Towards Precise Vehicle-Free Point Cloud Mapping: An On-Vehicle System with Deep Vehicle Detection and Tracking

Mengdan Feng*, Sixing Hu†, Gim Hee Lee‡ and Marcelo Ang§
* Mechanical Engineering
National University of Singapore, Singapore
fengmengdan@u.nus.edu
† Computer Science
National University of Singapore, Singapore
hu.sixing@u.nus.edu
‡ Computer Science
National University of Singapore, Singapore
dcslgh@nus.edu.sg
§ Mechanical Engineering
National University of Singapore, Singapore
mpeangh@nus.edu.sg

***Abstract*—While 3D LiDAR has become a common practice for more and more autonomous driving systems, precise 3D mapping and robust localization is of great importance. However, current 3D map is always noisy and unreliable due to the existence of moving objects, leading to worse localization. In this paper, we propose a general vehicle-free point cloud mapping framework for better on-vehicle localization. For each laser scan, vehicle points are detected, tracked and then removed. Simultaneously, 3D map is reconstructed by registering each vehicle-free laser scan to global coordinate based on GPS/INS data.**

Instead of direct 3D object detection from point cloud, we first detect vehicles from RGB images using the proposed YVDN (YOLOv2 Vehicle Detection Network). In case of false or missing detection, which may result in the existence of vehicles in the map, we propose the K-Frames forward-backward object tracking algorithm to link detection from neighborhood images. Laser scan points falling into the detected bounding boxes are then removed. We conduct our experiments on the Oxford RobotCar Dataset [1] and show the qualitative results to validate the feasibility of our vehicle-free 3D mapping system. Besides, our vehicle-free mapping system can be generalized to any autonomous driving system equipped with LiDAR, camera and/or GPS.

***Index Terms*—Vehicle-free 3D mapping; Point Cloud; object detection; YOLOv2; Lucas-Kanade tracker**

I. Introduction

Precise 2D/3D map from LiDAR plays an important role in the localization and navigation of autonomous vehicles. Nowadays, more and more autonomous driving systems are equipped with 3D LiDAR due to its higher resolution and precision. Most existing LiDAR-based localization algorithms use local features to match with the projected 2D grid map [2], [3] or global 3D map [4]. Thus, a good map should contain rich and stable feature points about surrounding environments without redundancy. However, during data collection for mapping, moving objects (refer to objects that are moving or going to move away within a short time), such as person, vehicles, bicycles, etc., appear regularly, which causes the existence of unstable feature points in the map. As a result, localization is affected due to false laser scan matching.

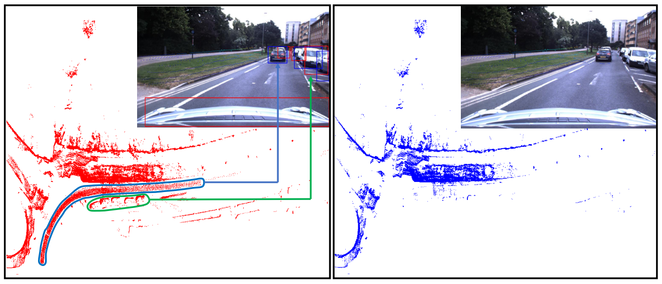
Fig. 1. Sample 3D mapping results with/without vehicles from a section of the route. Left: full 3D Point Cloud map. Blue contours: point cloud trajectory of the arrow-pointed moving vehicles in the image. Green contours: LiDAR points from pointed vehicles in the image. Right: vehicle-free 3D map.

In this paper, we propose a precise vehicle-free 3D point cloud mapping framework using deep object detection network and our proposed object tracking algorithm. Fig. 1 shows an example of our vehicle-free point cloud map. The main idea is to detect moving objects from each laser scan and remove these points. Then point cloud reconstruction can be done either by registering each clean laser scan data to the global coordinate using global poses from GPS/INS or transforming each vehicle-free laser scan to existing point cloud using relative poses from odometer, such as visual odometry [5] or wheel encoders. In our experiments, we use global poses from INS for point cloud registration since the Oxford RobotCar Dataset contains the GPS/INS data.

