Ultra-Fast Lidar Scene Analysis Using Convolutional Neural Network

Houssem Moussa, Valentin Gies, Thierry Soriano

Ultra-Fast Lidar Scene Analysis using Convolutional Neural Network

Houssem Moussa[1], Valentin Gies[2][0000−0002−2099−1150], and Thierry Soriano[3][0000−0002−4883−0447]

[1] Université de Toulon, SeaTech, France
[2] Université de Toulon, Laboratoire IM2NP - UMR 7334, France
gies@univ-tln.fr
[3] Université de Toulon, Laboratoire COSMER - EA 7398, France
thierry.soriano@univ-tln.fr
//https://cosmer.univ-tln.fr/en/

Abstract. This work introduces a ultra-fast object detection method named FLA-CNN for detecting objects in a scene from a planar LIDAR signal, using convolutional Neural Networks (CNN). Compared with recent methods using CNN on 2D/3D lidar scene representation, detection is done using the raw 1D lidar distance signal instead of its projection on a 2D space, but is still using convolutional neural networks. Algorithm has been successfully tested for RoboCup scene analysis in Middle Size League, detecting goal posts, field boundary corners and other robots. Compared with state of the art techniques based on CNN such as using Yolo-V3 for analysing Lidar maps, FLA-CNN is 2000 times more efficient with a higher Average Precision (AP), leading to a computation time of of $0.025ms$, allowing it to be implemented in a standard CPU or Digital Signal Processor (DSP) in ultra low-power embedded systems.

Keywords: Lidar processing, Scene analysis, Convolutional Neural Networks, Low-power

1 Introduction

Mobile autonomous robots in unknown or changing environment need to take decisions based on the surrounding scene analysis. For this task, two types of sensors are mainly used : cameras and LIDAR. This latter is an interesting exteroceptive sensor in robotics providing reliable maps of the surrounding environment, with a better precision in object positioning than using only cameras. This interesting feature greatly helps to ensure a high level of safety in human-robots interactions.

This paper focuses on autonomous robot soccer scene analysis, including robots, humans, goals (posts), and field boundary using a planar Lidar, but with limited computing capabilities, and at least without using a GPU (such as for image processing). Chosen Lidar is a Pepperl+Fuchs R2000 UHD one generating a 1D sequence of 1440 distance measurements at each scan of the scene,

50 times per second, with a maximum range of 60 m. Application of this paper is analysis of RoboCup Middle Size League scenes. RoboCup is an international competition where robot teams play soccer autonomously. Its main objective in the future is to won against a professional, human soccer team by 2050. In the Middle Size League (MSL), teams are composed of five robots playing autonomously on a 22 m by 14 m soccer field with a real soccer ball.

Having limited computing capabilities have a deep impact on the algorithms that can be used in a robot. Most scene analysis algorithms are using 2D or even 3D representations of the scene, leading to huge computation time on limited computing systems. This paper aims at introducing a novel algorithm for a LIDAR scene analysis, using state of the art CNN with end to end learning, but without using a 2D representation of the scene. Instead of that CNN is applied to the raw 1D lidar signal, using the 1440 distance data of each LIDAR rotation as input tensor. This algorithm is called Fast Lidar Analysis using Convolutional Neural Network (FLA-CNN). It achieves an excellent precision while having a very low detection time and a low power consumption, making it usable for mobile robots with CPU or DSP for real time detection, without requiring a GPU.

This work is divided into 3 parts :

- In section 2, an overview of scene analysis techniques using cameras or LIDAR is presented, focusing on their advantages and disadvantages.
- In section 3, Fast Lidar Analysis using Convolutional Neural Network (FLA-CNN) algorithm is introduced, allowing to analyse scenes with low latency and computing power requirements.
- In section 4, application to the RoboCup scene analysis is presented, with a focus on dataset creation and labeling, training process, and a discussion on results and performance.

