3D-PSH: Lightweight 3D LiDAR Object Detection Using Adaptive Clustering and 3D Point Spatial Histograms

Junaid Baber, Olivier Aycard

3D-PSH: Lightweight 3D LiDAR Object Detection Using Adaptive Clustering and 3D Point Spatial Histograms

Junaid Baber, Olivier Aycard
GIPSA-LAB, University of Grenoble Alpes, France
junaidbaber@ieee.org, aycardol@univ-grenoble-alpes.fr

***Abstract*—The advent of 3D LiDAR technology has revolutionized object detection in applications such as autonomous driving, robotics, and advanced driver assistance systems. However, existing methods often require substantial computational resources, limiting their practicality for real-time applications on devices with constrained hardware capabilities. This paper presents an efficient and lightweight 3D LiDAR object detection framework, 3D-PSH, that combines adaptive clustering with 3D Point Spatial Histograms (3D-PSH) and classical classification techniques to address these challenges.**

Our framework begins with an adaptive clustering algorithm that segments the point cloud data into distinct clusters, representing potential objects. 3D Point Spatial Histograms (3D-PSH) are then computed from these clusters and subsequently quantized into a Bag of Visual Words (BoVW) to create a compact and informative representation. These representations are then classified using robust classical classification methods to identify object types, such as pedestrians and vehicles. This multi-step approach ensures a balance between computational efficiency and detection accuracy, making it suitable for real-time deployment.

Extensive experiments on the KITTI dataset and our live sensor data demonstrate the effectiveness and efficiency of our proposed framework. The results indicate that our method achieves competitive accuracy while significantly reducing computational requirements compared to traditional approaches. This framework offers a practical solution for deploying 3D object detection in a wide range of applications, particularly where computational resources are limited.

I. INTRODUCTION

The advent of 3D LiDAR technology has revolutionized object detection in applications such as autonomous driving, robotics, and advanced driver-assistance systems (ADAS) [1]. However, existing methods often require substantial computational resources, limiting their practicality for real-time applications on devices with constrained hardware capabilities. Furthermore, the dependence on GPUs for processing large point cloud data can be a significant barrier for widespread deployment in resource-constrained environments.

Deep learning-based methods, such as PointNet [2], PointNet++ [3], and VoxelNet [4], have demonstrated remarkable performance in 3D object detection using point cloud data. PointNet introduced a novel architecture that directly consumes point clouds, while PointNet++ extended this approach by applying PointNet recursively to capture local features. VoxelNet represented the point cloud as a sparse 3D voxel grid, which was then processed by a 3D convolutional neural network. Although effective, these methods often rely on complex architectures and require substantial computational resources, limiting their applicability in resource-constrained environments. Other approaches, such as Frustum PointNets [5] and SECOND [6], have sought to improve efficiency by combining 2D image-based detection with 3D point cloud processing. Frustum PointNets first detected objects in 2D images and then applied PointNet on the corresponding 3D point cloud frustums. SECOND employed sparse 3D convolutional neural networks and utilized 2D object proposals to reduce computational complexity. While these methods offer improved efficiency, they still require significant computational resources and may not be suitable for real-time applications on resource-constrained devices. Moreover, adapting these deep learning-based approaches to detect new object types, such as mobile robots in indoor environments, tables, or chairs, can be complex and time-consuming. Retraining or fine-tuning these models often requires extensive labeled data and significant computational resources, limiting their flexibility and adaptability to new scenarios.

This paper presents an efficient and lightweight 3D LiDAR object detection framework, 3D-PSH, that combines adaptive clustering with 3D Point Spatial Histograms (3D-PSH). The Bag of Visual Words (BoVW) representation is also employed to make the object description more compact. Off-the-shelf classifiers such as Support Vector Machines (SVM), Decision Trees (DT), or Naive Bayes techniques make the framework easy to train for real-time applications. The visual flow of the proposed framework is shown in Figure 1.

Our framework begins with an adaptive clustering algorithm that segments the point cloud data into distinct clusters, representing potential objects. Keypoint descriptors such as SHOT [7], FPFH [8], BTC [9] or Link3D [10] can be extracted from these clusters and then quantized into a Bag of Visual Words (BoVW) to create a compact and informative representation. These representations are then classified using robust classical classification methods to identify object types, such as pedestrians and vehicles. This multi-step approach ensures a balance between computational efficiency and detection accuracy, making it suitable for real-time deployment.

