End-to-end 3D object detection with Machine Learning

Saavedra, Miguel. Salazar, Gustavo

{*miguel.saavedra gustavo.salazar*} @uao.edu.co

Universidad Autónoma de Occidente

***Abstract*—3D object detection is a problem that has gained popularity among the research community due to its extensive set of application on autonomous navigation, surveillance and pick-and-place. Most of the solutions proposed in the state-of-the-art are based on deep learning techniques and present astonishing results in terms of accuracy. Nevertheless, a set of problems inherits from this sort of solutions such as the need of enormous tagged datasets, extensive computational resources due to the complexity of the model and most of the time, no real-time inference. In this paper we propose an end-to-end classic Machine Learning (ML) pipeline to solve the 3D object detection problem within cars. Our proposed method is leveraged on the use of frustum region proposals to segment and estimate the parameters of the amodal 3D bouning box. Here we do not deal with the problem of 2D object detection as for most of the research community this is considered solved with Convolutional Neural Networks (CNN). In the results section is shown how the use of classical ML techniques can considerably improve the inference speed of such systems and allow the deployment for real-world applications.**

***Index Terms*—3D object detection, Self-driving vehicles, Machine Learning, Frustum, Instance segmentation, Regression.**

I. INTRODUCTION

OVer the past years object detection has become a crucial element in every robotics-vision pipeline. As an example 2D object detection has been extensively used to detect object in a scene [1] and obtain a sense of environmental awareness. Some of the applications where these techniques have been used are video surveillance [2], License Plate Recognition (LPR) [3], robotic arms pick-and place [4] and autonomous navigation of self-driving vehicles [5]. Notwithstanding the benefits of 2D object detection techniques, these do not provide information regarding the pose of the object in the physic space.

Considerable research efforts have been conducted to the creation of 3D object detection methodologies where the agent has information of the current pose of the interest object. Consider a pick and place task where the final actuator knows the relative pose of an interest object, this knowledge allows improved accuracy over the pick-and place task [4]. In a similar fashion, 3D object detection has been extensively used with autonomous vehicles to obtain the relative poses of the objects around the vehicle for safer decision-making during the navigation [6].

Most of the applications mentioned before can be easily deployed in real-world settings as a business model and monetize. Accurate Pick-and-place can be used to improve the efficiency in warehouses, self-driving vehicles could automate the transportation systems and automatic surveillance are a feasible alternative to avoid repetitive tasks for human beings. Modern Deep Learning (DL) methods such as the ones based on frustum proposals [7] have demonstrate astonishing results leveraging the use of 2D object detection technique with LiDAR point clouds to estimate the 3D bounding box parameters. Nevertheless, the current state-of-the-art with DL for 3D object detection as presented in [7]–[9], requires considerable computational resources which are not always available. Furthermore, another problem is the need of substantial human

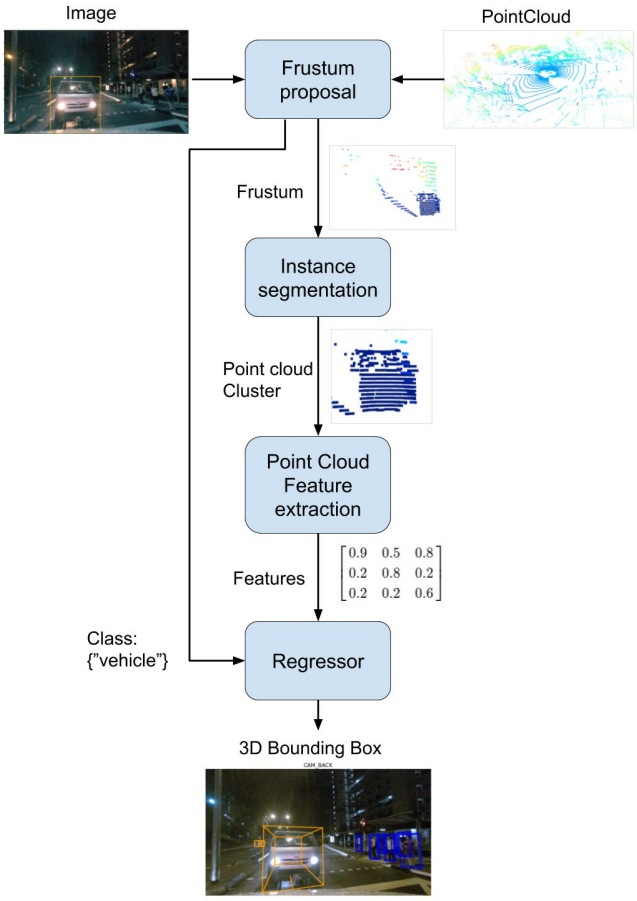


Fig. 1: 3D object detection pipeline.

efforts to tag the datasets needed to train the DL model.

To address these issues we propose the development of an end-to-end Machine Learning (ML) pipeline for 3D object detection as shown in the diagram presented in Fig. 1. Our proposal relies on the use of classical ML techniques as these have proved high efficiency working with limited data samples and real-time inference with restricted computational resources. For this ML solution, we leverage our 3D detection system on the use of frustum region proposals [7] as these have shown remarkable results in terms of accuracy. Moreover, as the 2D object detection problem is considered solve in the research community using Convolutional Neural Networks (CNN) [10], therefore, we will only address the instance segmentation, feature extraction and parameters regression problems.