2 State of the art in robot scene analysis

Deep neural networks, and particularly convolutional neural networks are considered as state of the art models for feature extraction due to their great ability to learn their features and classifiers from data into an end to end learning process. However, their main drawback is to require high computing capabilities, making them difficult to implement in an embedded computer or microcontroller. Moreover, using these algorithms for controlling fast robots in real time, requires a fast processing time considering the speed of the robots (up to 6 m/s in Middle Size League) corresponding to at least 30 frame per second (FPS). Combining these two aspects, high frame rate and embedded processing, is the key for efficient embedded robot control.

2.1 Object detection based on images

Camera is one of the most used sensor in robotics combined with processing for scene analysis. Among the most efficient ones are detectors based on CNN, hav-

ing one stage or two stages. Both of them have interesting performances in terms of accuracy, but are relatively slow [1, 2]. [1] gives an interesting comparison of these detectors on Coco dataset : using an Intel i7-8700 CPU and an NVIDIA GeForce RTX 2080Ti 12GB GPU on 640×960 images, Average Precision (AP) reaches 32.4 on the most accurate models (*i.e.* Faster RCNN Res2Net101), but at the price of a computation time of 63 ms. Reaching a computation speed of 30 FPS requires to use Mobile Net Models at the price of a loss in accuracy (mAP = 24.3).

An interesting result in [1] is that two stages models [3–8] are more accurate than one stage ones such as YOLO [9–11], even if the main trend is nowadays to use these latter. Fig. 1) shows a result of scene analysis on a 360 degrees image using YOLOv3. This image has been recorded using an omnidirectional camera and transformed to a panorama image because Yolo algorithms are not rotation invariant and cannot be applied directly to omnidirectional images. Models have been proposed to cope with this issue [12, 13] but are more resource consuming.

Fig. 1. YOLOv3 for object detection running on a GPU with omnidirectional camera.

In conclusion, cameras are potentially rich sensors but requires a high processing power for extracting segmenting the scene at a high FPS rate, making them difficult to use on embedded CPU or micro-controllers. Considering stereovision cameras would also be interesting, but it would require more computing power making them out of scope on proposed application to MSL robots.

2.2 Object detection based on 2D Map lidar images

An alternative to the use of cameras, is to use 2D or 3D lidars. An example of 2D map obtained using a Pepperl+Fuchs R2000 lidar is presented in Figure 2.

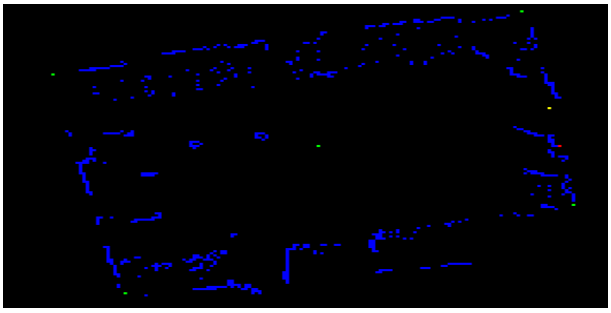


Fig. 2. 2D Map Image created with 1D Lidar signal

This kind of 2D image can be analysed using image processing algorithms for detecting shapes or objects. However, doing that would lead to the same drawbacks as for image processing : important computation time and computer power requirement.

Considering the strong constraints of embedded systems in terms of limitations in computing power and the need for high FPS, another approach is proposed in this paper, using CNN on th raw 1D-Lidar signal.

3 Contribution : Fast Lidar Analysis using Convolutional Neural Network (FLA-CNN)

In this paper, Fast Lidar Analysis using Convolutional Neural Network (FLA-CNN) is introduced. It aims at analysing 1D raw lidar distance data using a state of the art deep learning model predicting corners of a RoboCup field and posts of a RoboCup goal. FLA-CNN design has been inspired by YOLO : its design is being presented in details.

