Master Project

Landmark detection and classification from point clouds

Akhil Thomas

Examiner: Prof. Dr. Wolfram Burgard

Adviser: Tayyab Naseer

Albert-Ludwigs-University Freiburg
Faculty of Engineering
Department of Computer Science
Autonomous Intelligent Systems

June 16th, 2017

Abstract

Detecting and classifying landmarks is essential to a wide range of topics including Simultaneous Localization and Mapping, urban modeling, surveillance etc. Landmarks of interest is specific to the application; trees and pole-like structures are often the favorites. Point cloud data gives more information about the underlying environment than images, and hence the problem of landmark detection and classification is intuitively easier from point clouds than with images.In this research two novel functional pipelines for detecting and classifying landmarks, specifically pole-like objects and trees, from point clouds are designed. The belief that better clusters are a prerequisite for better classification is the prime motivation for these approaches. Both the designed tools use Difference of Normals based segmentation and thresholding. One tool detects only pole-like structures by applying simple rules on the extracted cluster. However in the second tool, a combination of features, namely ensemble of shape functions and eigenvalue based, are extracted from the clusters. Later, a random forest multi-class classifier is trained to classify the clusters as pole-like objects, trees or neither of them. The designed pipeline is trained and tested using two point cloud maps of the Technical Faculty at University of Freiburg.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

Effective detection and classification of landmarks in an environment is a crucial component to many applications. Some more prominent applications include Simultaneous Localization and Mapping (SLAM), autonomous driving, urban modelling, surveillance, and road safety assessments [1, 2]. The landmarks of interest varies as per the application. For autonomous driving, classifying signboards, traffic lights etc are important; meanwhile in the case of forest surveillance, detecting trees are of more importance. For a general SLAM problem, the landmarks of interest may include trees, traffic lights, lamp posts, buildings etc.

Point Cloud Library (PCL) [3] defines a point cloud as a data structure used to represent a collection of multi-dimensional points. It is commonly used to represent three-dimensional data. Most 3D sensors available in market, stereo cameras or Light Detection and Ranging (LIDAR) sensors can output their data as point clouds; or their output could be converted to point clouds. A 3-D point cloud is a set of points that have x, y, and z dimensions. Such a point cloud could be made more informative by adding more dimensions to it, like colour, normals, curvature etc. It is hence quite advantageous to use point clouds for detection and classification problems. They can store more information about the captured environment than its two-dimensional counterpart, images. Detecting various kinds of landmarks is intuitively more reliable and easier from point clouds than with images. The research described in this report produces tools that could detect landmarks, specifically trees and poles, from point clouds.

1.1 Motivation

An ongoing research at Autonomous Intelligent Systems group in University of Freiburg required a lot of labeled training images of landmarks. Application being SLAM, the landmarks could include different types of poles, trees and buildings. These could be then used to train a neural network for images. This task could be simplified by the availability of a tool which can automatically detect and label such

landmarks on 2D images. As discussed in the introduction, such a tool could be intuitively developed relatively easier on point clouds. And if we know the extrinsic calibration between the used 2D and 3D sensors, then this tool could be used to label data on 2D images. Developing such a tool that could detect landmarks from point clouds was the motivation for this research.

1.2 Problem statement

The problem tackled by this research is to develop a tool that autonomously detects and classifies the following landmarks from the input point cloud:

- Vertical poles like objects (PLOs) - including lamp posts, direction boards, sign boards, traffic lights etc.,
- Trees.

Additionally, the following features of the landmarks have to be characterized:

- Centroid
- Height

The tool should be easily extendable to include more landmarks. Further, the problem is constrained by the unavailability of much training data.

2 Related Work

The task of identifying landmarks from point cloud have been taken up by many researchers. If we would generalize the approach followed by researchers, it would involve three main steps. Firstly, preprocessing of point clouds to remove outliers, followed by segmentation to extract clusters, and later detecting and classifying landmarks of interest from the extracted clusters.

To detect specifically pole-like or vertical structures have also been considered by many as an interesting topic. They also follow the generalized approach described above. A well cited recent approach is the one from Yokoyama etal. [4]. Their topic was to detect PLOs from urban LIDAR data and classifying them. In their approach, ground removal was followed by segmentation and later by smoothing. Then each point in each segment is classified to be part of PLO or not. They used Principal Component Analysis based local features for this. Then depending on the extent of PLOs points in each segment, the segment is classified to PLOs or not. Further, they used both shape and context features to classify into various kinds of PLOs. Ishikawa [5] used Support Vector Machines (SVM) to detect road objects including PLOs and walls. Their approach included building blocks from the input point cloud, and then extract features and later use a trained SVM to classify them. An accurate segmentation step was missing from both these cases.

Lehtomäki et al. [6], another well cited approach, tackled the problem of detecting PLOs by using scan line based segmentation. Each scan line is individually segmented and later they are clustered together based on several rules. And some set of rules are checked with the final clusters to classify them. Landa etal. [7] in a way generalized this approach by using horizontal cut based segmentation instead of scan line based. The input point cloud was cut into several horizontal cuts. Euclidean clustering was performed on these cuts individually. The clusters which are on top of each other are then stitched together using angle to vertical axis as a criterion. And later, the stitched clusters are detected as poles based on height and angle to vertical. Such approaches actually helps in removing problems that euclidean clustering would have if the poles or trees are connected to other objects. However, extending such

approaches to include more landmarks is not a straightforward task.

In the segmentation part, Difference of Normals (DON) based segmentation [8] is a recent interesting approach that helps in extracting better clusters. It is basically the difference in normals estimated on the point cloud using a smaller radius and a larger radius. Depending on the smaller and larger radii chosen, different objects are segmented in a very nice way. This feature is included in the presented research.