Our proposed pipeline will works as follows. First a frustum region proposals is extracted mapping the LiDAR point cloud to the image frame and cropping only the points within the interest 2D bounding box. Then, an unsupervised learning technique like [11] is employed to cluster the points of interest and remove outliers. Afterwards, the segmented point cloud is used to create a feature representation which is finally used in a supervised learning regressor such as [12] to predict the amodal 3D bounding box parameters.

The main contributions of this work is a complete ML system to solve the 3D object detection task using frustum region proposals. For the known of the authors, not related work has been presented until today where ML techniques have been employed to address this task. Finally, the code and results will be release in an open source repository in Github*.

II. Related work

The 3D object detection task is constantly addressed with the use of deep neural networks. For instance there are two big family of solutions employed to solve it: birds-eye-view [13] and frustum proposals [7]. The first one uses a top-view of the lidar point clouds to detect the objects of interest and predict its 3D parameters in the scene. This sort of architectures can be seen in [14] where the authors proposed Pixor, a neural network architecture that is capable to predict the amodal 3D bounding box parameters of the objects within the scene.

Despite birds-eye-view (BEV) might be seen as a feasible approach to address the 3D object detection task, this is often hindered by the fact that small objects are not properly detected with this sort of technique. Furthermore, as this is a deep learning based approach, the complexity of the model tends to be high and thus, the inference in real-time is limited to around 5 - 10 FPS even with strong computational resources [7].

Based on the last drawbacks the frustum proposals methods for 3D object detection hatched. This technique leverages the 3D detection with the use of 2D detection in the image plane [15]. The idea was introduced in [7] where the authors modified the PointNet architecture given in [9] to estimate the 3D bounding box parameters. This neural network architecture works by creating a frustum region proposal out of the 2D detection. Then, the point cloud within the frustum is segmented and used in a multi-layer perceptron to predict the bounding box parameters.

Since the release of this methodology, several works have tried to improve the architecture as shown in [15]–[17]. However, the principal drawbacks of this technique are the need of substantial computational resources and tagged datasets. Therefore, we aim to propose an end-to-end 3D object detection pipeline based on region proposals and classic machine learning techniques to improve the inference time of the task at hand.

For the known of the authors, there is no similar work with ML techniques for 3D object detection. Therefore, we will employ similar approaches as the ones presented in [18] where the authors used the DBSCAN algorithm with automatic epsilon estimation to compute the interest cluster of interest out of the frustum proposal. Similarly, as we are not using a neural networks to extract a feature representation of the segmented point cloud, we will employ point cloud global descriptors as this are capable to encode object geometry. Some examples of global descriptors are the Viewpoint Feature Histogram (VFH) [19] or The Ensemble of Shape Functions (ESF) [20] which are widely used for object detection and classification tasks with point clouds.

Finally, a regression methodology such as [12] will be utilized to predict the amodal 3D bounding box parameters. This work does not aim to overcome the current state-of-the-art for 3D object detection but to present a different approach to solve the task with real time inference employing classical machine learning techniques.

III. Nuscenes Dataset

In a first approach KITTI [22] and nuScenes [21] datasets are compared as options to address this problem. However, the nuScenes dataset is preferred over KITTI as it is the first large-scale dataset to provide information from an extended sensor suit with high-data quality of an autonomous vehicle. There are about 14 sensors (6 Cameras, 1 LiDAR, 5 RADAR, GPS, IMU) which take information from the environment, condensing a wide representation of the surrounding world (front, back and sides) as shown in Fig. 2.

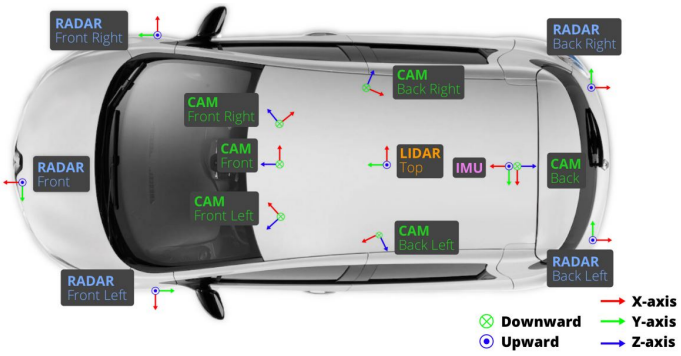
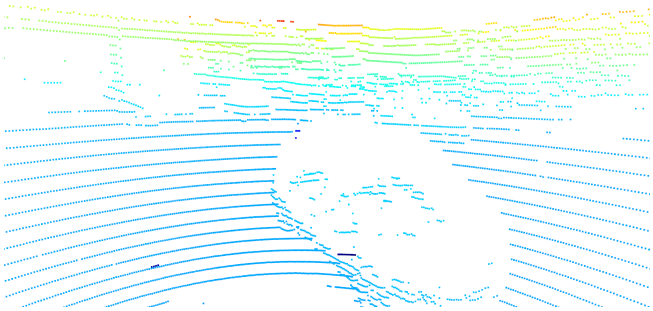


Fig. 2: nuScenes sensor set up [21].