Notation We use $P = (P_d, P_\theta, P_{conf}, P_{class}) \in \Re^4$ to denote a ground-truth points, where P_d, P_θ are the distance and its corresponding angle in robot referential, $P_d \in [0, 60]$ and $P_\theta \in [0, 2\pi]$. P_{conf} is the confidence score of an existing object and P_{class} is the index of the corresponding class name to the detected object. Similarly $\hat{P} = (\hat{P}_d, \hat{P}_\theta, \hat{P}_{conf}, \hat{P}_{class}) \in \Re^4$ denotes a predicted object.

3.1 CNN Architecture

FLA-CNN network can be divided into two parts: feature extractor neural network (FNN) (Eq. 1) and object regression network (ORN) (Eq. 2).

$$F = FNN(D) \tag{1}$$

$$T = ORN(F) \tag{2}$$

where $D \in R^{1440}$ is the input tensor featuring the measured distance for each angle during a full rotation. F denotes a feature vector and T denotes a list of predicted points that represents the objects in 1D signal in transformed notation (the relationship between T and P will be defined soon – Eq. 11). Figure 5 shows the overall the FLA-CNN architecture, while below we describe each stage in some depth.

Feature extractor network FNN: The feature extractor network, takes an input vector D and outputs a list of features T. This stage is composed with 2 layers, each layer apply a convolution 1D, batch normalisation, max pooling and leaky relu as an activation function. The first layer takes as input the 1D signal(D), the second layer takes as input the output of the first layer. This network extract features from signal, only one layer is not capable to identify the signal for that reason we added the second layer that will complete the recognition stage.

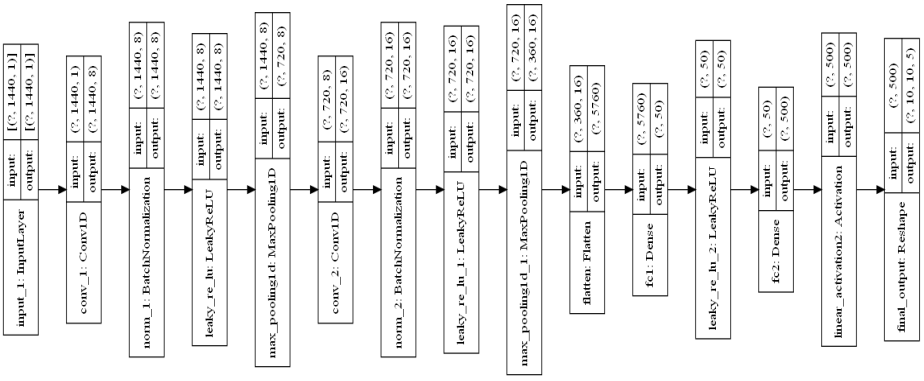


Fig. 3. Model network For object detection using 1D lidar signal detection

The convolution 1D is faster than 2D one, decreasing drastically computational complexity, making it suitable for running on mobile devices, embedded systems, and on some microcontrollers and DSP.

In each convolutional layer, the forward propagation is expressed as follows :

$$x_k^l = b_k^l + \sum_{i=1}^{N_{l-1}} conv1D(w_{ik}^{l-1}, s_i^{l-1}) \tag{3}$$

where x_k^l is defined as the input, b_k^l is defined as the bias of the k^{th} neuron at layer l, s_i^{l-1} is the output of the i^{th} neuron at layer l-1, w_{ik}^{l-1} is the kernel from the i^{th} neuron at layer l-1 to the k^{th} neuron at layer l.

The output of the convolutional layer is followed by a batch normalisation layer for fixing means and variances. Learning is done using stochastic optimisation due to the memory limits, for reducing over-fitting and training time. Normalisation process is expressed as follows, where B is a mini-batch of size m of the whole training dataset. The mean and variance of B could be expressed as :

$$\mu_B = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{4}$$

$$\sigma_{B^2} = \frac{1}{m}\sum_{i=1}^{m}(x_i - \mu B)^2 \tag{5}$$

For a layer of the network with d-dimensional input, $x = (x^{(1)}, ..., x^{(d)})$, each dimension of its input is then normalized (i.e. re-centered and re-scaled) separately,

$$\hat{x}_i^k = \frac{x_i^k - \mu_B^k}{\sqrt{\sigma_{B^2}^{k^2} + \epsilon}} \tag{6}$$

where $k \in [1, 1440]$ and $i \in [1, m]$; $\mu_B^{(k)}$ and $\sigma_B^{(k)^2}$ are the per-dimension mean and variance, respectively.