The primary contributions of this work include: (1) an

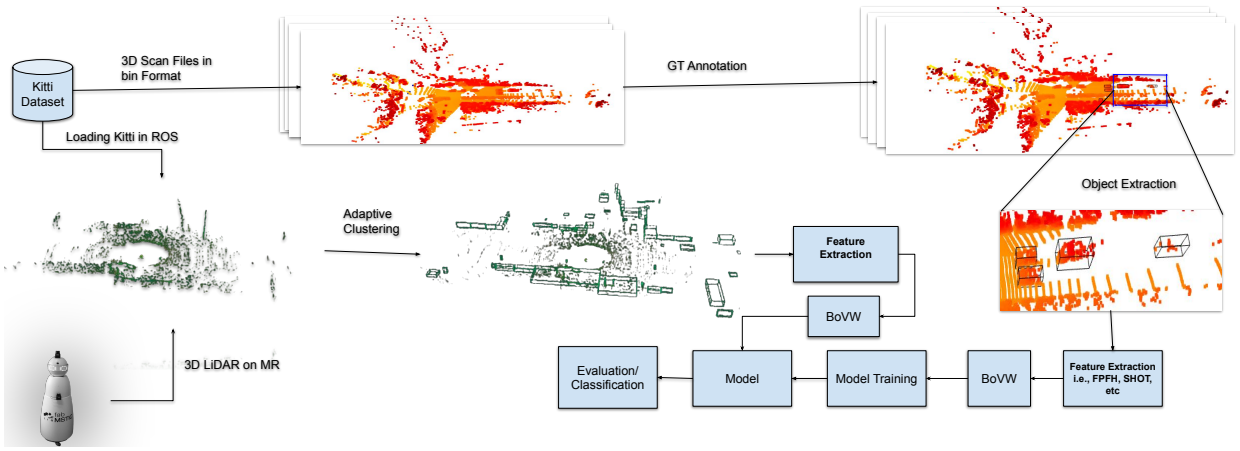

Fig. 1. Flow diagram of the proposed framework. The dataset, along with ground truth (GT) annotations, is utilized to extract objects and their corresponding labels. Features are computed for these objects and represented using the Bag of Visual Words (BoVW) approach. In an online scenario, the 3D LiDAR scan is processed by the clustering module, which identifies clusters treated as candidate objects. Features are then extracted from these clusters and passed to the classification model for final object identification.

adaptive clustering algorithm capable of dynamically adjusting to varying point densities within the point cloud, (2) the use of keypoint descriptors and Bag of Visual Words for efficient cluster representation, (3) the application of classical classification techniques that provide accurate object detection with minimal computational overhead, and (4) a framework designed for real-time feasibility on resource-limited devices.

Extensive experiments on the KITTI dataset and our live sensor data demonstrate the effectiveness and efficiency of our proposed framework. The results indicate that our method achieves competitive accuracy while significantly reducing computational requirements compared to traditional approaches. This framework offers a practical solution for deploying 3D object detection in a wide range of applications, particularly where computational resources are limited.

II. Related Work

3D object detection using LiDAR point cloud data has been an active area of research, with various approaches proposed in recent years. These methods can be broadly categorized into deep learning-based and classical machine learning-based techniques.

A. Deep Learning-based Approaches

Deep learning techniques have shown remarkable performance in 3D object detection tasks. PointNet [2] is one of the pioneering works that directly processed raw point cloud data using deep neural networks. It proposed a novel architecture that learned local and global features from unordered point sets. PointNet++ [3] extended this work by applying PointNet recursively to capture hierarchical features and improved the performance for complex scenes. VoxelNet [4] took a different approach by representing the point cloud as a sparse 3D voxel grid, which was then processed by a 3D convolutional neural network. This method achieved impressive results on various benchmarks but required substantial computational resources. Frustum PointNets [5] combined 2D image-based object proposals with 3D point cloud processing. It first detected objects in 2D images and then applied PointNet on the corresponding 3D point cloud frustums to obtain refined 3D bounding boxes. SECOND [6] employed sparse 3D convolutional neural networks and utilized 2D object proposals to reduce computational complexity. It demonstrated a good trade-off between accuracy and efficiency compared to other methods. While deep learning-based approaches have shown excellent performance, they often require extensive computational resources, specialized hardware (such as GPUs), and large amounts of labeled data for training. Additionally, adapting these models to detect new object types can be challenging and time-consuming.