For feature extraction part, an appropriate feature descriptor can be chosen from two main categories: local descriptors and global descriptors. A local descriptors evaluates features for each individual point in the point cloud. However, a cluster of points is given as input to the global descriptor. Most prominent local descriptors include Signature of Histograms of Orientations, Fast Point Feature Histogram [9] etc. Global descriptors include Viewpoint Feature Histogram, Point Feature Histogram, Ensemble of Shape Functions (ESF) [10] etc. Segmatch [2] is a recent approach that used global feature descriptors in their pipeline. Their aim was to detect similar segments to help in loop-closure of SLAM algorithm. They used global feature descriptors ESF and an eigenvalue based feature descriptor [9]. They further trained a random forest to classify clusters as "seen before" or "new cluster". A similar approach is used in the classification part of the presented research.

3 Approach

Two different tools which are designed to solve the problem detailed in Section 1.2 are presented in this section. Both are inspired from the generic approach deducted from the literature review. Both tools have three steps: preprocessing, segmentation and then detection/classification. Both have the same preprocessing and segmentation steps, but they differ in the detection/classification step. DON based segmentation is used by the tools in segmentation step, which is the main reason behind their success. In the first tool, pole detector, simple rule based detection step is used to detect only poles. However the second tool, landmark classifier, detects and classifies poles and trees using a trained multi-class classifier.

3.1 Pole detector overview

Algorithm 1 Initial Approach

Input: Point cloud as pcd file
Result: $\mathbf{l_{poles}}$, List of detected PLOs.
Read point cloud $\mathbf{X}$ from pcd file
$\mathbf{X'} \leftarrow$ statisticalOutlierRemover($\mathbf{X}$) ▷ Perform outlier removal
$\mathbf{X_p} \leftarrow$ groundPlaneRemover($\mathbf{X'}$) ▷ Remove ground plane
$\mathbf{l_{clusters}} \leftarrow$ euclideancluster($\mathbf{X_p}$) ▷ Euclidean clustering
List $\mathbf{l_{stitched}} \leftarrow$ clusterStitcher($\mathbf{l_{clusters}}$) ▷ Stitching vertical clusters
foreach cluster $\mathbf{c} \in \mathbf{l_{stitched}}$ **do**
    height, xyBound, heightAboveGnd $\leftarrow$ findFeatures($\mathbf{c}$)
    if isPole(height, xyBound, heightAboveGnd) = true **then**
        add $\mathbf{c}$ to list $\mathbf{l_{poles}}$ ▷ clusters satisfying the rules are poles
    end if
end for

In the initial phase of this research a very simple approach was used to detect only poles, as detailed in Algorithm 1 and represented in Figure 1. This is based on the generalized approach deducted from literature review. The approach starts with preprocessing, which includes outlier removal and ground plane removal. It is

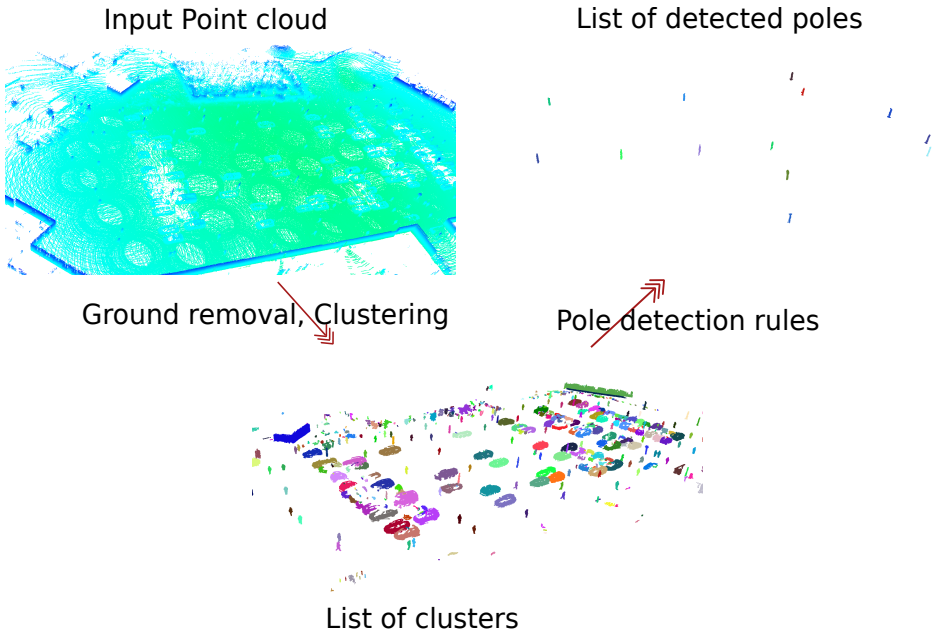


Figure 1: Representation of the initial approach

followed by Euclidean distance based segmentation. Then, a step of cluster stitching is used to merge together vertical clusters which are fragmented.