3D object detection and tracking remain challenging due to the irregular shape of 3D point cloud and its computational complexity. Most deep learning methods for 3D object detection discretize point cloud to voxel grid [6] or encode point cloud to 2D maps [7], [8] for feature learning and bounding box regression. However, compared with 3D object detection, state-of-the-art 2D object detectors using deep networks are faster and more accurate [9], [10]. Therefore, we make use of 2D object detectors in our work. We also propose a tracking method to avoid missing and false detection from one frame. Motivated by Frustum PointNet [11] which fuses information from RGB-D images for 3D object detection, we utilize the information from both the LiDAR and the camera for vehicle-free 3D mapping.

The main contributions of our work are: 1) We propose a general framework for precise vehicle-free 3D point cloud mapping for autonomous driving systems. 2) Based on YOLOv2 [9], we modify and train our YOLOv2 Vehicle Detection Network (YVDN) to achieve robust vehicle detection. 3) We propose the K-Frames forward-backward object tracking algorithm for missing and false detection. 4) We provide qualitative results on the Oxford RobotCar Dataset to validate the feasibility of our vehicle-free 3D mapping systems.

II. Related work

Vehicle-free point cloud mapping requires vehicle detection from point cloud and simultaneously mapping. Tracking is necessary due to missing or false detection. Over the past few years, a number of algorithms for object detection and tracking on RGB images or point clouds have been proposed, especially with the recent breakthroughs of deep networks.

3D Object Detection from Point Cloud: Earlier work on 3D object detection from point cloud usually includes three steps: removing ground points, clustering remaining points and labeling each cluster [12]–[14]. [13] proposes hierarchical segmentation and classifies each segment using different bag-of-word classifiers. Rather than segmentation, [14] chooses sliding window approach on each discretized 3D grid feature for SVM classification.

With the success of convolutional networks on 3D object detection, different networks [6], [8], [11], [15], [16] are proposed to achieve better performances on popular benchmarks. Most existing methods represent point cloud as voxel grid or project points to images for deep feature extraction. [15] is an earlier method of 3D volumetric CNN for landing zone detection. VoxNet [6] extends the binary detection to a general object detection task. However, direct 3D CNN is memory consuming and less efficient. MV3D [8] achieves faster and more accurate object detection through fusion of LiDAR and images. By projecting LiDAR data to bird's eye view, MV3D learns 3D region proposals and fuses ROI features from LiDAR views and images to jointly predict object class and 3D location. Considering the lag of MV3D, Frustum PointNets [11] uses 2D object detectors on RGB images to generate frustum proposals for 3D bounding boxes regression.

Nevertheless, object detection on RGB images are generally faster and more accurate than on point cloud. Thus, we detect objects on images and associate them to LiDAR data to filter moving objects.

2D Object Detection from RGB Images: There are mainly two categories on the state-of-the-art 2D object detection methods: region proposal based detection and grid-based detection. Region proposal methods select proper proposals and classify each proposal. The most common region proposal methods are R-CNN series methods [17]–[19]. R-CNN [17] uses selective search for region proposals and labels each proposal with SVM classifier. Fast R-CNN [18] learns the region proposals and jointly processes all the ROIs to accelerate training. Faster R-CNN [19] replaces hand-crafted region proposals with region proposal networks and merges Fast R-CNN for detection, which achieves state-of-the-art accuracy. On the contrary, grid-based detectors discretize the input to grids and predict grid label and object locations, such as YOLO [20], YOLOv2 [9] and SSD [10]. These methods propose real-time, end-to-end networks for grid-wise class probabilities and bounding box positions over the entire image. We use YOLOv2 as our basic network model due to its efficiency and accuracy.

Object Tracking Object detection methods cannot guarantee that target objects always be detected in continuous frames, which leads to the existence of moving objects in the final map. Thus, multiple object tracking (MOT) is of crucial importance for the generation of vehicle-free clean 3D map and better localization performance.

Most existing MOT methods can be divided into two categories, one is detection-based tracking, which first detects target objects in each frames and link objects in each frame by tracking, such as [21], [22]. The other is detection-free tracking which requires manual initialization of objects and joint prediction of object locations and correspondences, such as [23]. Some deep learning methods are also proposed [24]–[26]. In this paper, we adopt the tracking-by-detection method since we get detection from state-of-the-art objectors and use our K-Frames forward-backward algorithm for object tracking.