B. Classical Machine Learning-based Approaches

Several works have explored classical machine learning techniques for 3D object detection using LiDAR data. These methods typically involve handcrafted feature extraction and classification algorithms. Wang et al. [11] proposed a method that combined clustering and support vector machines (SVMs) for pedestrian detection. They extracted geometric features from the clusters and used SVMs for classification. Xiao et al. [12] developed a two-stage approach, where they first generated object proposals using a sliding window approach and then classified these proposals using a random forest classifier. They explored various handcrafted features, including geometric and statistical features, for their classifier. Zhao et al. [13] utilized a divide-and-conquer approach, where they first segmented the point cloud into ground and non-ground regions. They then employed a density-based clustering algorithm to identify object candidates and classified them using a random forest classifier. Classical machine learning-based approaches tend to be more computationally efficient and easier to interpret than deep learning methods. However, they often rely on handcrafted feature engineering and may not generalize

as well as deep learning techniques for complex scenarios. Our proposed framework aims to combine the advantages of both deep learning and classical machine learning approaches. By leveraging adaptive clustering and simple classifiers, we strive to achieve a balance between computational efficiency and detection accuracy, while maintaining the flexibility to adapt to new object types with minimal effort.

III. Framework Description

Our framework processes 3D LiDAR point cloud data to achieve efficient object detection. The processing pipeline consists of four main stages: Adaptive Clustering, 3D-Point Spatial Histograms (3D-PSH) computation, Bag of Visual Words (BoVW) model representation, and Classifier.

A. Adaptive Clustering

Using a fixed threshold for clustering can be highly sensitive to the chosen value, leading to either over-segmentation or under-segmentation of clusters. If the threshold is too small, a single object might be split into multiple clusters, whereas if it is too large, multiple objects could be incorrectly merged into one cluster. This sensitivity makes fixed threshold-based clustering less reliable, especially in dynamic environments where distances between the sensor and objects vary significantly. To overcome these issues, adaptive threshold-based clustering is employed [14].

The adaptive clustering algorithm for 3D LiDAR point clouds begins with the removal of the ground plane to filter out floor points. This is achieved by retaining only the points above a certain height threshold, z_{min} [14]. The remaining points, represented as $P \in \mathbb{R}^{n\times 3}$, are then subjected to clustering based on their Euclidean distance in 3D space. To handle varying distances efficiently, the clustering threshold, d^*, is made adaptive. This threshold is calculated using the scan range r and the vertical angular resolution ϕ of the LiDAR sensor, formulated as $d^* = 2r\tan(\phi/2)$.

To optimize computational efficiency and improve robustness, the space around the sensor is divided into nested circular regions, each applying different distance thresholds. These regions are defined at fixed distance intervals Δd, and the maximum cluster detection range r_i for each threshold is computed and rounded down to form the radius of the circular area. Clusters, defined as $C = \{C_j\}$, are extracted based on the minimum Euclidean distance between points, ensuring no overlap, and filtered using human-like volumetric constraints. Each cluster C_j contains n_j points and can be represented as $C_j \in \mathbb{R}^{n_j\times 3}$.

B. 3D-Point Spatial Histograms (3D-PSH)

For each identified cluster C_j, we compute the 3D-Point Spatial Histograms (3D-PSH) for each point in the cluster. The 3D-PSH feature computation captures the local spatial geometry around each point. The pseudo step for 3D-PSH are shown in Algorithm 1. The algorithm starts by normalizing the point cloud to ensure consistency. Normals are computed for each point in the cloud, which are essential for capturing the spatial orientation of points relative to each other. These normals are initially generated randomly but could be derived more precisely in a real-world application, similar to SHOT [7].