ϵ is added in the denominator for numerical stability and is an arbitrarily small constant we used $\epsilon = 1e^{-6}$. The resulting normalized activation $\hat{x}^{(k)}$ have zero mean and unit variance, if ϵ is not taken into account. To restore the representation power of the network, a transformation step then follows as

$$y_i^k = \gamma^k x_i^k + \beta^k \tag{7}$$

Object regression network Following the FNN, a separate MLP is applied to each feature vector F to produce a transformed version of object points predictions, noted $T < N_\theta, N_d, 3 + N_{classes} >$, a 3-dimensional matrix where $N_\theta = 10$ is the numbers of angular cells (size of each cell is 0.63 radian). $N_d = 10$ is the numbers of radial cells (size of each cell is 6 meter). First and second indexes correspond to the location (θ, d) in the prediction polar grid, and last index corresponds to the intra-cell predicted position and classification of the point in the considered polar grid cell with the following information : $T_{Cell\theta}$ the predicted point angle in the considered cell, T_{Celld} the predicted distance in the considered cell, $T_{CellConf}$ the probability that an object exist in the considered cell and $T_{CellClass_i}$ the probability that the object belongs to the i^{th} class.

Exact coordinates of the object in the signal can be obtained using the following equations :

$$P_d = (d + Sig(T_{Celld}))S_d \tag{8}$$
$$P_\theta = (\theta + Sig(T_{Cell\theta}))S_\theta \tag{9}$$
$$P_{conf} = Sig(T_{CellConf})) \tag{10}$$
$$P_{class_i} = softmax(T_{CellClass_i}) \tag{11}$$

where Sig($\cdot$) is the logistic (sigmoid) activation function is defined by :

$$sig(x) = \frac{1}{1 + e^{-x}} \tag{12}$$

where Softmax() is the normalized exponential function is defined by :

$$softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{k} e^{z_j}} \text{ for i=1...k and } z = (z_1...z_k) \in \Re^k \tag{13}$$

where S_d is the radial grid size $S_d = \frac{max(d)}{N_d}$
and where S_θ is the angular grid size $S_\theta = \frac{2\pi}{N_\theta}$

A flatten layer has been added between the FNN and OPN, and all the feature vectors obtained with the CNN 2 layers in the FNN are transformed to one vector that will be the input for the final multi layer perceptron. The number of neurons in the last MLP layer must be equal to $(N_d * N_\theta * (N_c + 3)$, where N_θ is the number of grid bellowing to x, N_d is the number of grid bellowing to y axis and 5 represent the predictions coordinates ($\hat{T} = (\hat{T}_{Celld}, \hat{T}_{Cell\theta}, \hat{T}_{CellConf}, \hat{T}_{CellClass_i})$ $\in \Re^5$) where N_c is the class numbers.

4 Application to the RoboCup scene analysis

4.1 Data set creation and labelling

Data uses in this work have been recorded using Robot Club Toulon (RCT) robots participating to RoboCup Middle Size League, using a Pepperl+Fuchs 1D lidar delivering 1440 distance measurements per rotation (angular resolution is 0.25°), 50 times per second with a maximum range of 60m. Its precision is approximately $\pm 1cm$. This work aims at detecting field boundary corners and goal posts. These information are sufficient for computing position of our robot in the field using distance and angles of each detected post, and developing strategies for playing soccer and shooting with precision.

