3D OBJECT DETECTION USING TEMPORAL LIDAR DATA

Scott McCrae and Avideh Zakhor

Department of Electrical Engineering and Computer Sciences
University of California, Berkeley
{mccrae, avz}@berkeley.edu

ABSTRACT

3D object detection is a fundamental problem in the space of autonomous driving, and pedestrians are some of the most important objects to detect. The recently introduced PointPillars architecture has been shown to be effective in object detection. It voxelizes 3D LiDAR point clouds to produce a 2D pseudo-image to be used for object detection. In this work, we modify PointPillars to become a recurrent network, using fewer LiDAR frames per forward pass. Specifically, as compared to the original PointPillars model which uses 10 LiDAR frames per forward pass, our recurrent model uses 3 frames and recurrent memory. With this modification, we observe an 8% increase in pedestrian detection and a slight decline in performance on vehicle detection in a coarsely voxelized setting. Furthermore, when given 3 frames of data as input to both models, our recurrent architecture outperforms PointPillars by 21% and 1% in pedestrian and vehicle detection, respectively.

***Index Terms*—** LiDAR, recurrent network, object detection, autonomous driving

1. INTRODUCTION

In the space of perception for autonomous vehicles, there exist numerous pipelines for object detection, semantic segmentation, tracking, and more [1, 2, 3, 4, 5, 6, 7, 8]. Perception is a fundamental problem in autonomous driving, and object detection is a core aspect of perception. Recent works have been focused on feature extraction on LiDAR point clouds [9] and sensor fusion [5].

Many recent papers on deep learning on point clouds use either the PointNet [9] or PointNet++ [4] architectures. The former produces a global feature vector for the entire point cloud, which can then be used for a multitude of tasks. It was improved upon in [4] with the introduction of pointwise hierarchical features, addressing a major drawback of [9] by including a notion of local features in addition to global features.

There are several main approaches used to process LiDAR data for object detection. Some method operates directly on LiDAR data in the point cloud space [2], some voxelize the point cloud [10, 11], and others use the Birds Eye View (BEV) [3, 1, 6]. Additionally, recent work [5, 7, 12, 2] has demonstrated the efficacy of sensor fusion.

There has also been much research on 2D object detection [8, 13, 14, 15]. For instance, [15] operates on individual images yet is fast enough to run real-time on video. More recently, [8] demonstrated the effectiveness of temporal memory in producing fast and accurate detection results. Meanwhile, [1] extends the work on 2D sequential object detection to 3D. The authors modify [6], and compare stacked frames of BEV LiDAR data from KITTI as input versus using a Convolutional Long-Short-term-Memory (ConvLSTM) layer and processing data in a recurrent manner; they observe the ConvLSTM approach to be superior. [3] takes in several frames of BEV LiDAR data, but does not experiment with recurrent networks.

Until mid-2019, researchers were constrained by a lack of publicly-available datasets with large amounts of labelled temporal LiDAR data. Newly available datasets from companies such as Waymo and NuTonomy use more modern sensors and feature data that is captured in many series of 20 seconds each. While previous research has mostly been constrained to processing individual LiDAR frames, new data has opened the door to training models which use this sequential data. Since this data is new, not much research in this vein has been published yet.

We are motivated by the availability of this new data to develop methods which leverage temporal information in LiDAR. In autonomous driving applications, vehicles continuously collect data while driving, so it is natural to develop object detection solutions with this temporal characteristic in mind. We also recognize pressing safety concerns, and place a focus on fast, low-latency systems which can help avoid collisions. Along these lines, we propose a modification to an existing 3D object detection architecture in order to exploit the temporal aspects of available LiDAR frames. Specifically, we opt to modify the recently proposed PointPillars architecture [10], which allows for end to end learning on voxelized point clouds without using 3D convolution. Specifically, it uses an encoder that learns features on pillars, or vertical columns of the point cloud to predict 3D oriented boxes for objects.

The authors gratefully acknowledge funding from Berkeley DeepDrive.

