Proceeding of the IEEE
International Conference on Robotics and Biomimetics
Dali, China, December 2019

From Virtuality To Reality: A Learning-based Point Cloud Labeling Method With Synthesis Scene

Runjian Chen, Li Tang, Xiaqing Ding, Yue Wang and Rong Xiong
State Key Laboratory of Industrial Control Technology
Zhejiang University
Hangzhou, Zhejiang Province, China
{*rjchen, ltang, xqding, ywang24, rxiong*}*@zju.edu.cn*

***Abstract*— This paper proposes a machine learning based point cloud labeling algorithm. To classify point cloud in a sparse scan of both virtual and real scene as basic geometrical elements like planar and edge, a rendering dataset in virtual environment is created and labeled. Then the principal component analysis (PCA) is applied to calculate local geometrical features of point cloud. An in-depth analysis is performed by training several machine learning models with PCA features and experiments in which the trained models are applied to on both rendering point cloud and laser scan of real scene are conducted to validate that our approach is scale-invariant and effective on both rendering point cloud and point cloud of real scene.**

***Index Terms*— Point cloud labeling, Synthesis dataset, PCA**

I. Introduction

Point cloud labeling is a task to label points in the point cloud as different kinds, for example edge points, planar points or more semantically, desk points, chair points and so on. It is the fundamental problem to a lot of research problem, such as point cloud descriptor generation, data association between point clouds and so on. And accurate point cloud labeling as basic geometrical elements (egde and planar points) has been proved in many SLAM algorithm, like LOAM [17] and LIPS [8], to have the ability to improve the accuracy of point cloud registration.

There are two main stream of point cloud labeling method in the field. Traditionally, researchers label point cloud with hand-crafted features and empirical thresholds. Harris 3D [16] focus mainly on introducing Harris corner detector from 2-D image to point cloud in 3-D space and achieved excellent result on interest point detection. The edge and corner detection method based on local centroid shift distance in a neighborhood around a point in [1] successfully labels the point cloud from RGB-D sensors and show quite considerable labeling results on 3D point cloud of panel workpiece. Other works like [6] [3] also show robust labeling ability under certain conditions. However, the limits of these methods are quite obvious: the thresholds depend largely on the application scenes and once the scenes change the thresholds should be adjusted to accommodate the new conditions. In recent years, inspired by the trend of learning algorithm applied in images, there are much work on point cloud labeling with machine learning method. And the most representative works are PointNet [13], PointNet++ [14], PointCNN [11] which take the whole point cloud of an object or scene as input to train the model and perform excellently on semantic classification and segmentation of point cloud. These works mainly concentrate on the semantic of a dense set of point cloud. However, in the robotics localization problem, we can only get a sparse and incomplete observation of the environment and have to estimate motion with the sparse set of point cloud. So we need to label point cloud in a single scan of the environment.

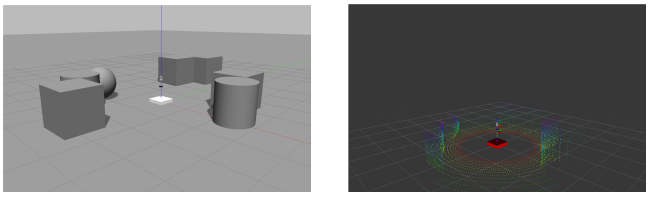
There are various labeled point cloud datasets available, like the KITTI [7], Shape Net [5], Stanford Semantic parsing dataset [2] and so on. However, none of them meet the needs of our method. What we need is multiple frames of point cloud from laser sensor in various positions and directions with labeling information.