Signal labelling application has been designed for labelling 1D lidar signal easily, with labels having the same format as the model output discussed in (3.1). This labelling tool uses data files containing timestamped Lidar data that is decoded and extracted to create one lidar file for every lidar sample, each file

name containing the timestamp of the lidar acquisition. Each file file contains 1440 lines and 2 columns, first one for the distance and second one for the lidar angle in robot referential.

In a LIDAR scene recorded in our RoboCup field, four points can be considered as field boundary corners, and two of them can be considered as posts (our dataset has been recorded with only one goal in the scene). Creating labels directly on the 1D signal is difficult and would lead to many faults in the dataset : using a 2D representation, only for labeling 1D signal, is a better way to easily find and label field corners and posts in the scene (Figure 4.1).

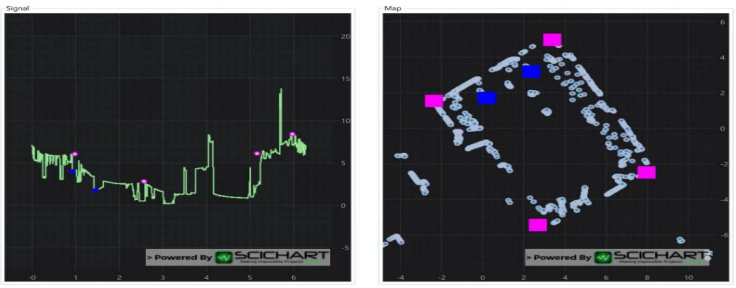


Fig. 4. LIDAR data signal labeling : 1D signal (left) and 2D map (right).

For transforming each 1D lidar sample to a 2D map, following transformation is used, where d is the measured distance, θ the measurement angle, and (x,y) the cartesian position of the obstacle in the robot referential :

$$x = \cos(\theta) * d \tag{14}$$

$$y = \sin(\theta) * d \tag{15}$$

Once the map is created, object class is selected manually, and cartesian map coordinates are transformed back to polar coordinates for labeling a specific point in the 1D signal. As shown in (Figure 4.1), 4 points have been labelled in pink for the boundary corners, and 2 points have been labelled in blue for the posts. Inverse transformation (from cartesian to polar point) is expressed as follows :

$$d = \sqrt{x^2 + y^2} \tag{16}$$

$$\theta = 2\arctan(\frac{y}{x + d}) \tag{17}$$

This process is repeated for every lidar Sample, after each annotation for each sample an xml file is created that contain distance, angle and the class

name of each object which is represented by a point on the 2D Map. Once the labeling process is finished, a new dataset with a unique *.txt* file containing lidar 1D data and its corresponding *.xml* file for the labels is generated.

4.2 Training

FLA-CNN has been trained from scratch on our dataset, as model has been fully customised for our application. During training, model prediction has been optimised by minimising the loss function (Eq.18). Output of our model is a tensor of dimansion $< N_d = 10 \times N_\theta = 10 \times (3 + N_{Classes}) = 5) >$, since we have 2 objects classes (goals and corners) to detect, so that every grid may contain 1 or in some cases 2 objects. Anchors boxes for multiple detection in the same grid cell are not used, since object bounding boxes have same size always in the 2D map.

Proposed loss function is a sum of squared errors between the different components of the predictions and the ground truth. First term is related to angle and distance only when there is an object. 1_i^{obj} is equal to 1 when there is an object in the grid cell i, and to 0 if there is no object. Second and third terms are related to prediction confidence when there is an object 1_i^{obj} and when there is no object 1_i^{noobj}. $\lambda_{pred_obj} = 3$ is higher than $\lambda_{pred_noobj} = 2.5$ to focus on objects. Fourth term is related to the probability that an object belongs to each one of the classes.