Later, pole detection is achieved based on some rules. These rules can be expressed as : "If the cluster is 'vertical', 'tall', 'thin' and 'standing on ground', then it is a

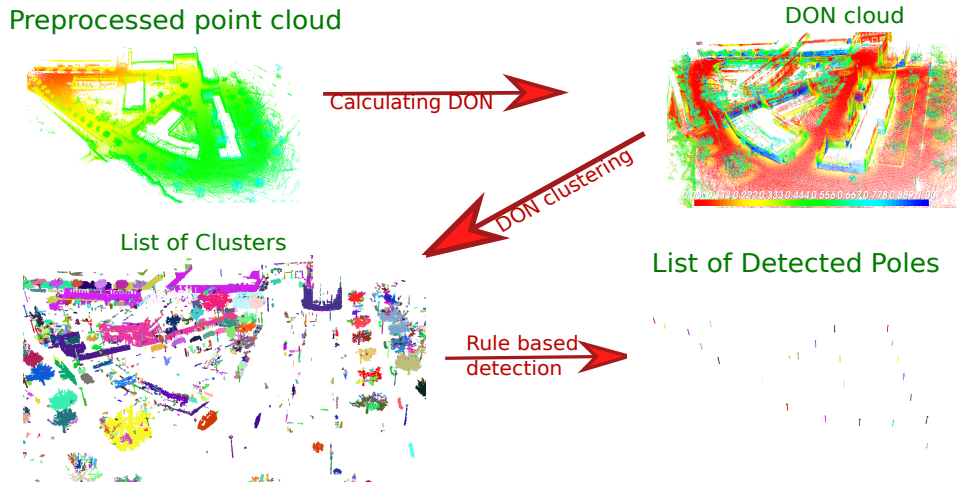


Figure 2: Representation of the pole detector tool

Algorithm 2 Pole Detector

Input: Point cloud as pcd file
Result: $\mathbf{l_{poles}}$, List of detected PLOs.
Read point cloud $\mathbf{X}$ from pcd file
$\mathbf{X'} \leftarrow$ statisticalOutlierRemover($\mathbf{X}$) ▷ Perform outlier removal
$\mathbf{l_{clusters}} \leftarrow$ **DonBasedSegmenter**($\mathbf{X_p}$) ▷ DON thresholding, clustering
List $\mathbf{l_{stitched}} \leftarrow$ clusterStitcher($\mathbf{l_{clusters}}$) ▷ Stitching vertical clusters
foreach cluster $\mathbf{x} \in \mathbf{l_{stitched}}$ **do**
    height, xyBound, heightAboveGnd $\leftarrow$ findFeatures($\mathbf{x}$)
    if isPole(height, xyBound, heightAboveGnd) = true **then**
        add $\mathbf{x}$ to list $\mathbf{l_{poles}}$ ▷ clusters satisfying the rules are poles
    end if
end for

pole" . We used custom values to threshold pole's angle to vertical, height, thickness, and height from ground. In the pole detection rules, it was initially checked if the height of the cluster is more than a threshold, we used 3m. If a cluster satisfies this condition, a bounding box was formed for it and a condition on the box's xy-plane dimension was checked. A condition based on height of cluster above ground was then checked to remove false positives like building components.

However using the feedback from results, it was realized that the segmentation step of the initial approach needs to be changed. DON [8] based segmentation was hence introduced. Also, the ground plane remover in the initial approach was replaced with DON thresholding. This resulted in the current state of pole detector tool, as described in Algorithm 2.

3.2 Landmark classifier overview

The landmark classifier tool differs only in the classification/detection part from the pole detector tool. This tool uses a trained random forest classifier to classify the clusters into three landmark classes: poles, trees or neither. The algorithm is condensed to Algorithm 3 and is represented in Figure 3.

Preprocessing is done at first to remove the outliers. After that DON based thresholding is done which removes ground points and several other unnecessary points. Next, DON based segmentation is performed. This results in clusters which are better than normal euclidean clustering. Further for each of the clusters, ESF and eigenvalue based features are evaluated. The selection of these features is justified in

Algorithm 3 Landmark Classifier

Input: Point cloud as pcd file
Result: $\mathbf{l_{classified}}$, List of clusters classfied as PLOs, trees or None.
Read point cloud $\mathbf{X}$ from pcd file
$\mathbf{X_p} \leftarrow$ statisticalOutlierRemover($\mathbf{X}$) ▷ Perform outlier removal
$\mathbf{l_{clusters}} \leftarrow$ **DonBasedSegmenter**($\mathbf{X_p}$) ▷ DON thresholding, clustering
List $\mathbf{l_{stitched}} \leftarrow$ clusterStitcher($\mathbf{l_{clusters}}$) ▷ Stitching vertical clusters
List $\mathbf{l_{feature}} \leftarrow$ **FeatureExtractor**($\mathbf{l_{stitched}}$) ▷ Extracting features for clusters
foreach cluster $\mathbf{x_f} \in \mathbf{l_{feature}}$ **do**
    $\mathbf{x_{classified}} \leftarrow$ randomForestClassifier($\mathbf{x_f}$) ▷ clusters are classified
    add $\mathbf{x_{classified}}$ to $\mathbf{l_{classified}}$
end for