III. Approach

Our framework aims to build a vehicle-free point cloud map for better localization from a sequence of images and LiDAR frames as well as the pose of each frame. LiDAR frames are captured by LiDAR sensor mounted on the top of the vehicle and image sequences are recorded by the on-vehicle camera. The camera and LiDAR sensor have different capture frequency. We first introduce the pipeline of our framework in section III-A. Then in section III-B, III-C and III-D, we introduce our proposed methods and technical details of our framework.

A. Framework pipeline

Our framework takes images from the camera, 3D points from LiDAR sensor and GPS/INS information as input. The result is the vehicle-free 3D point cloud map. For each image frame, the nearest LiDAR frame is found to be the

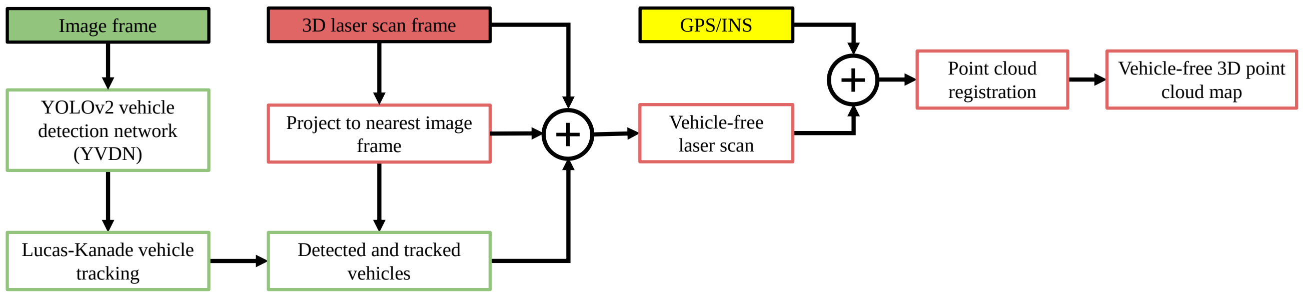
Image frame
3D laser scan frame
GPS/INS
YOLOv2 vehicle detection network (YVDN)
Project to nearest image frame
Vehicle-free laser scan
Point cloud registration
Vehicle-free 3D point cloud map
Lucas-Kanade vehicle tracking
Detected and tracked vehicles

Fig. 2. Proposed framework: vehicle detection and tracking are performed first for each laser scan, then vehicle-free 3D point cloud map is reconstructed.

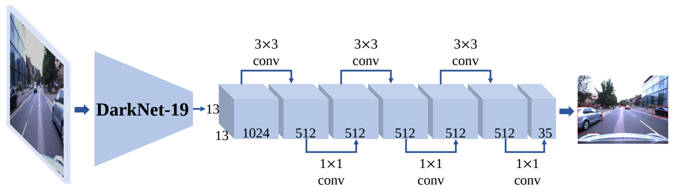

Fig. 3. The architecture of YOLOv2 Vehicle Detection Network.

corresponding LiDAR frame. We use our YOLOv2 Vehicle Detection Network (section III-B) to find the bounding boxes of vehicles. Then our K-Frames forward-backward bounding box tracking algorithm (section III-C) is applied to find the missing bounding boxes. Meanwhile, 3D points from LiDAR sensor is projected to the image frame through geometric transformation. The 3D points falling into the bounding boxes are regarded as vehicle points and are removed from the LiDAR frame. With the pose information from GPS/INS, all vehicle-free LiDAR frames are transformed to the first frame which is appointed as world coordinate system. The details of transformation is in section III-D. The overall pipeline of our proposed framework is illustrated in Fig. 2.