Algorithm 1 3D-PSH (3D-Point Spatial Histograms)

Function COMPUTE_NORMALS(cloud):
    Initialize normals list of size cloud.points.size()
    Generate random normals for each point
    return normals
Function NORMALIZE_PATCH(cloud):
    Normalize the point cloud as required
    return normalized cloud
Function COMPUTE_PSH(cloud, k):
    cloud = NORMALIZE_PATCH(cloud)
    normals = COMPUTE_NORMALS(cloud)
    Convert cloud.points to points_matrix
    Initialize KD-Tree with points_matrix
    Set histogram parameters: radial_bins b, azimuth_bins α, elevation_bins σ
    Initialize histograms to zero

for each point i in cloud **do**
  Find k nearest neighbors
  Compute max_radial_distance, radial_unit, azimuth_intervals, elevation_intervals
  for each neighbor j (excluding point itself) **do**
    Compute relative_point, r, azimuth, elevation
    Compute radial_bin, azimuth_bin, elevation_bin
    Increment corresponding bin in histograms
  end for
end for
Normalize each histogram row
return histograms

Next, the normalized points are organized into a KD-Tree to facilitate efficient nearest neighbor searches. For each point in the cloud, the algorithm identifies the k nearest neighbors, the default value is kept 20. The relative positions of these neighbors are used to compute spatial attributes such as radial distance, azimuth, and elevation. These attributes are then binned into histograms based on predefined parameters: radial bins b, azimuth bins α, and elevation bins σ. Each point's contribution to the histogram is determined by which bins their spatial attributes fall into.

The algorithm iterates over each point and its neighbors, incrementing the appropriate histogram bins. Once all points have been processed, each histogram row is normalized to ensure uniformity. The resulting histograms represent the spatial distribution of points within the cluster, encapsulating the local geometric structure. These histograms serve as robust descriptors for subsequent stages in the processing pipeline. The default values for $b = 2, \quad \alpha = 8, \quad \sigma = 2$ which makes 3D-PSH descriptor of dimension $2 \times 8 \times 2 = 32$. Thus, the given $C_j \in \mathbb{R}^{n_j\times 3}$ is represented by 3D-PSH $F_j \in \mathbb{R}^{n_j\times 32}$.

The 3D-PSH is inspired by SHOT and uses similar steps for computing the histograms. The SHOT descriptors provide a detailed and accurate representation of local 3D geometry by capturing the orientation of the points. Those are highly suitable for tasks requiring precise shape and structure information, such as object recognition and complex scene understanding. However, they are computationally intensive due to the need for accurate normal estimation and the high dimensionality of the descriptors. Whereas, 3D-PSH offer a simpler and more computationally efficient approach to describing 3D point clouds. They capture the general distribution of points in space, making them suitable for applications where computational resources are limited or where precise geometric details are less critical which makes 3D-PSH method ideal for real-time object detection and scenarios requiring quick processing.

C. Bag of Visual Words (BoVW) Model Representation

The next step involves quantizing the 3D-PSH features into a bag of words (BoVW) model. Each 3D-PSH feature is mapped to the nearest visual word in a pre-defined vocabulary, constructed using k-means clustering on a large set of training 3D-PSH features. The result is a histogram representation for each cluster, where each bin in the histogram corresponds to a visual word from the vocabulary. This BoVW model effectively summarizes the 3D-PSH features within each cluster, facilitating further processing and analysis.

Bag of Visual Words (BoVW) model is widely used for feature quantization. Each cluster descriptor, $F_j \in \mathbb{R}^{n_j \times 32}$, is quantized into a finite number of centroids from 1 to M, where M denotes the total number of centroids, also known as visual words, denoted by $V = \{V_1, V_2, \ldots, V_M\}$ and each $V_m \in \mathbb{R}^{32}$. Let a frame P be represented by some local cluster descriptors $F = \{F_1, F_2, \ldots, F_m\}$, where $F_j \in \mathbb{R}^{n_j \times 32}$. In the BoVW model, a function G is defined as $G : \mathbb{R}^{32} \rightarrow [1, M]$ and $F_{j_i} \rightarrow G(F_{j_i})$, G maps descriptor $F_{j_i} \in \mathbb{R}^{32}$ to an integer index. For a given frame P, the bag of visual words, $I = \{I_1, I_2, \ldots, I_M\}$, is computed. I_m indicates the number of times V_m appeared in frame P, and I is unit normalized at the end. Generally, k-means or hierarchical k-means clustering is applied to obtain the centroids (visual words), V. The value of M is kept very large for applications such as image matching or retrieval; the suggested value of M is 1 million. The accuracy of quantization mainly depends on the value of M; if the value is small, then two different key point descriptors will be quantized to the same visual words, which will decrease the distinctiveness. Conversely, if the value is very large, then two similar key point descriptors that are slightly distorted can be assigned different visual words, which will decrease the robustness.