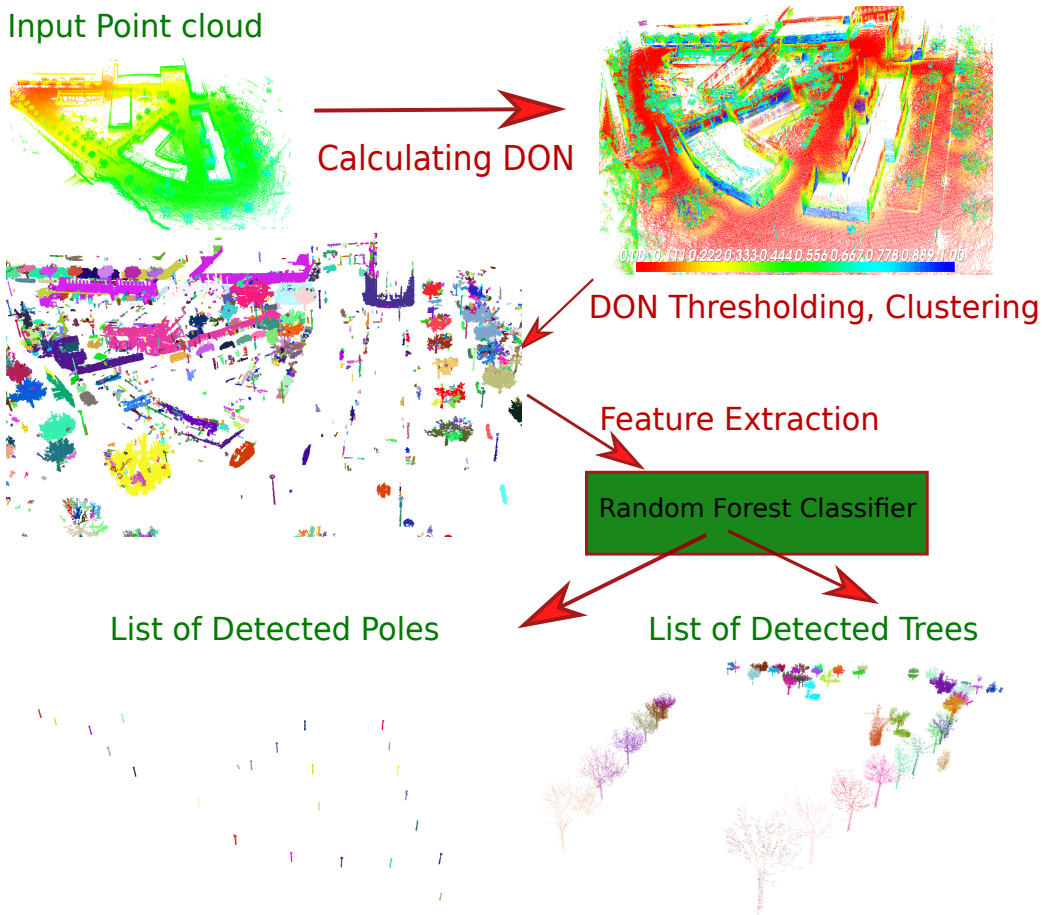


Figure 3: Representation of the landmark classifier tool

Section 3.5. These features are then fed to a trained random forest classifier which classifies the cluster as one among "Pole", "Tree" and "None" types.

3.3 Preprocessing

This section describes the preprocessing step of both pole detector and landmark classifier tools. In the final version of both approaches, this step includes only outlier removal.

Typical point cloud data obtained from sensor hardwares will include outliers. Outliers could also be generated while building a map out of individual 3D-sensor scans. The level of outliers present is however relative to application and sensing hardware. These outliers can cause significant errors while performing calculations on the point cloud like normal estimation, curvature estimation, feature extraction etc. Removal of outliers is however very important. In both the approaches, a statistical outlier removal filter from PCL [3] is used. The filter needs inputs mean and standard deviation to represent the underlying distribution. And for each point in the point clouds, the mean distance to its neighbor points is calculated. If this distance does not fit as per the normal distribution specified, then the point is considered as an outlier and is removed. Few trials with mean and standard deviation helped in finding a good approximation for the normal distribution. However, this has to be redone for a new point cloud input as per the level of outlier present.

A Ground plane removal step was included in the initial version of pole detector. This step used random sample consensus (RANSAC) method in PCL [3] to fit plane model iteratively to the point cloud. The step involved tuning of a parameter called distance threshold, which determines whether a given point is an inlier to the fitted model or not. The plane that is fitted at each iteration is removed from the point cloud and the iteration continues until it reduces the number of points in the point cloud to a set limit.

3.4 DON based segmentation

As described in Algorithm 1, the initial approach used simple euclidean distance based segmentation. However, the results from this approach had poor recall as well as poor precision. The main defect in this approach was deduced to be the inaccurate clusters that were formed in the segmentation step. It was noted that the initial approach didn't include normal information. However normals are promising with

regard to segmentation. Hence, an approach of segmentation using normals was adopted, namely DON based segmentation [8].

This new segmentation approach included calculation of DON from input point cloud, applying a threshold to remove undesired points and clustering using DON calculated curvature on the DON-cloud. DON is essentially the difference in normals which are estimated using a small radius and a large radius on the point cloud. This calculation provides normals along each direction as well as curvature. The estimated curvature however depends on the radii pair used. This curvature is then used for both thresholding and segmentation. Thresholding the point cloud using DON curvature is more effective than just ground plane removal. Also, the curvature value based segmentation is very much effective than euclidean distance based segmentation. Hence, the algorithm pipeline was modified by removing ground plane removal from preprocessing, and euclidean clustering from segmentation to include DON based thresholding and segmentation. The algorithm for DON part is shown in Algorithm 4 and represented in Figure 4.

However, three parameters had to be optimized here. They are the DON parameters: small and large radii, and clustering parameter: cluster tolerance.

Algorithm 4 DON based thresholding and segmentation

Input: Preprocessed Point cloud $\mathbf{X_p}$
Result: $\mathbf{l_c}$ - List of clusters
$\mathbf{X_{DON}} \leftarrow$ DonCalculator($\mathbf{X_p}$) ▷ DON calculation on input
$\mathbf{X_{th}} \leftarrow$ DonThresholder($\mathbf{X_{DON}}$) ▷ DON thresholding - removes ground
$\mathbf{l_{clusters}} \leftarrow$ DonBasedClustering($\mathbf{X_{th}}$) ▷ DON based clustering

Figure 4: Block diagram of segmentation part using DON

3.5 Random forest classifier

This section describes the feature extraction and random forest classifier used in Algorithm 3 (landmark classifier). When the pole detector described in Algorithm 2 was extended to detect trees, it gave very poor results. The extension was done by merging clusters which are nearby and applying a new set of rules with xy bound

and height. The reason for failure was deducted to be the lack of features used for describing the clusters. Hence it was decided to use a state of the art classifier with good feature descriptors for the purpose.