Aiming to provide more robust labeling algorithm for better point cloud descriptor generation and localization, this paper focus on applying machine learning algorithm to classify laser points in a scan of real scene as basic geometrical elements: planar, edge and others, including noise, curve edge and curve surface. Due to the lack of dataset that meets our need and inspired by the recent trend to apply model learned from virtual rendering data to real scenes, we first create a multi-scale synthesis dataset that consists of multiple frames of point cloud rendered from virtual HDL-32E laser sensor with random gaussian noise. The virtual environment consists of simple objects, including boxes, cylinders and spheres, which is shown in Fig. 1. And with the position and pose of simple objects, we can easily label points in the rendering data. With this labeled point cloud, we further calculate local geometry features of every point based on PCA, the principal component analysis. We use several machine learning algorithm including SVM with a gaussian kernel [9], KNN using a cubic distance metric [10], weighted KNN [12], Tree model [15], trained with the features. Finally, we apply models learned from rendering data to both untrained rendering data and data from real scene to prove the labeling ability of our method.

978-1-7281-6321-5/19//$31.00 ©2019 IEEE

Our contributions in this paper are summarized as follows:

- A learning-based algorithm is proposed, which is scale-invariant, to label point cloud in a sparse scan as basic geometrical elements.
- A virtual rendering point cloud dataset is created, in which points are labeled as basic geometrical elements.
- Models trained by virtual rendering data is proved to be able to label points in both rendering dataset and a scan of real scene.



(a) Gazebo environment (b) Point clouds in rviz

Fig. 1: Virtual environment

II. Algorithm Details

A. The principal component analysis (PCA)

Our method in local feature calculation was inspired by the PCA, principal component analysis. What PCA does is to project the multi-dimensional data onto the axis on which the covariance of the projected data is maximized. Assume that we have a set of n d-dimensional data, which can be denoted as $X_i = (x_{i1}, x_{i2}, ..., x_{id})^T$, X_i is the i column of the data matrix X i = 1,2,3,...,n , X is an $n \times d$ matrix and what PCA does is to project X on to a l-dimensional space (l¡d) to maximized the covariance of the projected data.

To maximize the covariance of the projected data on the first axis, we aim to find an axis $z = (z_1, z_2, z_3, ..., z_d)^T$, to maximize the covariance of the projected data, that is:

$$max\ Var(z^T X)\ st.\ z^T z = 1 \tag{1}$$

And we could derive the expression of $Var(z^T X)$ as below:

$$\begin{aligned} Var(z^T X) &= E((z^T X)^2) - E(z^T X)^2 \\ &= \sum_{i,j=1}^{d} z_i z_j E(X_i X_j^T) - \sum_{i,j=1}^{d} z_i z_j E(X_i) E(X_j)^T \\ &= \sum_{i,j=1}^{d} z_i z_j S_{ij} \\ &= z^T S z \end{aligned} \tag{2}$$

Where S is the covariance matrix of X

Then we apply Lagrangian Multiplier Method to solve the maximum problem:

$$max\ z^T S z - \lambda(z^T z - 1) \tag{3}$$

Take the derivative of the equation (3) and let it equal 0, we have

$$Sz = \lambda z \tag{4}$$

Thus, if we take the maximum eigen value of the covariance matrix of X, the correspondence eigen vector defines the axis z on which the covariance of the projected data reach the maximum. Besides, the eigen values of the covariance matrix stand for the covariance of the projected data on the axis determined by eigen vectors. And as for the other axis, the derive process is similar and those axis are eigen vectors of the corresponding eigen values.