.

$$\begin{aligned} \text{loss} = \lambda_{cord} \sum_{i=0}^{S^2} 1_i^{obj} [(P_\theta^i - \hat{P_\theta}^i)^2 + (P_d^i - \hat{P_d}^i)^2] \\ + \lambda_{pred\_obj} \sum_{i=0}^{S^2} 1_i^{obj} (P_{conf}^i - \hat{P}_{conf}^i)^2 \\ + \lambda_{pred\_noobj} \sum_{i=0}^{S^2} 1_i^{noobj} (P_{conf}^i - \hat{P}_{conf}^i)^2 \\ + \sum_{i=0}^{S^2} \sum_{j=1}^{k} 1_i^{obj} (P\_class_j^i - \hat{P}\_class_j^i)^2 \end{aligned} \tag{18}$$

Network has been trained on 336 samples of 1440 elements. 200 epochs have been iterated, with a batch size of 8. Optimiser used is adam, with a learning rate $= 1e-4$, decay$= 0.005$ and beta $= 0.99$. To avoid overfitting issues, training has been stopped when loss function was increasing for 10 epochs. Only the best weights of the last 10 epochs will be saved.

5 Results

The dataset that we created in section 4 is devided in 2 parts, 80% for training and 20% for the evaluation, we used the AUC of the ROC curve to evaluate our detection results, ROC curve is a graph showing the performance of a detection model at all detection thresholds. This curve plots two parameters : True Positive Rate (TPR) (Eq. 19) and False Positive Rate (FPR) (Eq. 20)

$$TPR = \frac{TP}{TP + FN} \tag{19}$$

$$FPR = \frac{FP}{FP + TN} \tag{20}$$

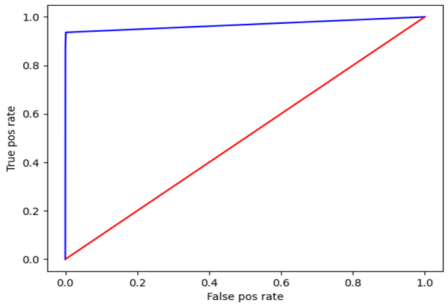
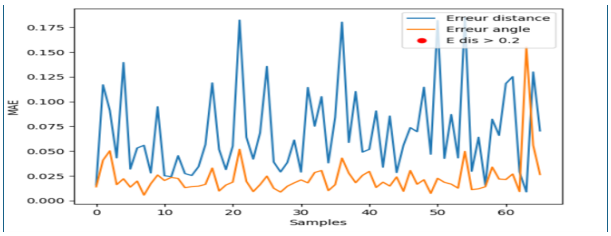


Fig. 5. Model evaluation with Roc curve

The ROC curve (Fig .5) that we obtain on the evaluation dataset for different threeshold starting with 0 ending with 1 with a step of 0.1, for our model the AUC = 0.94 which is an excellent result in term of precision.

We used other metric to evaluate the detection results which is the mean absolute error between the predictions of our model and the ground truth of the evaluation dataset (Fig.5), the maximum error of the distance is 0.17cm, and the maximum angular error is 0.05 radian (2 degree).



The detection results are shown above, the model predicts the distance and angle of each object that below to pink and blue on the signal and using the transformation discussed in (3) we are showing the correspondent object position in the 2D MAP.