In Segmatch algorithm [2], they used two feature descriptors and trained a random forest classifier to classify segmented clusters as seen before or not. Motivated by this approach, eigenvalue based features, ESF features and their combination was tested. A Kd-tree of extracted features from a small set of reference trees and poles was formed. Then, each cluster was matched against this Kd-tree to select the nearest neighbor. The cluster was then was allotted the same class as its nearest neighbor. This way, a framework to compare feature descriptors was made. From the comparison, the combination of the above feature descriptors was selected to be the best.

A cluster filter was added to the algorithm before feature extraction. The filter applied height and a less strict xy-bound on the clusters. The intuition being, we don not want very short or very large clusters. This significantly reduced the number of candidate clusters. For the passed clusters, ESF feature and Eigenvalue based features were computed. The final feature extractor algorithm is described in Algorithm 5.

Algorithm 5 Feature Extractor

Input: $\mathbf{l_{stitched}}$ - List of stitched point cloud clusters
Result: $\mathbf{l_{feature}}$ - List of clusters with features extracted
foreach cluster $\mathbf{x} \in \mathbf{l_{stitched}}$ **do**
  if clusterFilter($\mathbf{x}$) = false **then**
    $\mathbf{x'} \leftarrow$ EigenFeatureExtractor($\mathbf{x}$)
    $\mathbf{x_f} \leftarrow$ EsfFeatureExtractor($\mathbf{x'}$)
    add $\mathbf{x_f}$ to $\mathbf{l_{feature}}$
  end if
end for

A state of the art classifier, random forest was selected as the classifier for the approach. Even though there was a shortage of training data, it is in belief that it could be re-trained at a later point of time with more training data that this decision was taken. Also, it would be easier to extend to include more landmarks. The training data, clusters labeled as "Trees", "Poles" or "None" was obtained semi-autonomously. The poles were detected using the elementary pole detector described earlier with manual correction. For trees, the same pole detector was tweaked and used with more amount of manual correction. Many of the false positives of the above approach was labeled as "None" class. Also, some random clusters were generated for this class.

The trained classifier is fed with clusters with features extracted. And it predicts which class it belongs to.

4 Results and Discussion

In this chapter, the results of testing the designed tools on two maps of University of Freiburg Technical Faculty campus, namely map A and map B, are presented. The map A was obtained from Velodyne HDL-64E LIDAR scans and map B from Velodyne HDL-32E LIDAR scans. The results of using the initial approach, with euclidean distance clustering, is presented first. Later, inclusion of DON to the pole detector is presented. This section also includes the optimization of DON parameters - the radii pair and cluster tolerance. The final section shows the results of using the landmark classifier on the above maps. This section also justifies the adoption of the selected feature descriptors.

4.1 Initial approach

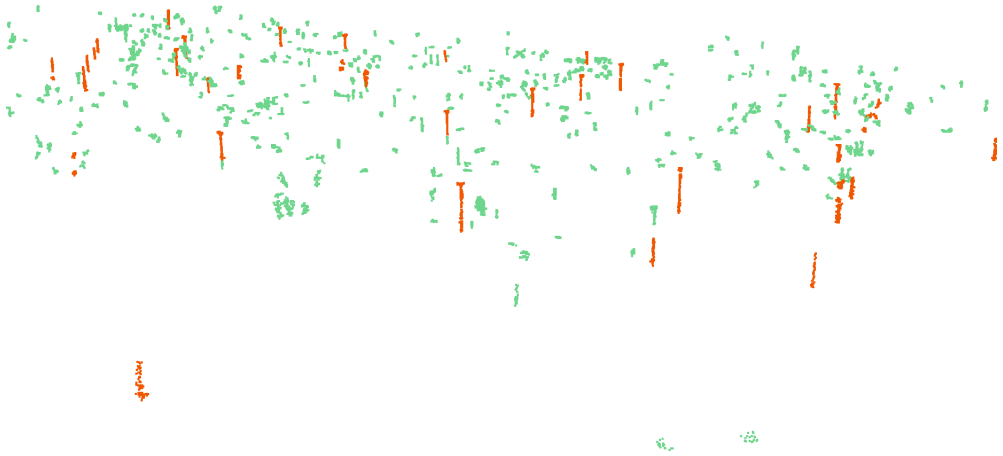


Figure 5: Clusters (green) overlaid with poles (orange) detected using initial pole detector

The results of initial approach described in Algorithm 1 on map A is shown in

Figure 5. The number of true poles detected in this case was 19, out of a total of 37 detections; the ground truth being 29. This result lacked both recall and precision. Also, from the figure it can be seen that most of the poles detected were of low quality. In general, most of the clusters were poorly segmented. This is deducted to be due to the inefficiency of only using euclidean distance for clustering.

4.2 Pole detector

The tuning of DON parameters are discussed first, followed by comparison of pole detector with initial approach.

4.2.1 DON parameter tuning

As mentioned in Section 3.4, calculating DON on the point cloud will result in extracting more information about the underlying environment. Of the information extracted, the curvature value estimated is used in the subsequent processes. The curvature value estimated for the map A is shown in Figure 6. It can be seen from the figure that the trees and poles is distinguishable from its surroundings.