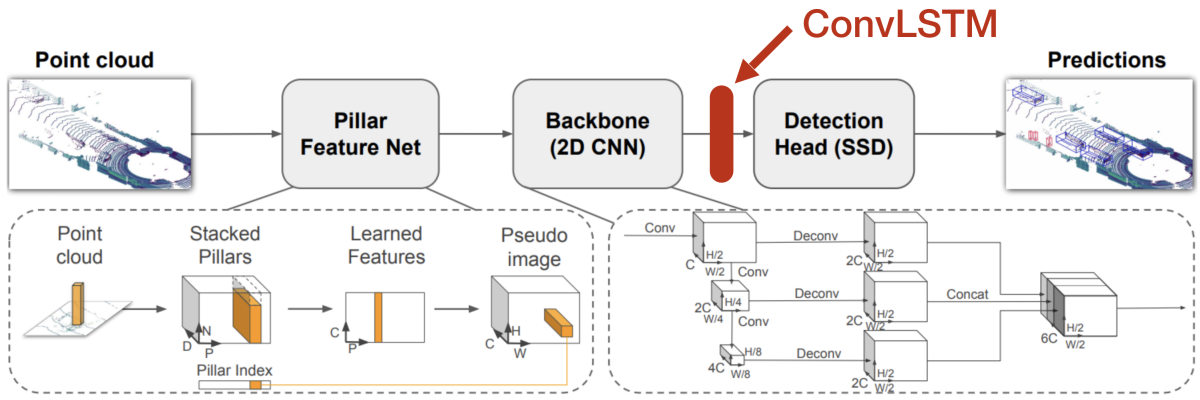

Fig. 1: *Recurrent PointPillars architecture. The original work in [10] has three main stages: a) feature extraction from the point cloud, b) 2D CNN for processing the point cloud pseudo-image, and c) an SSD detection head. Our ConvLSTM layer, shown in red, takes input from the 2D CNN backbone and the ConvLSTM output is directly used by the SSD for predictions.*

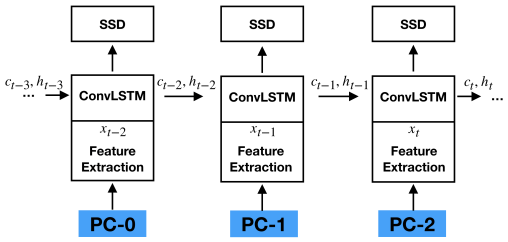

Fig. 2: *An illustration of how information recurrent information is propagated through our network. Recurrent features are passed from one iteration to the next via the ConvLSTM, which accepts recurrent features and the featurized point cloud as input, and outputs features for the SSD detection head.*

PointPillars has several advantages over most existing object detection methods: First, by learning features instead of relying on fixed encoders, PointPillars can leverage the full information represented by the point cloud. Further, by operating on pillars instead of voxels there is no need to tune the binning of the vertical direction by hand. Finally, pillars are highly efficient because all key operations can be formulated as 2D convolutions which are extremely efficient to compute on a GPU. An additional benefit of learning features is that PointPillars requires no hand-tuning to use different point cloud configurations. For example, it can easily incorporate multiple LiDAR frames, or even radar point clouds.

The outline of the paper is as follows: In Section 2, we provide an overview our proposed approach. Section 3 includes experimental results, and Section 4 is conclusion.

2. PROPOSED APPROACH

We begin with an overview of the original PointPillars [10] method, followed by our extension to a recurrent architecture.

2.1. Overview of PointPillars

The PointPillars pipeline operates on LiDAR data to produce a 2-dimensional pseudo-image of the 3D space around the ego-vehicle and uses this representation to generate 3D bounding boxes of objects in the scene. The network operates on LiDAR data. Figure 1 shows the basic PointPillars architecture with the addition of our proposed modification, namely ConvLSTM. As seen, it first produces a $W \times L$-dimensional grid of pillars with height H around the ego-vehicle, and uses PointNet [9] to extract an n-dimensional feature vector describing the LiDAR points in each pillar. Each grid location receives a feature vector, effectively producing a $W \times L \times n$ featurized representation of the scene. This representation is subsequently passed through a 2D convolutional neural network (CNN) backbone, and then fed into a single-shot detector (SSD) head. This framework avoids the expensive 3D convolution operation, although it does effectively voxelize the 3D space.

2.2. Recurrent PointPillars

To leverage the temporal data available in nuScenes [16], PointPillars [10] uses data from the vehicle's inertial measurement unit to estimate and apply a rotation and translation on prior point clouds to transform them to the ego-vehicle's current reference frame. It also decorates each point with a timestamp to indicate when each point in the combined point cloud was recorded.