B. Local feature calculation with PCA

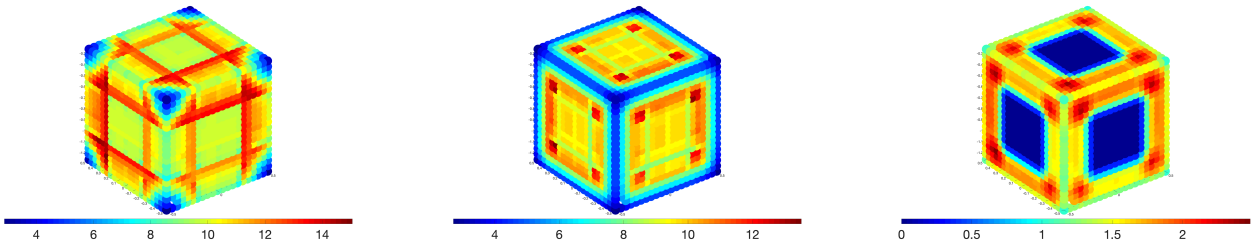
For a query point P, we first search in its neighborhood, which is defined by a Euclidean ball with fixed radius $\boldsymbol{r_{search}}$ and get all the points in the neighborhood. Then we calculate the covariance matrix (a 3×3 matrix) of the (x, y, z) value of the points in the set and attain the three eigen value of the covariance matrix. After calculating PCA eigen values in the Neighborhood, we simply sort the eigen values in decreasing order. And the ordered eigen values serve as the features that we feed into the training model.

As shown in Fig. 2, the PCA eigen value of points in 3-D space is quite an outstanding local geometrical feature for 3-D points. As we can see in the figure, corner points have the weakest response on the max eigen value, edge points have the weakest response on the median eigen value while the planar points have the weakest response on the minimum eigen value. So, the PCA eigen values can be a very excellent feature for the labeling task.

C. Training and validating

Given a set of 4-dimensional training data, $Data = D_i, D_i = (ev_{i1}, ev_{i2}, ev_{i3}, Label_i)$, where $(ev_{i1}, ev_{i2}, ev_{i3})$ are the three PCA eigen values of the i^{th} point in decreased order and $Label_i$ is the class of the i^{th} point, we randomly sample the data to make the number of each class more balanced in order to avoid the bias of machine learning algorithm caused by the uneven distribution of the data. Then we feed the data into four kinds of machine learning classifier, Tree classifier, Cubic KNN classifier, Weighted KNN classifier and Gaussian SVM classifier, and get trained models.

We then apply the trained classifiers to label both rendering data and point cloud in randomly selected frames of the NCLT dataset [4], in which laser point cloud was obtain from a HDL-32E velodyne sensor, the same as the one in the virtual environment, to compare their performance.



(a) Maximum eigen value (b) Median eigen valuen (c) Minimum eigen value

Fig. 2: The three PCA eigen values of each point of a box are sorted in decreasing order and showed respectively in Fig. (a) ,Fig. (b) and Fig. (c). The comparative value are indicated by color. The deep blue indicates the lowest value and the deep red indicates the highest value compared in each figure.

D. Training data preparation

First, we randomly generate the position and pose of the simple objects and the laser scanner and place them in the virtual environment. Once the synthetic point clouds are rendered, we label the points in those frames, with the known respective position and pose between the laser scanner and objects, by Algorithm 1 below. Noting that $P_i = (x_i, y_i, z_i)$ denotes a point in current frame, $|P_{neighbor}|$ denotes the number of the points in neighborhood of P_i, Q denotes the point set of all simple objects in the virtual environment, h denotes the height of the laser scanner above the virtual ground, $EDGE_j$ denotes a edge point in Q, $PLANAR_j$ denotes a planar point in Q, δ_{ground} denotes the threshold indicating how much tolerance we have to identify a point on the ground, δ_{edge} denotes the threshold indicating how much tolerance we have to identify a point near the edge, δ_{planar} denotes the threshold indicating how much tolerance we have to identify a point near the plane, $|\cdot|_2$ denotes the l-2 norm.

Algorithm 1 Label rendering data

Input: $Point\ P_i$
Output: $Type$

set $Type = None$
if $|P_{neighbor}| = 1$ **then**
  set $Type = Others$
else if $|z_i - h|_2 < \delta_{ground}$ **then**
  set $Type = PLANAR$
else if $\exists\ EDGE_j \in Q\ , |EDGE_j - P_i|_2 < \delta_{edge}$ **then**
  set $Type = EDGE$
else if $\exists\ PLANAR_j \in Q\ , |PLANAR_j - P_i|_2 < \delta_{plane}$ **then**
  set $Type = PLANAR$
else
  set $Type = OTHERS$
end if

Noted that we label the intersection of planes, the intersection of plane and edge, the intersection of as edge.