D. Classifier Training

Finally, the histogram representations obtained from the BoVW model are used to train a classifier. A Support Vector Machine (SVM) or any other suitable classifier can be trained on these histograms to classify the given cluster. The classifier is trained using labeled data, where each histogram is associated with a label indicating whether the cluster represents a human or not. This trained classifier can then be used to classify new clusters in real-time.

The overall process can be summarized as follows:

1) Represent the point cloud data as $P \in \mathbb{R}^{n \times 3}$.
2) Apply adaptive clustering to P to obtain clusters $C = \{C_j \mid C_j \in \mathbb{R}^{n_j \times 3}\}$.
3) For each cluster C_j, compute the 3D-PSH features F_j.
4) Quantize the 3D-PSH features F_j into a bag of words model to obtain a histogram representation for each cluster.
5) Train a classifier on the histogram representations to distinguish between human and non-human clusters.

This pipeline ensures efficient and accurate processing of 3D LiDAR data for human detection and tracking, leveraging adaptive clustering, robust feature descriptors, and effective classification.

IV. Experiments and Results

To evaluate the performance of proposed framework and the descriptor, we train the SVM classifier on Kitti dataset [15]. The KITTI dataset is a widely used benchmark in the field of autonomous driving and contains various types of data, including point clouds, images, and ground truth annotations. For our experiments, we focus on the 3D point cloud data and the corresponding ground truth annotations for object detection. The ground truth annotations in the KITTI dataset provide detailed information about the objects present in each frame, including their class labels and 3D bounding boxes. We extracted the instances of different classes from these annotations to create our training dataset, as illustrated in Figure 1.

TABLE I
Classification Accuracy of SVM Model on KITTI Dataset Extracted by Ground Truth Annotation

Class Type	3D-PSH	SHOT	FPFH	LINK3D
Car	96.5%	97.4%	89.4%	88.7%
Pedestrian	89.5%	90.0%	85.7%	81.8%
Cyclist	91.0%	91.5%	86.5%	84.9%
Person Sitting	83.0%	83.5%	80.5%	80.9%
Overall	95.2%	96.0%	88.9%	87.6%

For each instance, we extracted the 3D Point Spatial Histograms (3D-PSH) as described in the previous sections. These histograms serve as the feature descriptors for our SVM model. The 3D-PSH descriptors are then quantized into a Bag of Visual Words (BoVW) model to create a compact representation of the features.

In the BoVW model, we experimented with the number of visual words, varying the total number of centroids from 500 to 1500. We found that setting the total number of visual words to 1000 provided a good trade-off between the dimensionality of the feature vectors and the computational speed. This setting ensures that the feature descriptors are

compact enough for efficient processing while maintaining sufficient distinctiveness for accurate object classification.

Using the extracted and quantized features, we trained a Support Vector Machine (SVM) model to classify the objects into their respective classes. The SVM model was chosen for its robustness and ability to handle high-dimensional feature spaces effectively. The training process began with splitting the dataset into training and validation sets, typically with an 80% training and 20% validation split. Next, the feature vectors were normalized to ensure that all features contributed equally to the model training. We then trained the SVM model using the training set, employing the Radial Basis Function (RBF) kernel, which is well-suited for non-linear classification problems. The training accuracy of the model on objects extracted from the KITTI dataset is summarized in Table I.

In the overall evaluation of the framework, the point cloud data is processed through the entire system. The framework begins by performing adaptive clustering to segment the point cloud into clusters, each representing a potential object. Passing all clusters to the classifier would be computationally expensive, so we filter and reduce the number of clusters to be processed. Clusters that are too large, too small, or have an aspect ratio that suggests they belong to non-object areas such as roads or walls (in indoor environments) are filtered out based on predefined criteria. Specifically, the number of points in a cluster should be greater than 85 and less than 1500, and the aspect ratio should be less than 3 m. The remaining clusters, which are likely to be valid objects, are then passed through the classification model.