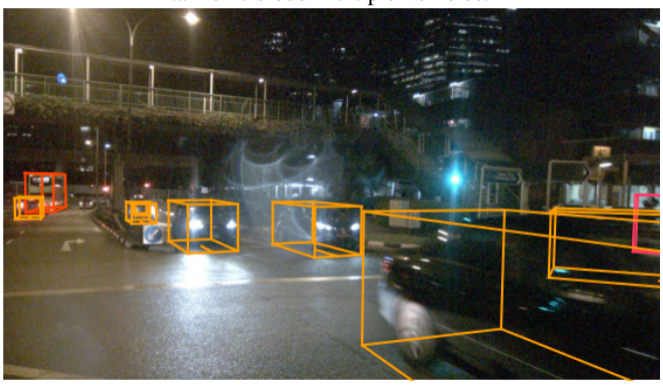
NuScenes collects approximately 15 hours of driving data in Boston and Singapore, that sums almost 1000 scenes of

*https://github.com/MikeS96/3d_obj_detection

20 seconds each, and a sampling frequency of 2 Hz in every scene. NuScenes has a subset of its full dataset called “Mini” which is used in this work, it contains 10 full scenes with its annotations for all kind of classes from pedestrians to traffic cones. For the scope of the project it is only used the ”vehicle” category as shown in Fig. 3(a-b).



a: Point cloud multiple vehicles.



b: Camera vehicles annotations.

Fig. 3: Point cloud and camera ”vehicle” annotation from sample.(a)Multiple vehicles in point cloud. (b)Multiple vehicles annotations.

As the 2D object detection problem is considered solved using CNN, the process of creating the point cloud dataset for instance segmentation starts with all samples from each scene. In every sample the annotations are filtered by “vehicle” category and a minimum of 100 points by instance. The information associated with the vehicles in the current sample is stored in JSON format with data such as annotation, sample and camera tokens, frustum point cloud path, also orientation, width, length, height bounding box values and X, Y, Z centroid coordinates.

Frustum point clouds are saved as an array in an individual TXT file, that is used in the clustering process for instance segmentation. In the JSON file the bounding box values are used as targets for the regression task.

IV. Problem formulation

The problem presented in this paper is the development of a 3D object detector pipeline for self-driving vehicles’ applications using classic machine learning techniques. For the sake of simplicity, we will only address the detection of other vehicles. This problem is divided in two fundamental machine learning tasks: clustering and regression.

Clustering is an unsupervised learning task where the objective is to obtain a group of clusters based on input data without labels. This methodology is employed to segment the instance of interest from a given point cloud in an image. The idea of this unsupervised learning approach is to remove outliers and noise that does not belong to the point cloud of the interest vehicle.

In a similar fashion, there is a supervised learning component where the amodal 3D bounding box parameters must be estimated for a given segmented point cloud. The interest parameters which describes a 3D geometry in the space are given in (1). There, x, y, z correspond to the centroid coordinate with respect to the LiDAR frame, l, h, w are the length, height and width of the bounding box; respectively. Finally, θ is the orientation in yaw of the bounding box with respect to the LiDAR coordinate frame.

$$\boldsymbol{X} = [x, y, z, l, h, w, \theta]^T \tag{1}$$

The use of supervised learning strategies for regression need a set of samples or training data which will be provided for each car in a scene. The goal is to exploit the information provided by the unsupervised learning model to feed a supervised regressor architecture and estimate the bounding box of a given vehicle.

V. Frustrum generation and 3D object detection

The following section introduces the techniques used to obtain a 3D detection of a vehicle using an image-LiDAR based approach. First, we will explain how we use data preprocessing to create the frustum region proposal. Subsequently, an unsupervised technique (clustering) is used to filter noise out of the frustum. Finally, a global-feature extractor is used to obtain a feature representation from the point cloud and this is feed into a supervised regressor to predict the bounding box parameters.

A. Data pre-processing for Frustum region proposal

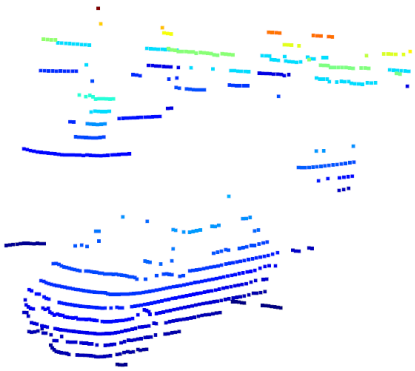
To obtain a frustum region proposal, several pre-processing steps must be followed. First of all, as the NuScenes dataset does not provide 2D labels for object detection, 3D labels must be mapped into 2D. Naturally, the 3D instances are measured with respect to the world coordinate frame, therefore, each one of these is mapped with transformations between different frames to one of the 6 cameras which took the instance’s photo.

Furthermore, as a 3D bounding box contains eight points to describe the geometry and a 2D bounding box only four points, an additional pre-processing step is carried out where the points that does not match a rectangular geometry in an image are discarded. An example of a 2D bounding box obtained after this process can be seen in 4(a)

Once the 2D bounding box is obtained, the LiDAR point cloud is mapped to the interest image frame. This step is needed as we need to identify the points laying within the

a: 2D detection.

b: Point cloud from 2D detection.



c: Frustum obtained

Fig. 4: Frustum region proposal selection process. (a) a 2D bounding box is obtained in the image frame; (b) the LiDAR point cloud is mapped to the image frame and the points outside the bonding box are removed. (c) a frustum region proposal is obtained out of the 2D boudning box and LiDAR point cloud.

detection. To accomplish this, intrinsic and extrinsic transformation matrices provided by the NuScenes dataset are used. Despite that this step might seems sufficient to generate the frustum, and additional algorithm had to be used to create an accurate frustum as the one shown in 4(b).