E. Implementation details

As we take Euclidean ball as a neighborhood, the fixed radius should be determined.

To prove the ability for accommodating under different scene, we set $r_{search} = 0.15\ meter$ for both the rendering data and point cloud in real scene.

We set $delta_{ground} = 0.01\ meter$ and $delta_{edge} = delta_{edge} = \frac{r_{search}}{2}$.

III. Experiment and Results

A. Data description

In order to verify the scale-invariant ability of our method, we create four datasets $D_1\ D_2\ D_3\ D_4$. The size of objects in $D_2\ and\ D_4$ is twice as that of $D_1\ and\ D_3$. And the noise intensity of virtual laser scanner in $D_3\ and\ D_4$ is twice as that of $D_1\ and\ D_2$, considering that virtual objects are much smoother than the objects in real scene and we want to compare the performance of models trained by different noise intensity.

In the training and testing process, we divide the four dataset respectively into two halves and use the first half for training and the other for testing.

B. Experiment 1

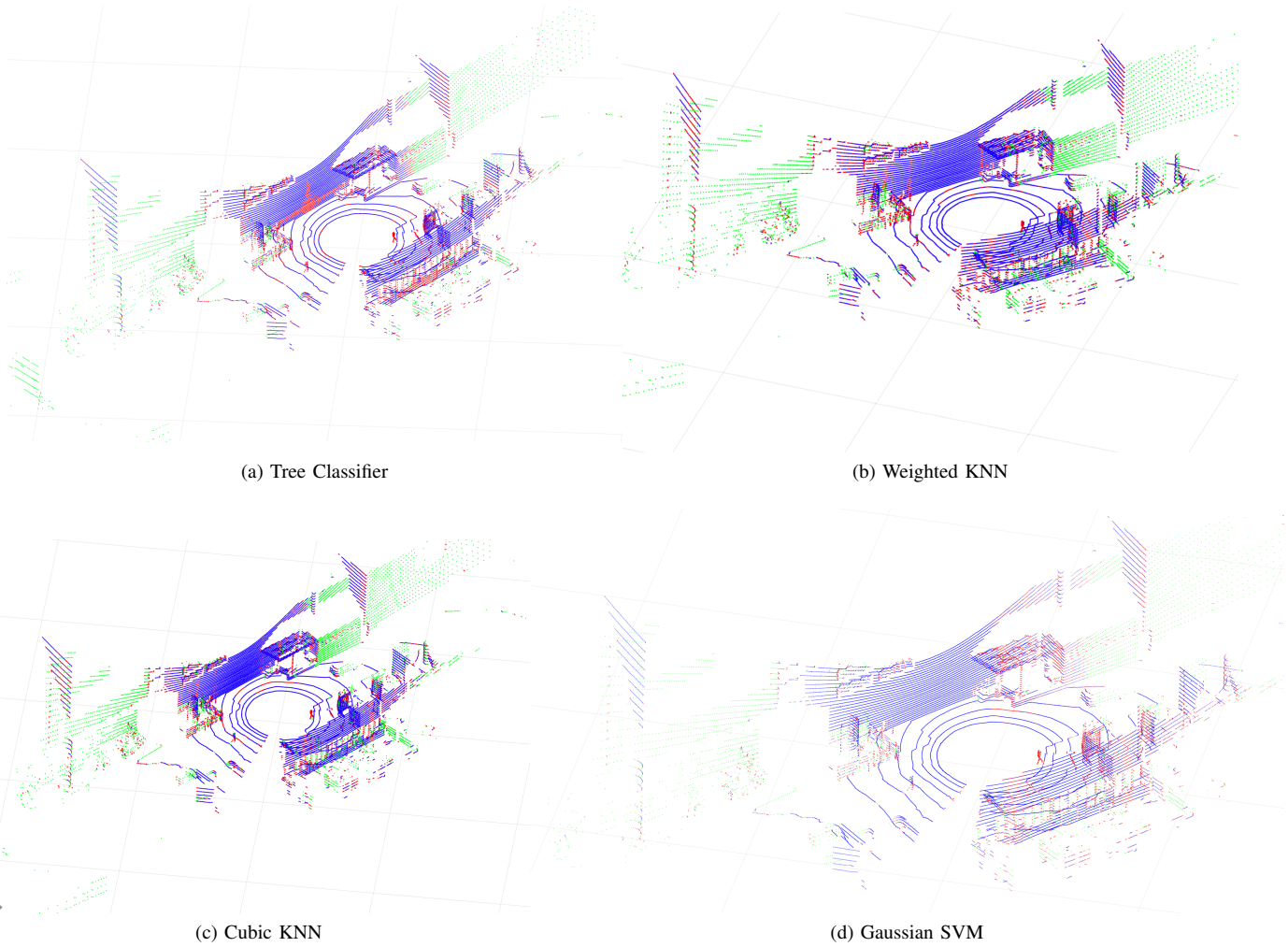
In this experiment, we train and test all the four classifier mentioned above with lower noise and original scale dataset (D_1).