One of the main weaknesses in [10] is that its input is 10 frames of LiDAR data, aligned by applying rotations and translations as described above. With the LiDAR sensor measuring at 20Hz, this is 0.5 seconds worth of data. Travelling at 60 miles per hour, a vehicle might cover 44 feet in the time it takes to collect data for detection. Furthermore, while stationary objects may be reinforced by this operation, moving objects will be smeared in the final point cloud.

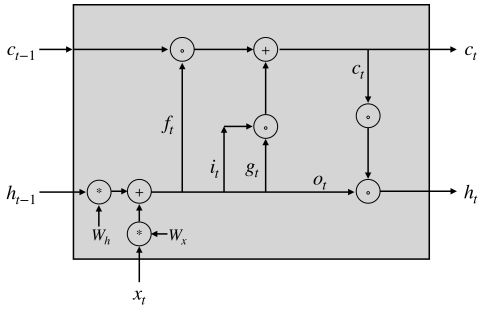

Fig. 3: *Convolutional Long Short-term Memory diagram.* $\circ$ *denotes the Hadamard product, and* $*$ *denotes convolution.* h_t *is taken as the output of this layer. In our implementation,* x_t *is the current featurized point cloud representation, and* h_{t-1}, c_{t-1} *capture the state of the ConvLSTM layer.*

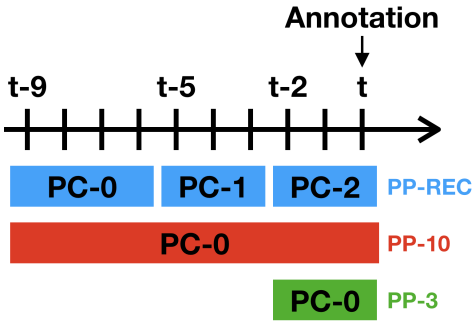

Fig. 4: *An illustration of point cloud processing at training time. Each tick mark represents a LiDAR frame, with a ground truth annotation at time* t*. Each colored rectangle represents a point cloud comprised of the LiDAR frames for the ticks it overlaps with, with corresponding labels.*

To address these problems, we propose using a recurrent network architecture as shown in Figure 1. The PointPillars pipeline naturally lends itself to a ConvLSTM layer due to its convolutional backbone. We insert a ConvLSTM module between the output of the convolutional backbone and the input of the single-shot detector. This ConvLSTM module allows the network to propagate information contained in each featurized point cloud pseudo-image through time, as shown in Figure 2. As seen in Figure 3, the ConvLSTM layer, introduced in [17], extends the fully connected LSTM to have convolutional structure. ConvLSTM removes redundancies in the fully connected LSTM approach [17] by leveraging spatial information via convolution. Rather than requiring a dense point cloud comprised of many LiDAR frames as input, our network can take a point cloud with fewer frames and instead leverage its memory of the past. This leads to several benefits: first, processing less data at the feature extraction stage reduces complexity; and second, we preserve more structural integrity in the point clouds due to less smearing or artifacting due to object motion.

At training and testing time, we use data looking backwards from the ground truth annotation. Figure 4 illustrates how our proposed method and the original differ in data handling. Our method, referred to as PP-REC, divides the most recent 10 LiDAR frames into three point clouds. We run our network on each of these point clouds in sequence, using the older two point clouds, PC-0 and PC-1, to build recurrent memory, and using the most recent point cloud, PC-2, to produce detection results. Our model is given 3 LiDAR frames, namely PC-2 and its recurrent memory, which is generated with up to 7 additional prior LiDAR frames. In contrast, our implementations of the original PointPillars approach, referred to as PP-10 and PP-3, use 10 or 3 frames of LiDAR respectively. Although not necessary for autonomous driving applications, we abide by the nuScenes convention of using at most 10 LiDAR frames for the detection task [16]. Notably, this means that each pass through our detection network uses fewer frames than the original PointPillars work; specifically, our network uses 3 LiDAR frames and recurrent memory, while the original work would require 10 LiDAR frames for each detection.