If the point cloud is cropped as-it-is, there will be a lot of outliers which correspond to the road. Many of this road points would be confused as part of the interest object during a clustering step as they are very close to the wheels. Hence, the RANSAC algorithm [23] was used to obtain a plane model that contains the road points. Fig. 5 shows how the plane obtained is capable to filter the road points (colored as red) from the point cloud. This step guarantees more reliable results once the point cloud is cropped with respect to the 2D bounding box. The final result after processing the point cloud and cropping it can be seen in Fig. 4(b). The Pseudo-code for the RANSAC algorithm for road plane estimation is given in 1.

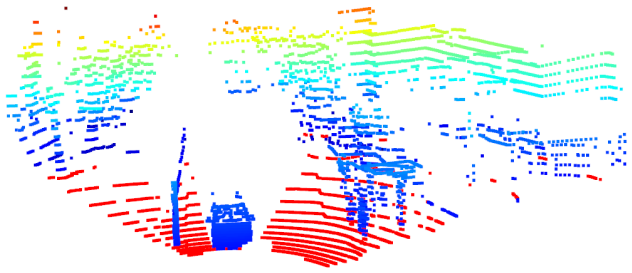


Fig. 5: Road segmentation in a LiDAR point cloud using RANSAC for plane estimation. The red points belong to the road based on the plane estimated with RANSAC.

Algorithm 1 RANSAC for plane estimation

Initialization:
  Given a plane model in the form $ax + by + xz + d = 0$, find the smallest number of samples, M, from which the model can be computed
for each iteration **do**
  Randomly select M points from the frustum
  Compute model parameters a, b, c, d using the selected M samples
  Check how many samples from the rest of the data actually fits the model (Inliers C).
  if $C >$ inlier ratio threshold or maximum iteration reached **then**
    End and return best inlier **C** set
  else
    Iterate again
  end if
end for
Recompute model parameters a, b, c, d from entire best inlier set

Once the points within the 2D bounding box are found, the final step is to obtain these points not in the image frame but the LiDAR reference frame. The indexes of the interest points are saved and retrieved from the original point cloud. The final frustum region proposal is the point cloud presented in Fig. 4(c). Notwithstanding that the frustum is obtained, there are outliers that need to be removed out of the frustum to obtain the points of the interest instance. This step is called instance segmentation and is accomplished by using unsupervised learning (clustering).

B. Clustering

The clustering step is fundamental to accurately estimate the 3D bounding box parameters as noisy points might drastically affect the performance of a regressor. Understand noise as the points that do not belong to the car instance. An example of noise are the points behind the vehicle in Fig. 4(c).

In this project we assess three different clustering algorithms: DBSCAN, K-means and Agglomerative Clustering. Each one of these unsupervised learning algorithms has its advantages and pitfalls. For instance, K-means is a clusters-based algorithm whereas the other two are agglomerative-based. The principal advantage of DBSCAN and Agglomerative clustering is that not prior number of clusters are given to the algorithm to work. They will start automatically generating clusters and stop once a criteria is met.

As there are no class labels for this clustering task, we could use metrics such as Silhouette Coefficient [24] where no class labels are needed to assess the performance of the clustering algorithm. However, we will select the best clustering algorithm based on the visual results presented over different samples. The metric evaluation part will be presented with the supervised learning regressor.

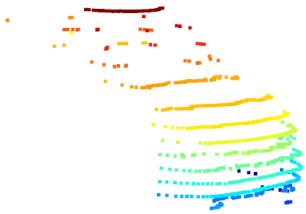


Fig. 6: Segmentation of a car out of a given frustum.

As the output of these unsupervised learning algorithms are a label for each point in the cloud, we will automatically obtain the interest points belonging to the vehicle as the cluster with more points within in. This assumption is based on the fact that the interest object should have the greater amount of points hitting it. An example of a segmented point cloud of a car can be seen in Fig. 6.

Once the segmented point cloud is obtained, it is ready for an additional step which is the feature extraction through global descriptors.

C. Regression

The final step to estimate the 3D bounding box starts with the point cloud instance segmented by the clustering process, this cloud contains all the spatial, orientation and volumetric information needed for the regression work.

To extract the volumetric features we use The Ensemble of Shape Functions (ESF), which was introduced in [20] and is a descriptor of ten 64-bin sized histograms with a total of 640 features that are part of the input set for the regression model. Addressing the spatial characteristics we use mean, median and middle statistics operations for each axis of the segmented point cloud to find a related centroid, this allows the model to detect a relation between the object size and its centre.

The orientation features are quite more difficult to obtain due to the lack of knowledge between the point cloud and the real front of the vehicle. In order to find the best way to address this problem we use Principal Components Analysis (PCA) in which we can find the direction of maximum variance (Component 1). Computing dot product between an unitary vector and the first component we manage to find Cos and Sin associated values that are used in as an orientation approximation input data for the regression model. However, this still does not tell us the real orientation (front or back) of the vehicle.

This previous processed features plus targets are collected, organized and saved in a CSV file for an easy use in the training task. The information is taken from the file and split in 80 % for Training and 20 % for Test, then it is transformed using MinMax and Standard scaler to compare, this data conversion help us normalize the features data for an easier model training.

This bring us to the model selection, we start to compare architectures as Linear model with Lasso, Support Vector Regression (SVR) and Random Forest using Grid search changing different values to find the best Hyperparameters combination for each one. Then we contrast Training and Test prediction scores between all to finally choose the model that can generalize better the 3D bounding boxes as seen in Fig. 7.

