSDF network on **unseen** data after being trained on point cloud data using the method proposed in this paper. The fit was produced by running gradient descent in the latent shape space to find the shape minimizing the loss of the network. For the fit, the output of the network at each point was set to $-\varepsilon$ as that is the result of equation (1) assuming the points are on the surface of the object. The continuous representation is discretized using marching cubes [25] to produce the visualization.

this paper should describe both the data it was trained on and generalize to unseen data. To test this the DeepSDF [9] architecture was trained twice, once on the plant point clouds described in Section IV and once on point clouds sampled from the ModelNet10 [14] meshes in the chairs category. Since the ε-correction is used there is no need to throw out models if they are non-watertight as was required in the original DeepSDF methodology. Except for using point clouds and the function from this work, the training process exactly followed the process and hyperparameters given in [9]. In Figs. 5 and 6 it is shown that both of these tasks are completed well on both datasets. The shapes produced by the network are quite obviously instances of their respective objects, chairs and corn plants. While it is not obvious in the ModelNet10 objects, the inflation of the objects due to the ε-correction is obvious in the plants. This is especially true in the leaves which now obviously have some thickness. This is especially visible in Fig. 1 where a point cloud and its continuous representation are shown side by side. But besides the slightly bloated appearance, the object representations obtained for both datasets describe the objects well. This test proves the applicability of the function proposed in this work because accurate continuous representations for both known and unknown data can be obtained from only point cloud data rather than requiring mesh data.

VI. Conclusion

In this work a new function to estimate the signed distance function of an object given only the object's point cloud was proposed and validated. This work shows that there is no need for the requirement of mesh data in continuous object representation learning frameworks. The relaxation of this restriction allows current methods to be used on non-watertight shapes after the proposed ε-correction which was not possible previously. It also opens the door to using real world data for object representation learning rather than requiring mesh data which was difficult to obtain and often required manual edits to automatically generated meshes. The work thus allows large amounts of data to be used as it

removes many of the barriers encountered when building a large dataset of meshes. This work bridges the gaps between the machine learning and robotics communities allowing the robotics community to benefit from and apply the recent advances in continuous shape representations by enabling the use of point clouds, which are a common data format used in robotics. This work also opens up more areas of robotics application such as agricultural automation and phenotype extraction because it allows complex object surfaces such as plants to be modeled effectively without manual intervention, which was difficult or impossible with earlier methods. In the end this work is simply a modification to an existing framework, DeepSDF [9], that allows it to operate on a more commonly used data format. But while this paper has solved some of the problems in applying these methods to robotics, there are still others that are yet unsolved.

A. Limitations and Future Work

Like the original DeepSDF formulation, all models exist in a standard coordinate system so it is hard to complete partial models unless they are in the standard coordinate system. Also, like any currently existing learned object representation method, it must be trained from complete models. This means that partial models where all sides of the object are not observed cannot be used as training data. Another restriction this method inherits from DeepSDF is its inability to deal with surfaces with large or deep concavities because sign of the distance function is determined by visibility. All of these problems make utilizing these algorithms in robotic systems difficult because translation and rotation of objects is often unknown and data is often not observed from all sides. Because of this, these are excellent candidates for future works.

VII. Acknowledgements

The authors would like to thank all the members of the Center for Distributed Robotics Laboratory for their help. This material is based upon work partially supported by the Corn Growers Association of MN, the Minnesota Robotics Institute (MnRI), Honeywell, and the National Science Foundation through grants #CNS-1439728, #CNS-1531330, and #CNS-1939033. USDA/NIFA has also supported this work through the grant 2020-67021-30755.

References

[1] A. T. Miller, S. Knoop, H. I. Christensen, and P. K. Allen, "Automatic grasp planning using shape primitives," in *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)*, vol. 2, Sep. 2003, pp. 1824–1829 vol.2.

[2] N. Yamanobe and K. Nagata, "Grasp planning for everyday objects based on primitive shape representation for parallel jaw grippers," in *2010 IEEE International Conference on Robotics and Biomimetics*, Dec 2010, pp. 1565–1570.