The training validation ratio and testing accuracy are listed in TABLE I.

We then randomly sample a frame of laser point cloud in NCLT dataset [4] and applied the trained model on it. The result is showed in Fig. 3.

TABLE I: Result on rendering data of Experiment 1

$Classifier$	Edge points label accuracy when testing	Planar points label accuracy when testing	Other points label accuracy when testing
$Tree\ classifier$	91.3%	92.9%	99.6%
$Cubic\ KNN$	88.1%	94.4%	99.6%
$Weighted\ KNN$	87.6%	94.2%	99.6%
$Gaussian\ SVM$	85.0%	93.6%	99.6%



(a) Tree Classifier

(b) Weighted KNN

(c) Cubic KNN

(d) Gaussian SVM

Fig. 3: Point cloud label result on the real scene of NCLT dataset in Experiment 1. Blue indicates PLANNAR points, red indicates EDGE points and green indicates OTHER points.

C. Experiment 2

In this experiment, we train Gaussian SVM classifier with lower noise dataset (D_1) and enhanced noise dataset (D_3) respectively and test each trained model on both D_1 and D_3. The results showed in TABLE II on rendering data show an increased performance when enhancing the noise intensity of virtual laser sensor.

Then we applied them on the selected frame of laser point cloud in NCLT dataset [4]. The result is showed in Fig. 4. When we compare the details in the magnifier window, we can find that when the noise is enhanced in virtual environment, the model we train will be more robust and generate less faults on real data.

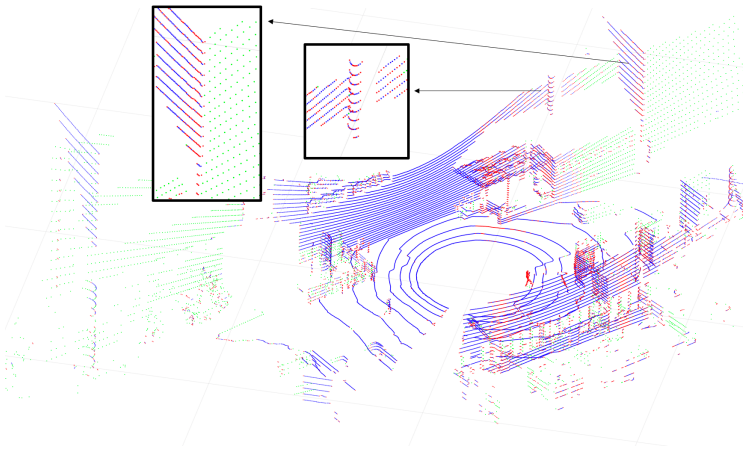
TABLE II: Result tested on rendering data of Experiment 2