3. EXPERIMENTS

We now describe and analyze the results of several experiments which demonstrate the efficacy of our recurrent approach. There are two important parameters which define the performance of the models we evaluate. Namely the amount of data used, and the degree to which the space around the ego-vehicle is voxelized. We train models using the original [10] architecture, shown in Figure 1, creating "coarse" pillars with dimension $0.3125\text{m} \times 0.3125\text{m}$ at the base, and "fine" pillars with dimension $0.25\text{m} \times 0.25\text{m}$ at the base. The recurrent hidden dimension, corresponding to c and h in Figure 3, is 64 for the coarse model and 128 for the fine model. Both of these factors affect the accuracy and speed of the network.

Table 1 displays the results of each model on the car and pedestrian classes in the nuScenes validation set at both coarse and fine voxelization levels. Our model compares favorably to PP-3: both use the same point cloud to generate their predictions, and our model successfully leverages its recurrent memory to increase pedestrian mAP from 31% to 52% in the coarse case, and from 51% to 57% in the fine case. The car category sees a more modest improvement in the coarse case, from 66% to 67%. We observe a decline of 4% mAP for car detection in the fine case.

Our model compares favorably with PP-10. In the coarse case, it is significantly more accurate than PP-10 in detecting pedestrians, while producing only slightly less accurate predictions for cars. In the fine case, PP-REC trails PP-10 for both categories, as does PP-3. We provide precision-recall curves for the pedestrian class at the distance threshold of 4 meters in Figure 5. The distance threshold measures the bounding box center distance between prediction and ground

Object Detection mAP (%)					
		Coarse		Fine	
	# of Frames	Car	Pedestrian	Car	Pedestrian
PP-10	10	**69.43**	44.27	**74.68**	**60.26**
PP-3	3	65.79	30.51	71.64	50.82
PP-REC (Ours)	3	67.04	**52.46**	67.97	56.87

Table 1: *Comparing mean average precision of different models. mAP is averaged across the four matching thresholds described in [16]. Note how PP-REC strictly outperforms PP-3 in the coarse case.*

Speed (Hz) and Memory Usage (GB)				
	Coarse		Fine	
	Speed	Memory	Speed	Memory
PP-10	116.08	1.17	88.97	1.35
PP-3	115.49	1.17	71.64	1.34
PP-REC (Ours)	32.98	1.82	22.38	2.39

Table 2: *Speed and memory usage of each model at run time.*

truth, measured in meters on the ground plane [16]. Note how PP-REC maintains higher positive predictive value as the recall rate increases compared to PP-3, specifically, precision remains around 0.9 as recall increases to 0.4 for PP-REC, while precision declines to about 0.55 for PP-3 at the same recall; this explains the reason PP-REC outperforms PP-3 in pedestrian detection mAP.

These results echo the findings of [1], which noted that using a recurrent approach leads to a more significant improvement when applied to a smaller model, Tiny-YOLO, than to a larger model, Mixed-YOLO. Our results, taken in conjunction with the results from [1], show that recurrent models are viable and sometimes preferable alternatives to fully connected models for object detection with temporal data. This serves to show that the conclusion of [18] for NLP and generic sequence modelling tasks, that fully connected models are better than recurrent models, does not necessarily hold for 3D object detection with LiDAR. We hypothesize that this difference is due to the different domains of 3D object detection and the sequence modelling studied in [18].

Table 2 shows the speed of all models on a system with an Intel i5-9600K CPU and an NVidia Titan RTX GPU. PP-REC runs slower and uses more memory than PP-3 and PP-10, although this is to be expected when adding another layer to the CNN backbone. All models run faster than the capture rate of the LiDAR sensor used in the nuScenes [16] dataset, and the coarse models run $1.3 - 1.4\times$ faster than the fine models [19].

4. CONCLUSION

In this paper, we show that recurrent networks are a viable solution to the latency issue and data-hungry nature of temporally-aware feed-forward object detection networks.

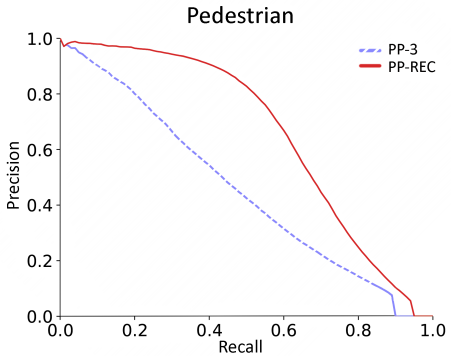

Fig. 5: *PR curves for the coarse models of PP-REC (red) and PP-3 (striped light blue) on the pedestrian class at a matching distance threshold of 4 meters.*

We demonstrate comparable detection accuracy to the original PointPillars work, while effectively using one third of the data. We also demonstrate an increase in detection accuracy of pedestrians in a coarse representation of 3D space, along with an analysis of the effects of changing the level of voxelization in the networks.