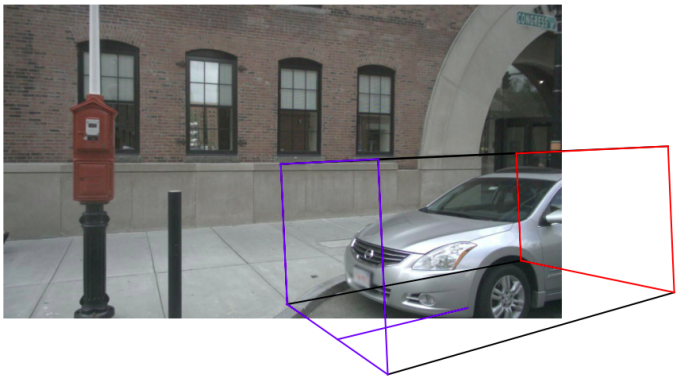
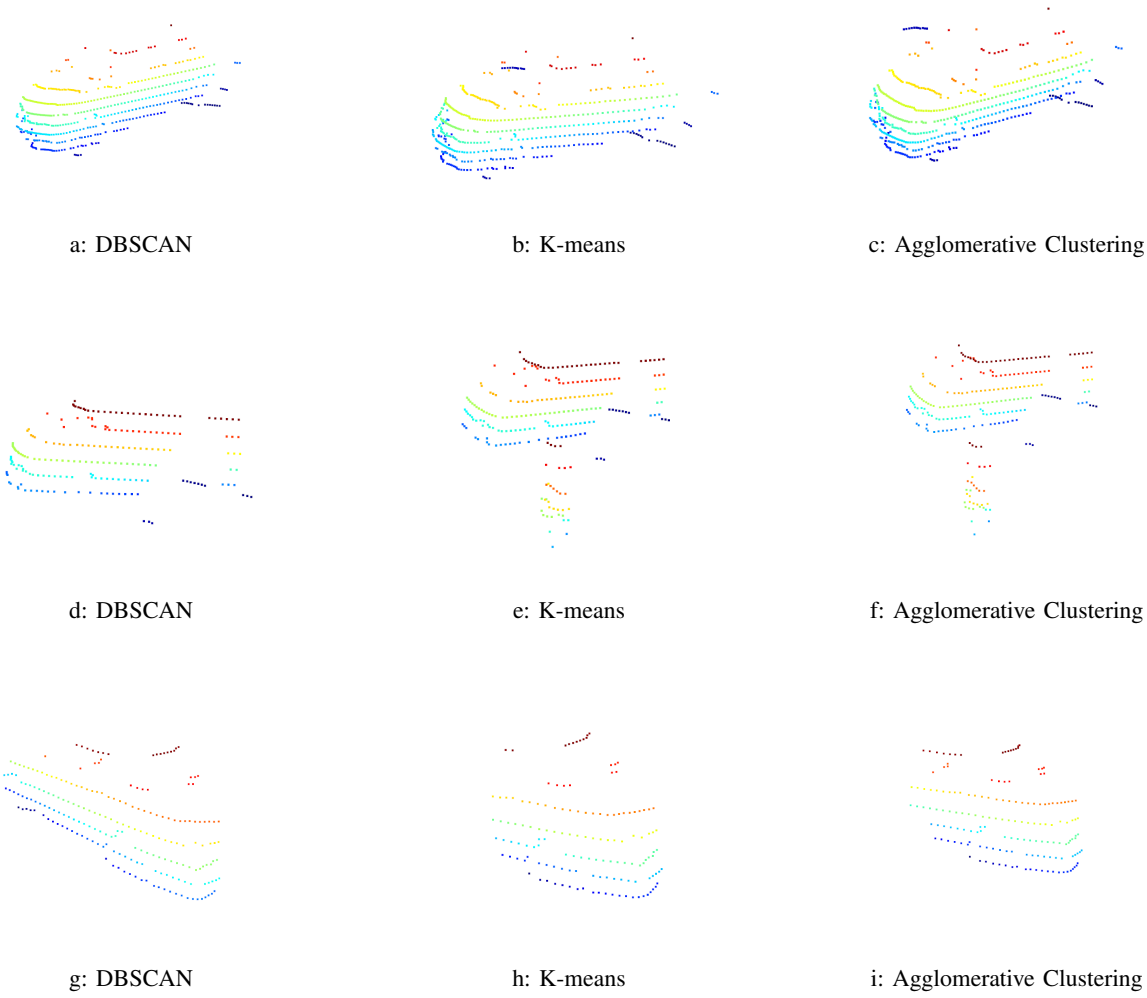


Fig. 7: 3D Bounding box prediction.

VI. Evaluation

This section presents the results obtained for the instance segmentation stage and prediction of the 3D bounding box parameters. As mentioned before, the clustering algorithms are visually assessed. In a similar fashion, the regressor is evaluated through the accuracy presented in the test set.



a: DBSCAN b: K-means c: Agglomerative Clustering

d: DBSCAN e: K-means f: Agglomerative Clustering

g: DBSCAN h: K-means i: Agglomerative Clustering

Fig. 8: Results obtained with different algorithms for clustering. The images within a same row are the same instance segmented with different algorithms.

A. *Clustering*

To assess the performance of the three clustering algorithms tested Fig. 8 presents the results obtained for three different samples with the unsupervised learning techniques. We decide to assess the unsupervised learning algorithms in a visual way as there are no targets which allow us to compare against a ground truth.

The parameters used for the algorithms are as follows. For DBSCAN we used an ϵ of 2.0 and a minimum number of 20 points per cluster. Similarly, for K-means we assume that there are five clusters in the scene to segment. Finally, the stopping criteria of Agglomerative clustering was set to finish once five clusters were found. These parameters were set by trail-and-error under different attempts and they showed the best performance.