B. *YOLOv2 Vehicle Detection Network (YVDN)*

YOLOv2 [9] is the state-of-art, real time and end-to-end object detection model. We propose the YOLOv2 vehicle detection network (YVDN) based on the YOLOv2 network architecture. The structure of our proposed YVDN is illustrated in Fig. 3. YVDN has 24 convolutional layers where first 18 convolutional layers are pre-trained for classification while last 6 convolutional layers are subsequently trained for classification and bounding box detection. The pre-trained classification network is the same as the Darknet-19 [20] excluding the last convolutional layer. Before end-to-end training, the classification network is trained on ImageNet [27] for classification task. For further feature extraction and fusion, on top of the 18 convolutional layers, we add three 3×3 convolutional layers, each of which is followed by a 1×1 convolutional layer. All those convolutional layers have 512 filters except the last layer. The output of the last layer is the prediction of object positions and the corresponding labels. Each vector of the final output tensor contains anchor box information of the corresponding receptive field of the input image. The set of receptive fields of last layer constitutes the grid on the input image and each receptive field is a cell of the grid. For each cell, we predict 5 anchor boxes with seven parameters, i.e. x, y, w, h, $confidence$ and two class labels, i.e. $vehicle$ and $non\text{-}vehicle$, per anchor boxes. Thus, the number of filters of output layer is 5×7, which is in total 35 filters.

C. *K-Frames Forward-Backward Vehicle Tracking*

As can be seen from Fig. 6 (2-4th columns), our YVDN network fails to detect all vehicles on some frames. In case of missing detection and false detection from YVDN, we propose the K-Frames forward-backward bounding box tracking algorithm to track vehicles. Our K-Frames forward-backward bounding box tracking algorithm is based on Lucas-Kanade method.

The tracking of two consecutive image frames of our proposed algorithm has four steps: (1) For the image frame I_i, we detect the feature points $\{p_1, \cdots, p_n\}$ within each ROI (the region within the detected bounding box from YVDN) using the Features from Accelerated Segment Test (FAST) algorithm [28]. (2) We track these feature points in subsequent image frame I_{i+1} using pyramid Lucas-Kanade method. The tracked points $\{p'_1, \cdots, p'_n\}$ in image I_{i+1} have confidence score to indicate the tracking quality between p_j and p'_j, $j = 1, \cdots, n$. (3) The Random Sample Consensus (RANSAC) algorithm [29] is applied to find robust geometric transformation between feature points $\{p_1, \cdots, p_n\}$ and $\{p'_1, \cdots, p'_n\}$ based on the confidence score between each pair of feature point and tracked point. (4) Each bounding box in image I_i is tracked to image I_{i+1} using the geometric transformation matrix from step 3.

To apply the tracking algorithm on all image frames, there are two basic ways, to track the bounding box only from the immediate neighboring frames or track the bounding box from all the other frames. However, these two ways have drawbacks. For tracking from the immediate neighbors, the vehicle will still fail to be detected if it is not detected on more than two consecutive frames. For tracking from all the other frames, there will be overwhelming accumulated error from false positive detection. Ideally, the bounding boxes which are not successfully detected in several frames should be tracked from all the other frames, while a few false detected bounding boxes should be discarded immediately.

To balance the missing detection and false detection, we propose a K-Frames forward-backward bounding box tracking algorithm. For the image frame I_i, we track all the bounding boxes of I_i forward from the image I_{i+1} to the image I_{i+K-1}. Reversely, the bounding boxes of I_i are back tracked from the image I_{i-1} to the image I_{i-K+1}. Thus, each of the K images contain all the detected and tracked bounding boxes. To avoid the spread of false detection, we only obtain the bounding boxes of the image I_{i+K} from the tracking from I_{i+1} to I_{i+K-1} without the image I_i. Finally the non-maximum suppression algorithm is applied to remove bounding boxes with large overlap. As can be seen from the algorithm, a larger K will make it track more missing bounding boxes but will make it bring more false bounding boxes. A smaller K is the opposite situation. The two basic ways applying tracking algorithm mentioned above is two extreme cases of our K-Frames forward-backward bounding box tracking algorithm. When $K = 2$, the proposed algorithm becomes tracking the bounding box only from the immediate neighboring frames. When K is the number of all frames, it becomes tracking the bounding box from all the other frames. In our application scenario, through the experiments, we observe that $K = 5$ provides best performance to track missing detection and to avoid false detection.