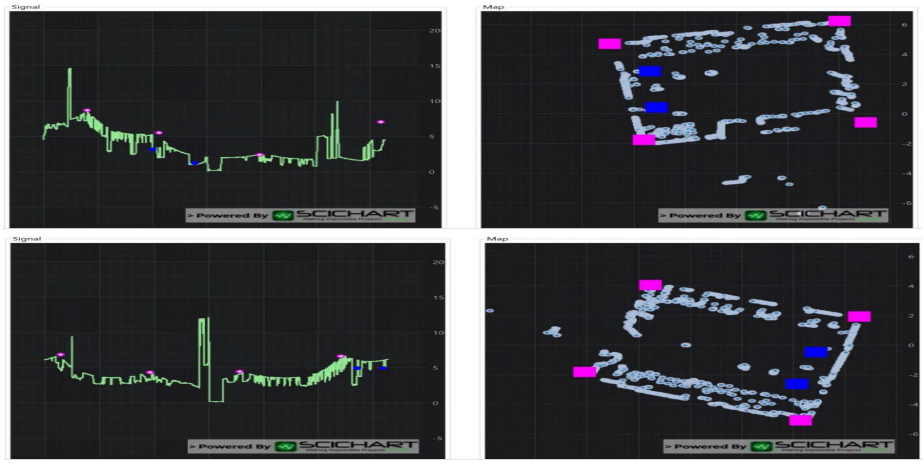


Fig. 6. Model predictions for different samples : : 1D signal (left) and 2D map (right).

6 Conclusion

This work introduces a ultra-fast object detection method named FLA-CNN for detecting objects in a scene from a planar LIDAR signal, using convolutional Neural Networks (CNN). Compared with recent methods using CNN on 2D/3D lidar scene representation, detection is done using the raw 1D lidar distance signal instead of its projection on a 2D space, but is still using convolutional neural networks.

Algorithm has been successfully tested for RoboCup scene analysis in Middle Size League, detecting goal posts, field boundary corners and other robots. Prediction inferences are computed in 25 μs with 94% of mAP. Thanks to the reduced size of the proposed CNN processing directly 1D lidar signal instead of converting it to a 2D image, implementing it in an embedded system is possible and doesn't require a high computational power such as a GPU, but can be achieved in a DSP or a microcontroller for real time detection in mobile robots.

In term of precision and speed our model achieve the better results compared to the start of art methods that we tested ourselves as explained as follows :

Method	Input size	mAP	time(ms)	computing system
SSD	321x321x3	45.4	61	GPU (GTX 1080)
SSD	513x513x3	50.4	125	GPU (GTX 1080)
YOLO_v3	416x416x3	55.3	29	GPU (GTX 1080)
YOLO_v3	608x608x3	57.9	51	GPU (GTX 1080)
Faster_RCNN	1000x600x3	73.2	142	GPU (GTX 1080)
Ours (FLA-CNN)	1440x1	94	0.025	CPU (i5-9500)

Even if proposed results are preliminary and require a large database with many objects and labels to detect for a full validation of the proposed algorithm, first results are promising and clearly show that converting a 1D lidar data into a 2D map leads to dramatically increase computation power requirement.

References

1. Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, 13(1), 2021.
2. Zhuangli Hu, Tong He, Yihui Zeng, Xiangyuan Luo, Jiawen Wang, Sheng Huang, Jianming Liang, Qinzhang Sun, Hengbin Xu, and Bin Lin. Fast image recognition of transmission tower based on big data. *Protection and Control of Modern Power Systems*, 3, 12 2018.
3. Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
4. Joseph Redmon and Ali Farhadi. Yolov3: An incremental improvement, 2018.
5. Koen E. A. van de Sande, Jasper R. R. Uijlings, Theo Gevers, and Arnold W. M. Smeulders. Segmentation as selective search for object recognition. In *2011 International Conference on Computer Vision*, pages 1879–1886, 2011.
6. F. Sultana, A. Sufian, and P. Dutta. A review of object detection models based on convolutional neural network. In *Advances in Intelligent Systems and Computing*, pages 1–16. Springer Singapore, 2020.
7. Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.
8. Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.
9. Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.
10. Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.
11. Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *CoRR*, abs/2004.10934, 2020.
12. Zhihao Duan, Mustafa Ozan Tezcan, Hayato Nakamura, Prakash Ishwar, and Janusz Konrad. Rapid: Rotation-aware people detection in overhead fisheye images. *CoRR*, abs/2005.11623, 2020.
13. Gui-Song Xia, Xiang Bai, Jian Ding, Zhen Zhu, Serge J. Belongie, Jiebo Luo, Mihai Datcu, Marcello Pelillo, and Liangpei Zhang. DOTA: A large-scale dataset for object detection in aerial images. *CoRR*, abs/1711.10398, 2017.