As discussed in Section 3.4, the performance of DON-based clustering relies heavily on three parameters: the smaller radius, larger radius and cluster tolerance. The

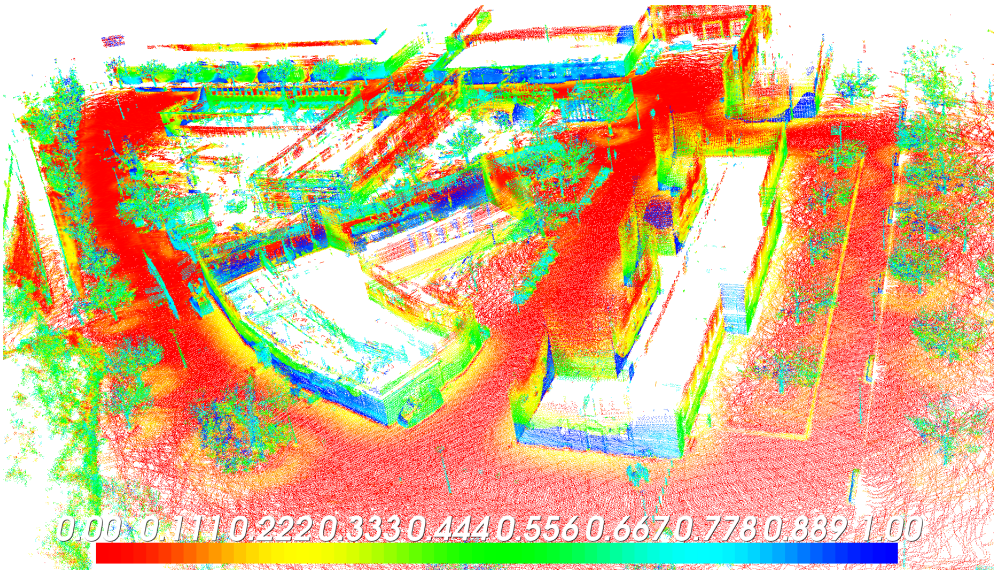


Figure 6: DON cloud showing the variation of curvature value across the point cloud

Cluster Tol.	# poles det.	# true pos.	recall	precision
0.15	35	25	0.86	0.71
0.2	33	23	0.79	0.66
0.25	30	23	0.79	0.73
0.3	28	22	0.76	0.79
0.4	26	22	0.76	0.85

Table 1: Table comparing the performance of pole detector for varying values of DON cluster tolerance

authors of [8] state that the bigger the parameter values are, the more favorable it is towards bigger objects. Also, they have advised to keep the ratio of larger to smaller radius as 10. Thus, in effect the parameters to be tuned became two - the radii pair and cluster tolerance.

The tuning of radii pair was done by getting feedback from the clusters generated. It was observed from experiments that higher DON radii made better clusters for poles as well as trees. This is probably because the ground plane was removed very well by the thresholding step and hence the clusters were disjoint with ground points. However, higher values needed more computation time; In addition they will make clusters combining poles with adjacent objects. An appropriate value was chosen

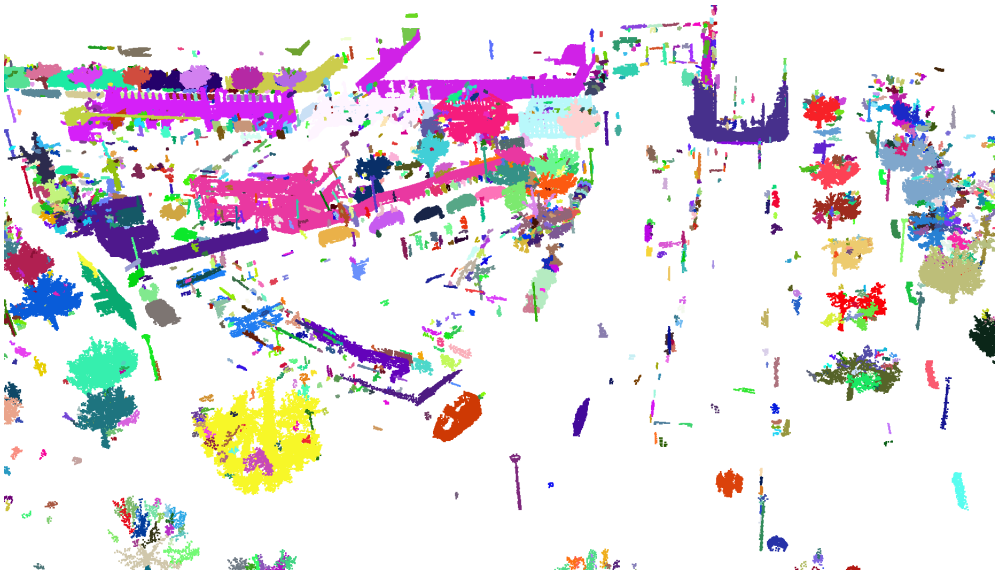


Figure 7: Clusters (random colors) extracted after tuning DON parameters and cluster tolerance

considering this trade-off.

The authors of [8] had kept cluster tolerance to be a constant for a particular DON radii pair. However, from the feedback of clusters, it was realized that the clustering tolerance is also a key parameter to be tuned. It was observed that lower cluster tolerance generated well separated clusters, however the clusters represented the shape of its objects better with higher cluster tolerance. This trade-off was kept under consideration while choosing this parameter Table 1. This tuning part helped in producing better clusters than before. Such a cluster is shown in Figure 7.

4.2.2 Comparison with initial approach

After combining DON based clustering to the aforementioned initial approach to detect poles, the results significantly improved. It was observed that when the xy-bound rule described in Section 3.1 is very strict, it could result in a high recall for poles since the clusters formed were of good quality. The results obtained on map A with ground truth of 29 poles is summarized in Table 2. The detected poles are overlaid with extracted clusters in Figure 8.