For the first sample Fig. 8(a - c) the three algorithms were capable to segment the car of interest. However, it is possible to see that K-means and agglomerative clustering have a line of blue points behind the vehicle which are noise.

Similarly, for the sample 2 Fig. 8(d - f) it is possible to see how DBSCAN is capable to accurately segment the car instance whereas the other two algorithms cluster a person as part of the vehicle. For the last sample Fig. 8(g - i), DBSCAN segment most of the chassis of the vehicle, however, K-means and agglomerative clustering crop almost half of the vehicle, dropping valuable information.

This behavior was seen in multiple samples during testing where DBSCAN outperformed the other clustering algorithms. Therefore, DBSCAN with an ϵ of 2.0 and a minimum number of 20 points per cluster was set to segment different vehicles within the frustum region proposal.

The idea of this instance segmentation part is to obtain the greatest number of points belonging to the instance, as more points would be translated as more information regarding the car geometry and volumetric information. This evaluation allowed us to conclude that DBSCAN is the ideal candidate to segment cars out of a given frustum region proposal.

TABLE I
Accuracy with best Hyperparameters for each model

Regression model	Training set	Test set
Lasso	77.1	75.4
SVR	95.1	**87.1**
Random Forest	**96.9**	78.0

B. Regression

The regression is in charge of predicting the 3D bounding box. To choose the best model we compare their accuracy in the Training and Test sets, each model is built with the best Hyperparameters found with GridSearch.

As seen in Table. I the best result for Training set is with Random Forest (96.9) and this have a reason, Decision Trees (DT) algorithms tend to overfit the training dataset. Also, is known that DT cannot extrapolate new data, this means its capacity to generalize with new information is very limited and this is a problem for this kind of scenarios where the inference data could have a set of new very different values. This model is discarded having in mind the problem explained before.

Comparing Training and Test scores between Lasso and SVR, this last one has better results in both sets, is shown in bold in Table. I. For this reason a complexity analysis is applied to this model as seen in Fig. 9.

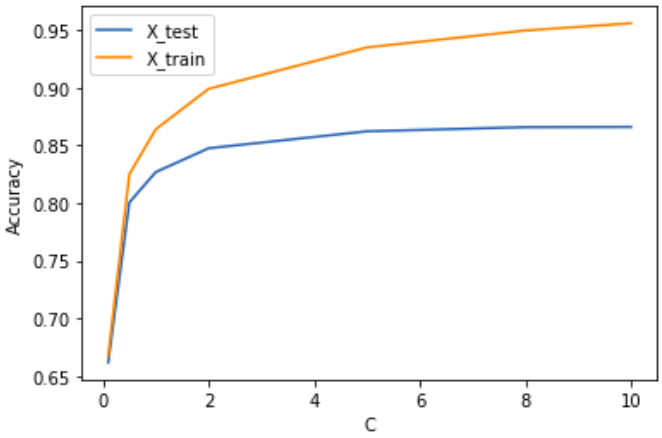
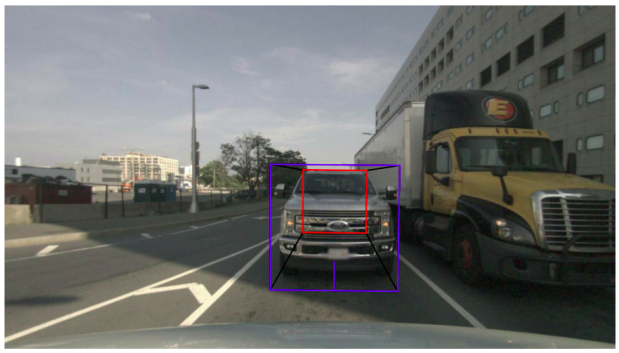


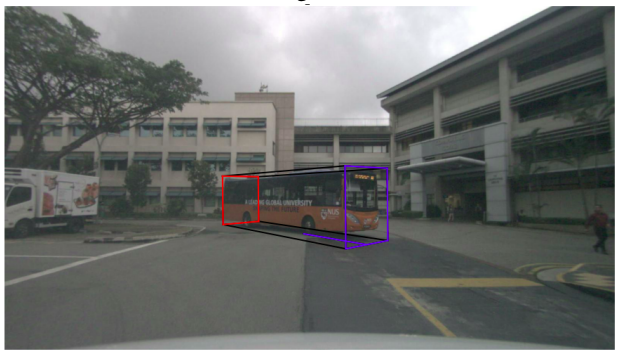
Fig. 9: SVR Complexity Analysis.

While changing parameter "C: Regularization parameter" for values [10, 8, 5, 2, 1, 0.5, 0.1], we are able to see how the accuracy changes in both sets for every new regularization value. Finally, the Test accuracy between C = 2 and C = 8 is very close, but the Training accuracy starts to get higher, and having in mind that C value is inversely proportional to the regularization parameter λ, for higher values of C means more complexity in the model and less regularization. Therefore, the regularization parameter considered is C = 8.