In [18], the authors conclude that simple feed-forward networks outperform recurrent architectures for sequence modelling. That work focuses on natural language applications, though, and contradicts the findings of [8, 1] as well as ours. We speculate that their finding does not transfer to object detection in neither the 2D nor 3D domains.

In future work, we plan to develop detection frameworks which leverage multi-sensor fusion in a recurrent fashion. We are also interested in RGB-based recurrent object detection networks which take advantage of sequential data, by generating more accurate monocular depth estimations and 2D video object detection.

5. REFERENCES

[1] Ahmad El Sallab, Ibrahim Sobh, Mahmoud Zidan, Mohamed Zahran, and Sherif Abdelkarim, "Yolo4d: A spatio-temporal approach for real-time multi-object detection and classification from lidar point clouds," *Neural Information Processing Systems (NeurIPS) Workshop MLITS*, 2018.

[2] Zhixin Wang and Kui Jia, "Frustum convnet: Sliding frustums to aggregate local point-wise features for amodal 3d object detection," *arXiv preprint arXiv:1903.01864*, 2019.

[3] Wenjie Luo, Bin Yang, and Raquel Urtasun, "Fast and furious: Real time end-to-end 3d detection, tracking and motion forecasting with a single convolutional net," *CVPR*, 2018.

[4] Charles R. Qi, Li Yi, Hao Su, and Leonidas J. Guibas, "Pointnet++: Deep hierarchical feature learning on point sets in a metric space," *Neural Information Processing Systems (NeurIPS)*, 2017.

[5] Gregory P. Meyer, Jake Charland, Darshan Hegde, Ankit Laddha, and Carlos Vallespi-Gonzalez, "Sensor fusion for joint 3d object detection and semantic segmentation," *arXiv preprint arXiv:1904.11466*, 2019.

[6] Waleed Ali, Sherif Abdelkarim, Mohamed Zahran, Mahmoud Zidan, and Ahmad El Sallab, "Yolo3d: End-to-end real-time 3d oriented object bounding box detection from lidar point cloud," *arXiv preprint arXiv:1808.02350*, 2018.

[7] Ming Liang, Bin Yang, Shenlong Wang, and Raquel Urtasun, "Deep continuous fusion for multi-sensor 3d object detection," *European Conference on Computer Vision (ECCV)*, 2018.

[8] Mason Liu, Menglong Zhu, Marie White, Yinxiao Li, and Dmitry Kalenichenko, "Looking fast and slow: Memory-guided mobile video object detection," *arXiv preprint arXiv:1903.10172*, 2019.

[9] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas, "Pointnet: Deep learning on point sets for 3d classification and segmentation," *CVPR*, 2017.

[10] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom, "Pointpillars: Fast encoders for object detection from point clouds," *CVPR*, 2019.

[11] Yin Zhou and Oncel Tuzel, "Voxelnet: End-to-end learning for point cloud based 3d object detection," *arXiv preprint arXiv:1711.06396*, 2017.

[12] Charles R. Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015.

[14] Ross Girshick, "Fast r-cnn," *arXiv preprint arXiv:1504.08083*, 2015.

[15] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi, "You only look once: Unified, real-time object detection," *arXiv preprint arXiv:1506.02640*, 2015.

[16] Holger Caesar, Varun Bankiti, Alex H. Lang, Sourabh Vora, Venice Erin Liong, Qiang Xu, Anush Krishnan, Yu Pan, Giancarlo Baldan, and Oscar Beijbom, "nuscenes: A multimodal dataset for autonomous driving," *arXiv preprint arXiv:1903.11027*, 2019.

[17] Xingjian Shi, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai kin Wong, and Wang chun Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," *Neural Information Processing Systems (NeurIPS)*, 2015.

[18] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," *arXiv preprint arXiv:1803.01271*, 2018.

[19] Yan Yan, "Second for kitti object detection," *GitHub repository*, 2019.