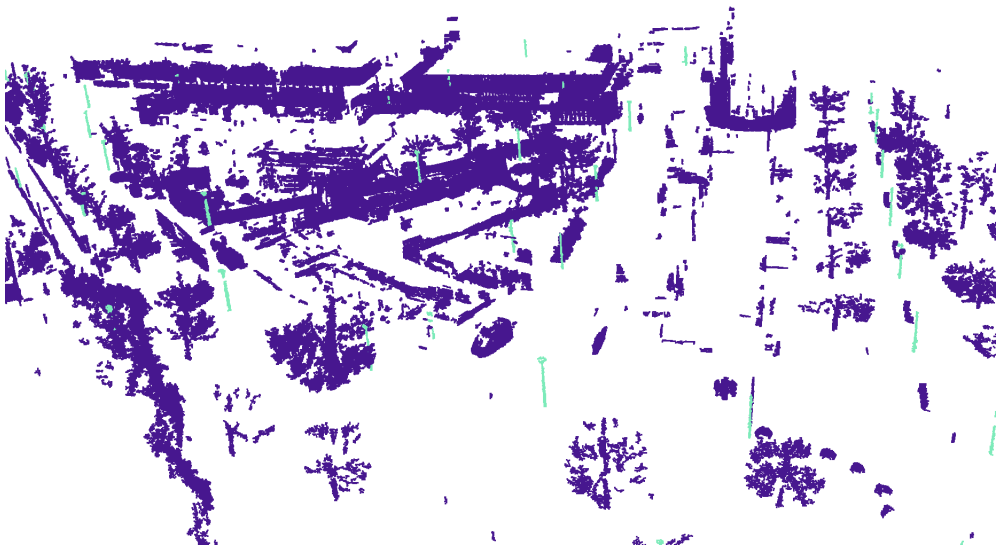


Figure 8: Clusters (violet) overlaid with poles (green) detected using pole detector with DON segmentation

Approach	# poles det.	# true pos.	recall	precision
Initial approach	37	19	0.66	0.51
Pole detector	35	25	0.86	0.71

Table 2: Table comparing the performance of pole detector tool with the initial approach

4.3 Landmark classifier

As discussed in Section 3.5, feature extraction step and random forest classifier was added to extend the algorithm to include more landmarks. In the research, we restricted to including additionally only trees. The feature selection and random forest performance is described in this section.

4.3.1 Feature selection

ESF features, eigenvalue based features and combination of both were the candidates for feature descriptor. To select the best feature descriptor, a framework using Kd-tree was designed as discussed in Section 3.5. Then it was tested on map A, which has 29 poles. It was observed that combination of both features gave better precision and

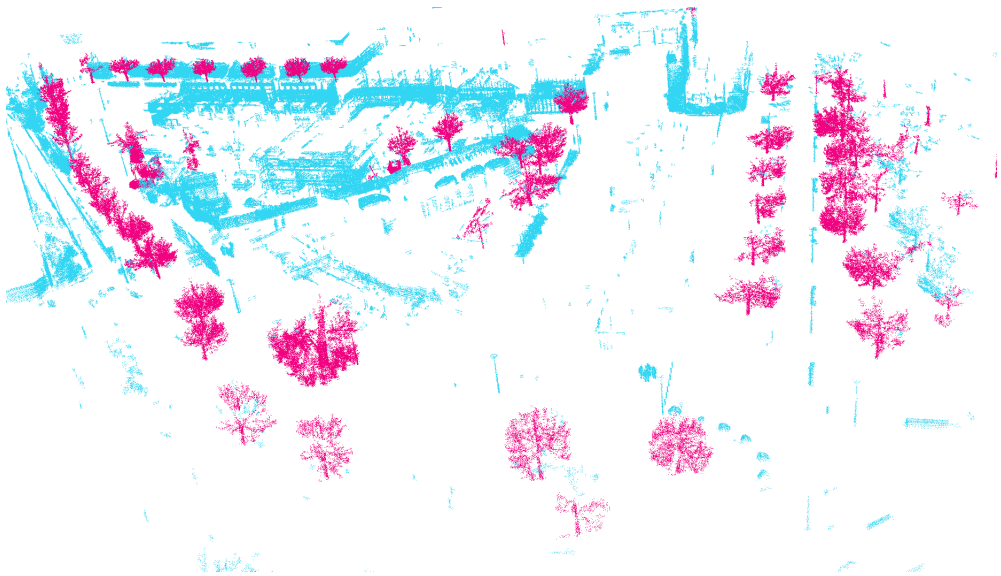


Figure 9: Clusters (blue) overlaid with trees (pink) detected using extracted combined esf and eigenvalue features

Feature Descriptor	# poles det.	# true pos.	recall	precision
ESF	31	22	0.76	0.71
Eigenvalue based	44	24	0.83	0.55
Combined	29	24	0.83	0.83

Table 3: Table comparing the performance of the different feature descriptors

recall than individual features. Table 3 shows that eigenvalue based feature had more recall than esf, however it was less precise. Combining both features gave a recall as good as eigenvalue based, and more precise than both of the individual ones.

Using a feature descriptor helped in including trees for classification. Trees detected using combination of ESF and eigenvalue feature descriptors is shown in Figure 9.