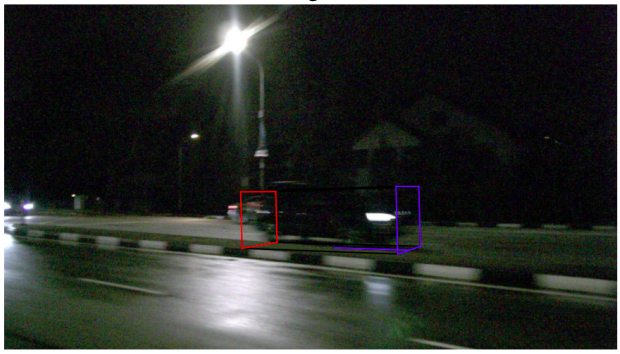
In most cases the SVR model is able to generalize the right values for centroid and bounding box size as seen for a truck Fig. 10(a), bus Fig. 10(b) and a car Fig. 10(c).



a: 3D Bounding box for Truck.



b: 3D Bounding box for Bus.



c: 3D Bounding box for Car.

Fig. 10: 3D Bounding box for different vehicles. (a) Bounding box for a Truck. (b) Bounding box for Bus with right orientation, centroid and size values. (c) Bounding box for a moving Car.

Most of the parameters for centroids are estimated accurately but bounding box size and orientation can be very variable as seen in Table. II. These results are calculated using R2 score or coefficient of determination as in Eq. 2 for the regression model and each output. SS_{res} is the sum of squares of residuals and SS_{tot} is the total sum of squares.

$$\boldsymbol{R^2} = 1 - \frac{SS_{res}}{SS_{tot}} \tag{2}$$

With this information we can compare which outputs are obtaining a better result than others between training and test, and know which characteristics present more problems or lower performance.

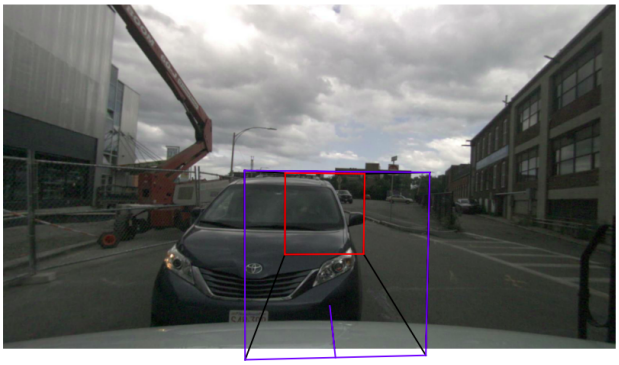
This let us focus on with which features can the model find a better relation with the target, obtaining a very good results

TABLE II
Accuracy for each output

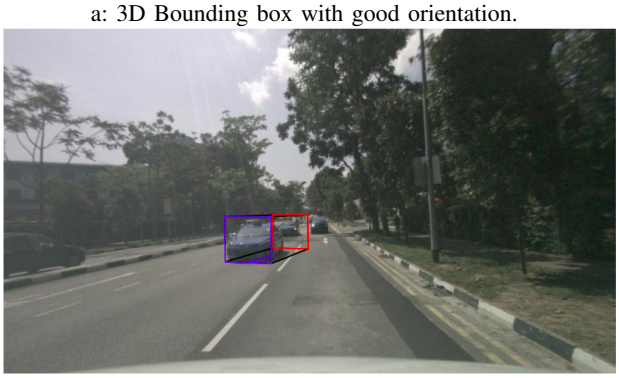
Dataset	X	Y	Z	Width	Length	Height	Yaw
Training	98.8	98.2	99.9	96.8	94.2	99.8	78.0
Test	96.6	97.8	95.4	80.7	88.4	95.0	55.7

and start to search how to enhance the prediction processes for orientation.

In some cases there are some wrong estimation of centroids as seen in Fig. 11 (b) or wrong orientation prediction as seen in Fig. 11 (a).



a: 3D Bounding box with good orientation.



b: 3D Bounding box with good centroid.

Fig. 11: Problems with bounding box regression. (a) 3D Bounding box with good orientation but wrong centroid. (b) 3D Bounding box with good centroid but wrong orientation.

VII. CONCLUSIONS

In this work, we propose a model that is capable to predict 3D bounding boxes for cars and shows promising results estimating its parameters using classical ML techniques. Even though this scenarios are commonly approached in the research community using Artificial Neural Networks (ANN) most of the times, these classical techniques have shown good performance in the tasks set. However, this could be increased with modifications in the algorithms and new assumptions to obtain state-of-the-art results with real-time inference provided by classic ML.

The orientation values are the hardest to generalize, but the regression model is powerful enough to predict good estimations for bounding box centroid and size of the detected vehicles. This means the volumetric characteristics are delivering rich information to the model thanks to the features found using ESF.

Having in mind the considerations and modifications to enhance the models performance, the project could be used in the industry of autonomous vehicles for perception and autonomous navigation. Furthermore, this project could be extended for pick-and-place tasks in warehouses or surveillance.

VIII. FUTURE WORK

Different evaluation metrics should be employed to assess the overall performance of the algorithm. Commonly, object detection tasks are assessed using the Intersection over Union (IoU) and using the area under the curve (AUC) of the Precision-Recall curve.

Furthermore, modifications such as the one presented in [18] should be considered to extend the algorithm for more class labels. Additionally, the regressor performance could be enhanced adding more features related to the orientation of the vehicle.

REFERENCES

[1] M. Irani and P. Anandan, "A unified approach to moving object detection in 2D and 3D scenes," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 20, no. 6, pp. 577-589, June 1998, doi: 10.1109/34.683770.

[2] J. C. Nascimento and J. S. Marques, "Performance evaluation of object detection algorithms for video surveillance," in IEEE Transactions on Multimedia, vol. 8, no. 4, pp. 761-774, Aug. 2006, doi: 10.1109/TMM.2006.876287.