Testing condition	Edge points label accuracy when testing[1]	Planar points label accuracy when testing[1]	Other points label accuracy when testing[1]
Tested on lower noise (D_1)	85.0% / 86.8%	93.6% / 93.4%	99.6% / 99.6%
Tested on enhanced noise (D_2)	85.5% / 87.6%	93.5% / 93.3%	99.6% / 99.6%

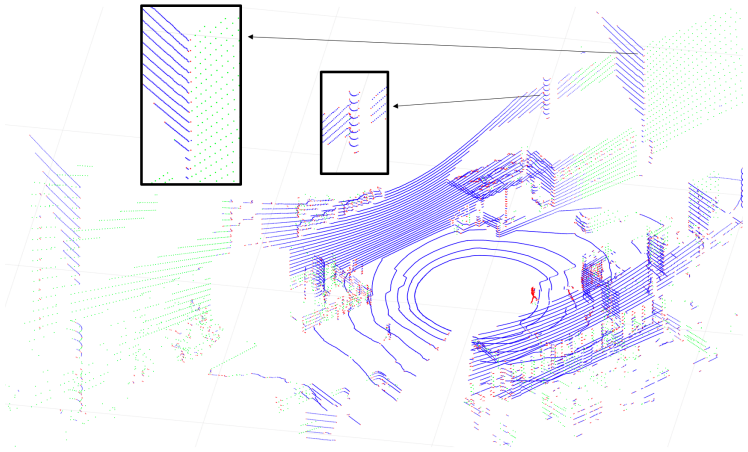
[1] Values represented for model trained on lower noise (D_1) / enhanced noise (D_3)

TABLE III: Result of test on rendering data in Experiment 3

Test condition	Edge points label accuracy when testing	Planar points label accuracy when testing	Other points label accuracy when testing
Tested on original scale (D_2)	87.6%	93.2%	99.6%
Tested on doubled scale (D_4)	87.9%	92.5%	97.0%



(a) Model trained with lower noise (D_1)

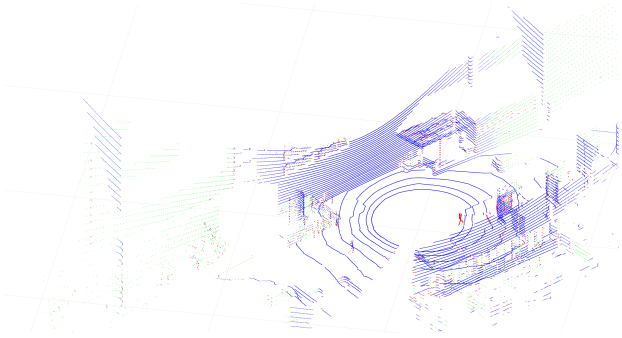


(b) Model trained with enhanced noise (D_3)

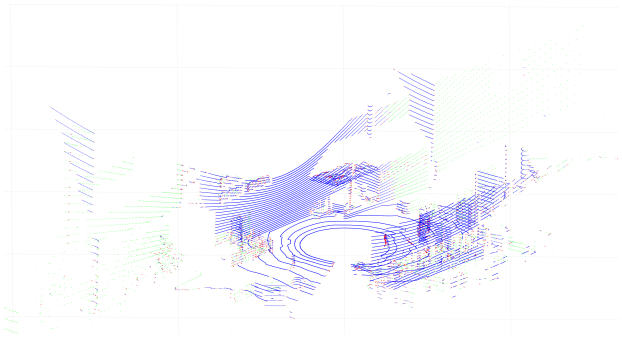
Fig. 4: Point cloud label result on the real scene of NCLT dataset in Experiment 2. Blue indicates PLANNAR points, red indicates EDGE points and green indicates OTHER points.