The accuracy of the overall framework on the KITTI dataset, compared with state-of-the-art frameworks, is summarized in Table II. Additionally, the framework was evaluated using a live sensor setup with a 3D-LiDAR Ouster mounted on a mobile robot in an indoor environment. This evaluation also demonstrated good accuracy, showcasing the framework's effectiveness in real-world applications.

TABLE II
PERFORMANCE OF VARIOUS MODELS ON CAR, PEDESTRIAN, AND CYCLIST DETECTION METRICS (AP).

Model	Car	Pedestrian	Cyclist
PointPillar	77.28	52.29	62.68
SECOND	78.62	52.98	67.15
SECOND-IoU	79.09	55.74	71.31
PointRCNN	78.70	54.41	72.11
PointRCNN-IoU	78.75	58.32	71.34
Part-A2-Free	78.72	65.99	74.29
Part-A2-Anchor	79.40	60.05	69.90
PV-RCNN	83.61	57.90	70.47
3D-PSH	75.45	58.11	69.54
3D-PSH + SHOT	76.11	58.72	70.34

V. CONCLUSION

The results of our experiments demonstrate the effectiveness of the proposed 3D-PSH and BoVW-based approach for 3D LiDAR object detection. The SVM model trained on the KITTI dataset achieves high accuracy across multiple classes while maintaining computational efficiency. The overall framework, which includes adaptive clustering and predefined filtering criteria, also shows competitive performance.

Our proposed framework is effective for real-time applications and can be deployed on live sensors without the need for GPU-based processing. Despite the absence of GPU requirements, the framework still delivers competitive accuracy, making it a practical solution for 3D object detection in various resource-constrained environments. This makes the framework suitable for a wide range of applications.

ACKNOWLEDGEMENT

This work is supported by the French National Research Agency in the framework of the "Investissements d'avenir" program (ANR-15-IDEX-02).

REFERENCES

[1] Z. Fan, H. Liu, J. He, S. Jiang, and X. Du, "Pointfpn: A frustum-based feature pyramid network for 3d object detection," in *2020 ICTAI*, pp. 1129–1136, 2020.

[2] 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 (CVPR)*, pp. 652–660, 2017.

[3] 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 (NeurIPS)*, vol. 30, 2017.

[4] Y. Zhou and O. Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4490–4499, 2018.

[5] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 918–927, 2018.

[6] Y. Yan, Y. Mao, and B. Li, "Second: Sparsely embedded convolutional detection," *Sensors*, vol. 18, no. 10, p. 3337, 2018.

[7] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *Computer Vision – ECCV 2010*, pp. 356–369, 2010.

[8] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *2009 IEEE International Conference on Robotics and Automation*, pp. 3212–3217, 2009.

[9] C. Yuan, J. Lin, Z. Liu, H. Wei, X. Hong, and F. Zhang, "Btc: A binary and triangle combined descriptor for 3-d place recognition," *IEEE Transactions on Robotics*, vol. 40, pp. 1580–1599, 2024.

[10] Y. Cui, Y. Zhang, J. Dong, H. Sun, X. Chen, and F. Zhu, "Link3d: Linear keypoints representation for 3d lidar point cloud," *IEEE Robotics and Automation Letters*, vol. 9, no. 3, pp. 2128–2135, 2024.

[11] D. Z. Wang, I. Posner, and P. Newman, "What could move? finding cars, pedestrians and bicyclists in 3d laser data," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 4038–4044, IEEE, 2012.

[12] W. Xiao, B. Vallet, K. Schindler, and N. Paparoditis, "Street-side vehicle detection, pose estimation and tracking using multi-view square cuboid intersection," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 11, pp. 3069–3079, 2016.

[13] B. Zhao, X. Nie, W. Chen, and X. Chao, "An efficient 3d object detection and pose estimation framework for lidar-based systems," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1338–1345, IEEE, 2018.

[14] Z. Yan, T. Duckett, and N. Bellotto, "Online learning for 3d lidar-based human detection: Experimental analysis of point cloud clustering and classification methods," *Autonomous Robots*, 2019.

[15] A. Geiger, P. Lenz, and R. Urtasun, "Are we ready for autonomous driving? the kitti vision benchmark suite," in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 3354–3361, 2012.