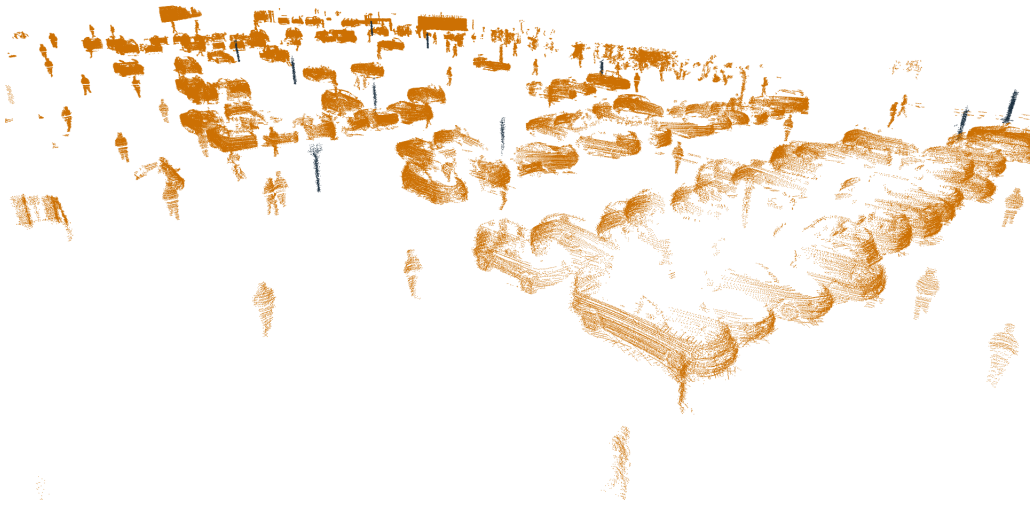
4.3.2 Landmark classifier performance

Once the features are extracted, a multi-class random forest classifier is trained. The training data is obtained from labeling some of the trees and poles in maps A and B. Labeling of training data is done as described in Section 3.5. The training data was however very small; including only 15 poles, 19 trees and 88 of class neither tree nor pole.

After using the best trained random forest classifier, the pole detection results are shown in Table 4; note that the poles in training data are removed from ground truth. It can be seen that the result became more precise. However, the recall was less compared to previous approaches. The latter is most likely due to the training data deficit. The poles detected from map B and trees detected from map A using this approach is shown in Figure 10.

Maps	# poles det.	# true pos.	recall	precision
Map A	16	13	0.65	0.72
Map B	8	8	0.73	1

Table 4: Table showing the performance of the trained random forest classifier on the two maps. Ground truth of poles (excluding those in training set) in map A was 20, and that in map B was 11.



(a) Poles detected overlaid with extracted clusters by landmark classifier tool

(b) Trees detected by landmark classifier tool

Figure 10: Trees and Poles classified by the landmark classifier tool after training random forest classifier

5 Conclusion

The research presented was successful in meeting the problem requirements as stated in Section 1.2. Two tools were developed to extract landmarks from point clouds.

The first tool, pole detector was successful in detecting poles from the input with high recall. Experimenting with pole detector gave the insight that good segmentation helps to obtain better results, even with very simple rules in the detection part. But this tool could not be extended easily to include more landmarks. However, the high recall of the tool makes it eligible to be starting filter tool for pole detection.

The second tool designed, landmark classifier, could automatically detect and classify poles and trees from the input point cloud. The tool could be extended to more classes easily by including more classes to the multi-class random forest classifier. Also, this would call for minor changes in the classification module of the developed tool. The performance of this tool could be improved by adding more training data. There was a limitation of training data availability during the research.

The novelties in the approaches undertaken are using DON in segmentation part and employing ESF and eigenvalue based features in the classification part. It was observed that DON needs to be optimized if the landmarks of interest changes. Also, experiments with landmark classifier tool gave the insight that better feature descriptors for clusters would help improve the precision and recall of the tool. So, more features could be used and compared to improve the results.

6 Acknowledgments

First of all, I would like to thank Prof. Wolfram Burgard who provided me an opportunity to do my master project at AIS. I would like to extend my sincere thanks to Tayyab Naseer, who have been guiding me throughout the project duration. His help and ideas were very vital in the success of this project.

Also, I would like to thank Philip Ruchti for helping me by sharing his random forest learner code as well as for providing me with a map of the campus for the test. Also, I would like to thank all my friends who have been supporting and encouraging me during the course of master project.

Bibliography

[1] D. Li and S. O. Elberink, "Optimizing detection of road furniture (pole-like objects) in mobile laser scanner data," *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci*, vol. 1, pp. 163–168, 2013.

[2] R. Dubé, D. Dugas, E. Stumm, J. Nieto, R. Siegwart, and C. Cadena, "Segmatch: Segment based loop-closure for 3d point clouds," *arXiv preprint arXiv:1609.07720*, 2016.

[3] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*, pp. 1–4, IEEE, 2011.

[4] H. Yokoyama, H. Date, S. Kanai, and H. Takeda, "Detection and classification of pole-like objects from mobile laser scanning data of urban environments," *International Journal of CAD/CAM*, vol. 13, no. 2, 2013.

[5] K. Ishikawa, F. Tonomura, Y. Amano, and T. Hashizume, "Recognition of road objects from 3d mobile mapping data," *International Journal of CAD/CAM*, vol. 13, no. 2, 2013.

[6] M. Lehtomäki, A. Jaakkola, J. Hyyppä, A. Kukko, and H. Kaartinen, "Detection of vertical pole-like objects in a road environment using vehicle-based laser scanning data," *Remote Sensing*, vol. 2, no. 3, pp. 641–664, 2010.

[7] J. Landa and V. Ondroušek, "Detection of pole-like objects from lidar data," *Procedia-Social and Behavioral Sciences*, vol. 220, pp. 226–235, 2016.

[8] Y. Ioannou, B. Taati, R. Harrap, and M. Greenspan, "Difference of normals as a multi-scale operator in unorganized point clouds," in *3D Imaging, Modeling, Processing, Visualization and Transmission (3DIMPVT), 2012 Second International Conference on*, pp. 501–508, IEEE, 2012.

[9] A. Aldoma, Z.-C. Marton, F. Tombari, W. Wohlkinger, C. Potthast, B. Zeisl, R. B. Rusu, S. Gedikli, and M. Vincze, "Tutorial: Point cloud library: Three-dimensional object recognition and 6 dof pose estimation," *IEEE Robotics & Automation Magazine*, vol. 19, no. 3, pp. 80–91, 2012.

[10] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3d object classification," in *Robotics and Biomimetics (ROBIO), 2011 IEEE International Conference on*, pp. 2987–2992, IEEE, 2011.