D. Experiment 3

In this experiment, we train Gaussian SVM classifier with original scale and enhanced noise (D_3) and test the trained model on both original scale with enhanced noise (D_3) and



(a) Model trained with original scale (D_2)



(b) Model trained with doubled scale (D_4)

Fig. 5: Point cloud label result on the real scene of NCLT dataset in Experiment 3. Blue indicates PLANNAR points, red indicates EDGE points and green indicates OTHER points.

doubled scale with enhanced noise (D_4). The results showed in TABLE III on rendering data show model trained with D_3 perform almost the same on both scales, proving the scale invariant ability of our method.

Then we train Gaussian SVM classifier with (D_3) and (D_4) and applied each trained model on the selected frame of laser point cloud in NCLT dataset [4]. The result is showed in Fig. 5. When we compare the results in details, we can find that scale of the virtual object has little influence on the result of labeling real scene.

IV. Conclusion

In this paper, we proposed a learning-based point cloud labeling method, which uses rendering data in synthesis scenes. Both the quantitative and qualitative results show that the feature we propose is able to label point cloud, in a sparse scan of both virtual and real scene, as basic geometrical elements. We also create a rendering dataset, which is different from other existent dataset and meets the need of our method. It is revealed that our method is scale-invariant and able to apply to real scene though only trained by virtual data.

Acknowledgment

This work was supported by the Science and Technology Project of Zhejiang Province (Grant No. 2019C01043), Science and Technology on Space Intelligent Control Laboratory (No.HTKJ2019KL502002), and the Fundamental Research Funds for the Central Universities

References

[1] Syeda Mariam Ahmed, Yan Zhi Tan, Chee Meng Chew, Abdullah Al Mamun, and Fook Seng Wong. Edge and corner detection for unorganized 3d point clouds with application to robotic welding. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 7350–7355. IEEE, 2018.

[2] I. Armeni, A. Sax, A. R. Zamir, and S. Savarese. Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*, February 2017.

[3] Dena Bazazian, Josep R Casas, and Javier Ruiz-Hidalgo. Fast and robust edge extraction in unorganized point clouds. In *2015 International Conference on Digital Image Computing: Techniques and Applications (DICTA)*, pages 1–8. IEEE, 2015.

[4] Nicholas Carlevaris-Bianco, Arash K Ushani, and Ryan M Eustice. University of michigan north campus long-term vision and lidar dataset. *The International Journal of Robotics Research*, 35(9):1023–1035, 2016.

[5] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015.

[6] Changhyun Choi, Alexander JB Trevor, and Henrik I Christensen. Rgb-d edge detection and edge-based registration. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1568–1575. IEEE, 2013.

[7] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013.

[8] Patrick Geneva, Kevin Eckenhoff, Yulin Yang, and Guoquan Huang. Lips: Lidar-inertial 3d plane slam. In *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 123–130. IEEE, 2018.

[9] S Sathiya Keerthi and Chih-Jen Lin. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation*, 15(7):1667–1689, 2003.

[10] Laszlo Kozma. k nearest neighbors algorithm (knn). *Helsinki University of Technology*, 2008.

[11] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Advances in Neural Information Processing Systems*, pages 820–830, 2018.

[12] Wei Liu and Sanjay Chawla. Class confidence weighted knn algorithms for imbalanced data sets. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 345–356. Springer, 2011.

[13] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas 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*, pages 652–660, 2017.

[14] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5099–5108, 2017.

[15] S Rasoul Safavian and David Landgrebe. A survey of decision tree classifier methodology. *IEEE transactions on systems, man, and cybernetics*, 21(3):660–674, 1991.

[16] Ivan Sipiran and Benjamin Bustos. Harris 3d: a robust extension of the harris operator for interest point detection on 3d meshes. *The Visual Computer*, 27(11):963, 2011.

[17] Ji Zhang and Sanjiv Singh. Loam: Lidar odometry and mapping in real-time. In *Robotics: Science and Systems*, volume 2, page 9, 2014.