[3] H. Li, P. Wang and C. Shen, "Toward End-to-End Car License Plate Detection and Recognition With Deep Neural Networks," in IEEE Transactions on Intelligent Transportation Systems, vol. 20, no. 3, pp. 1126-1136, March 2019, doi: 10.1109/TITS.2018.2847291.

[4] C. Rennie, R. Shome, K. E. Bekris and A. F. De Souza, "A Dataset for Improved RGBD-Based Object Detection and Pose Estimation for Warehouse Pick-and-Place," in IEEE Robotics and Automation Letters, vol. 1, no. 2, pp. 1179-1185, July 2016, doi: 10.1109/LRA.2016.2532924.

[5] Ess A, Schindler K, Leibe B, Van Gool L. Object Detection and Tracking for Autonomous Navigation in Dynamic Environments. The International Journal of Robotics Research. 2010;29(14):1707-1725. doi:10.1177/0278364910365417

[6] Yin Zhou, Oncel Tuzel; Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 4490-4499

[7] C. R. Qi, W. Liu, C. Wu, H. Su and L. J. Guibas, "Frustum PointNets for 3D Object Detection from RGB-D Data," 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Salt Lake City, UT, 2018, pp. 918-927, doi: 10.1109/CVPR.2018.00102.

[8] Ming Liang, Bin Yang, Yun Chen, Rui Hu, Raquel Urtasun; Multi-Task Multi-Sensor Fusion for 3D Object Detection. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 7345-7353

[9] C. R. Qi, H. Su, Kaichun Mo, & L. Guibas (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 77-85.

[10] Z. Zhao, P. Zheng, S. Xu and X. Wu, "Object Detection With Deep Learning: A Review," in IEEE Transactions on Neural Networks and Learning Systems, vol. 30, no. 11, pp. 3212-3232, Nov. 2019, doi: 10.1109/TNNLS.2018.2876865.

[11] D. Kellner, J. Klappstein and K. Dietmayer, "Grid-based DBSCAN for clustering extended objects in radar data," 2012 IEEE Intelligent Vehicles Symposium, Alcala de Henares, 2012, pp. 365-370, doi: 10.1109/IVS.2012.6232167.

[12] S. Ando, Y. Kusachi, A. Suzuki and K. Arakawa, "Appearance based pose estimation of 3D object using support vector regression," IEEE International Conference on Image Processing 2005, Genova, 2005, pp. I-341, doi: 10.1109/ICIP.2005.1529757.

[13] Zhou, Y., Sun, P., Zhang, Y., Anguelov, D., Gao, J., Ouyang, T., Guo, J., Ngiam, J. & Vasudevan, V.. (2020). End-to-End Multi-View Fusion for 3D Object Detection in LiDAR Point Clouds. Proceedings of the Conference on Robot Learning, in PMLR 100:923-932

[14] Bin Yang, Wenjie Luo, Raquel Urtasun; PIXOR: Real-Time 3D Object Detection From Point Clouds. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018, pp. 7652-7660

[15] Shen, Xiaoke & Stamos, Ioannis. (2020). Frustum VoxNet for 3D object detection from RGB-D or Depth images. 1687-1695. 10.1109/WACV45572.2020.9093276.

[16] Wang, Zhixin & Jia, Kui. (2019). Frustum ConvNet: Sliding Frustums to Aggregate Local Point-Wise Features for Amodal 3D Object Detection.

[17] P. Cao, H. Chen, Y. Zhang and G. Wang, "Multi-View Frustum Pointnet for Object Detection in Autonomous Driving," 2019 IEEE International Conference on Image Processing (ICIP), Taipei, Taiwan, 2019, pp. 3896-3899, doi: 10.1109/ICIP.2019.8803572.

[18] Wang, Chunxiao & Ji, Min & Wang, Jian & Wen, Wei & Li, Ting & Sun, Yong. (2019). An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation. Sensors. 19. 172. 10.3390/s19010172.

[19] Y. Salih, A. S. Malik, D. Sidibé, M. T. Simsim, N. Saad and F. Meriaudeau, "Compressed VFH descriptor for 3D object classification," 2014 3DTV-Conference: The True Vision - Capture, Transmission and Display of 3D Video (3DTV-CON), Budapest, 2014, pp. 1-4, doi: 10.1109/3DTV.2014.6874757.

[20] W. Wohlkinger and M. Vincze, "Ensemble of shape functions for 3D object classification," 2011 IEEE International Conference on Robotics and Biomimetics, Karon Beach, Phuket, 2011, pp. 2987-2992, doi: 10.1109/ROBIO.2011.6181760.

[21] H. Caesar, Varun Bankiti, A. Lang, Sourabh Vora, Venice Erin Liong, Q. Xu, A. Krishnan, Y. Pan, Giancarlo Baldan, & Oscar Beijbom (2020). nuScenes: A Multimodal Dataset for Autonomous Driving 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), 11618-11628.

[22] Andreas Geiger, Philip Lenz, C. Stiller, & R. Urtasun (2013). Vision meets robotics: The KITTI dataset The International Journal of Robotics Research, 32, 1231 - 1237.

[23] Bolles, R. and M. Fischler. "A RANSAC-Based Approach to Model Fitting and Its Application to Finding Cylinders in Range Data." IJCAI (1981).

[24] Peter J. Rousseeuw (1987). "Silhouettes: a Graphical Aid to the Interpretation and Validation of Cluster Analysis". Computational and Applied Mathematics 20: 53–65. doi:10.1016/0377-0427(87)90125-7.