[3] V. Lippiello, F. Ruggiero, B. Siciliano, and L. Villani, "Visual grasp planning for unknown objects using a multifingered robotic hand," *IEEE/ASME Transactions on Mechatronics*, vol. 18, no. 3, pp. 1050–1059, June 2013.

[4] D. Zermas, V. Morellas, D. Mulla, and N. Papanikolopoulos, "Estimating the leaf area index of crops through the evaluation of 3d models," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sep. 2017, pp. 6155–6162.

[5] F. Hosoi and K. Omasa, "Voxel-based 3-d modeling of individual trees for estimating leaf area density using high-resolution portable scanning lidar," *IEEE Transactions on Geoscience and Remote Sensing*, vol. 44, no. 12, pp. 3610–3618, Dec 2006.

[6] T. Kohonen, "The self-organizing map," *Proceedings of the IEEE*, vol. 78, no. 9, pp. 1464–1480, Sep. 1990.

[7] P. Achlioptas, O. Diamanti, I. Mitliagkas, and L. Guibas, "Learning representations and generative models for 3d point clouds," *arXiv preprint arXiv:1707.02392*, 2017.

[8] M. Zamorski, M. Zięba, P. Klukowski, R. Nowak, K. Kurach, W. Stokowiec, and T. Trzciński, "Adversarial autoencoders for compact representations of 3d point clouds," *Computer Vision and Image Understanding*, p. 102921, 2020.

[9] J. J. Park, P. Florence, J. Straub, R. Newcombe, and S. Lovegrove, "Deepsdf: Learning continuous signed distance functions for shape representation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2019, pp. 165–174.

[10] V. Sitzmann, M. Zollhöfer, and G. Wetzstein, "Scene representation networks: Continuous 3d-structure-aware neural scene representations," in *Advances in Neural Information Processing Systems*, 2019, pp. 1119–1130.

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in *Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, 1992, pp. 71–78.

[12] B. Curless and M. Levoy, "A volumetric method for building complex models from range images," in *Proceedings of the 23rd annual conference on Computer graphics and interactive techniques*, 1996, pp. 303–312.

[13] J. C. Carr, R. K. Beatson, J. B. Cherrie, T. J. Mitchell, W. R. Fright, B. C. McCallum, and T. R. Evans, "Reconstruction and representation of 3d objects with radial basis functions," in *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, 2001, pp. 67–76.

[14] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3d shapenets: A deep representation for volumetric shapes," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2015, pp. 1912–1920.

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, 2014, pp. 2672–2680.

[16] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[17] C. R. Qi, H. Su, K. Mo, and L. J. Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 652–660.

[18] C. R. Qi, L. Yi, H. Su, and L. J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," in *Advances in neural information processing systems*, 2017, pp. 5099–5108.

[19] T. F. Chan and L. A. Vese, "Active contours without edges," *IEEE Transactions on image processing*, vol. 10, no. 2, pp. 266–277, 2001.

[20] Y. L. Guisheng Yin and Y. Wang, "3d level set model for medical image segmentation," in *2009 International Conference on Future BioMedical Information Engineering (FBIE)*, Dec 2009, pp. 268–271.

[21] S. Katz, A. Tal, and R. Basri, "Direct visibility of point sets," *ACM Trans. Graph.*, vol. 26, no. 3, p. 24–es, July 2007. [Online]. Available: https://doi.org/10.1145/1276377.1276407

[22] R. B. Rusu and S. Cousins, "3D is here: Point Cloud Library (PCL)," in *IEEE International Conference on Robotics and Automation (ICRA)*, Shanghai, China, May 9-13 2011.

[23] J. L. Schönberger and J.-M. Frahm, "Structure-from-motion revisited," in *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[24] J. L. Schönberger, E. Zheng, M. Pollefeys, and J.-M. Frahm, "Pixelwise view selection for unstructured multi-view stereo," in *European Conference on Computer Vision (ECCV)*, 2016.

[25] W. E. Lorensen and H. E. Cline, "Marching cubes: A high resolution 3d surface construction algorithm," *SIGGRAPH Comput. Graph.*, vol. 21, no. 4, p. 163–169, Aug. 1987. [Online]. Available: https://doi.org/10.1145/37402.37422