D. 3D Point Cloud Mapping

To get the vehicle-free laser scan by removing the LiDAR points according to the image points in the bounding boxes, we need to know the correspondence between the 3D points and 2D image pixels. To project 3D points to the corresponding image, the transformation is first from LIDAR coordinate system to camera coordinate system and then to the image coordinate system. The whole procedure is pre-determinant after calibration on the LiDAR sensor and the camera. The homogeneous coordinate of 3D point from LIDAR sensor is denoted as $P_L = (X, Y, Z, 1)$. The homogeneous coordinate of 2D point on the image is denoted as $p = (x, y, 1)$. After the calibration, the intrinsic parameter matrix of the camera K and the displacement matrices of two sensors $R_{L->c}$ and $t_{L->c}$ are known. The transformation from P_L to p is

$$p = K[R_{L->c}|t_{L->c}]P_L \tag{1}$$

The points in different LiDAR frames are in different coordinate systems. To build a 3D point cloud map of the entire street, these points need to be transformed into a world coordinate system. We take the coordinate system of the first frame of the LiDAR coordinate system as the world coordinate system. The GPS/INS information is known. Therefore, the displacement, i.e. $R_{L->L_0}$ and $t_{L->L_0}$, from current LiDAR coordinate system to world coordinate system are known. The homogeneous coordinate of 3D point in world coordinate is denoted as P_W. The transformation from P_L to P_W is

$$P_W = [R_{L->L_0}|t_{L->L_0}]P_L. \tag{2}$$

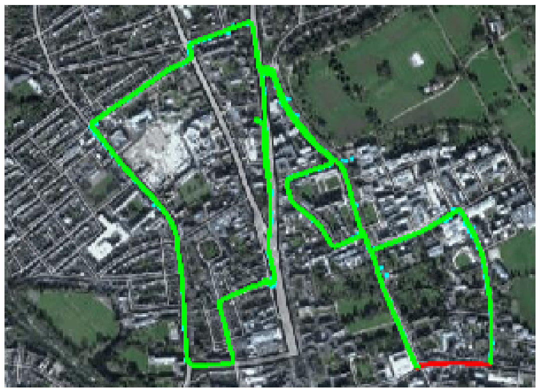
Fig. 4. We select one of the primary route from Oxford RobotCar Dataset. Blue: traversal path with good GPS. Red: pool GPS.

IV. Experiment

In this section, we illustrate our experimental results in three parts. We first describe the Oxford RobotCar Dataset [1] where we conduct our experiments. Next we present our vehicle detection and tracking system and the qualitative results on Oxford RobotCar Dataset. Finally we demonstrate the vehicle removal process from each laser scan and the final vehicle-free point cloud map.

A. Oxford RobotCar Dataset

The Oxford RobotCar Dataset contains a large amount of images, laser scan and GPS/INS data. These data are collected along the same route for many times in central Oxford under different weather and illumination conditions. To test our vehicle detection and tracking system as well as mapping algorithm, we pick one path collected in an overcast weather from 09:53:12 am, June 26, 2014, covering almost the full 10km route. The satellite map and vehicle path are visualized in Fig. 4. The images we use are collected from the Point Grey Bumblebee XB3 camera, with image resolution $1280 \times 960 \times 3$. The 3D laser scans are collected from the SICK LD-MRS 3D LiDAR, with 4 planes only. The route we choose contains 19102 image frames and 5801 laser scans.

B. Vehicle Detection and Tracking

For the training of our YVDN, we use the pre-trained high-resolution model (448×448 input) to initialize the Darknet-19. Then we train the whole network for vehicle detection on COCO dataset. Equipped with one NVIDIA TITAN X GPU, we train the network for 30 epochs with the initial learning rate 5×10^{-3}, weight decay 0.0001 and momentum 0.95 using Stochastic Gradient Descent. During inference, the classification threshold is set as 0.25. Sample results of the vehicle detection from our YVDN are visualized in Fig.5. As can be seen, the network is robust to lighting changes, large overlaps and small objects that are far away.

Once our YVDN predicts the vehicle positions for each image frame, we use our K-Frames forward-backward tracking algorithm for vehicle tracking. First, FAST [28] key-point detector is applied on each bounding box due to its efficiency

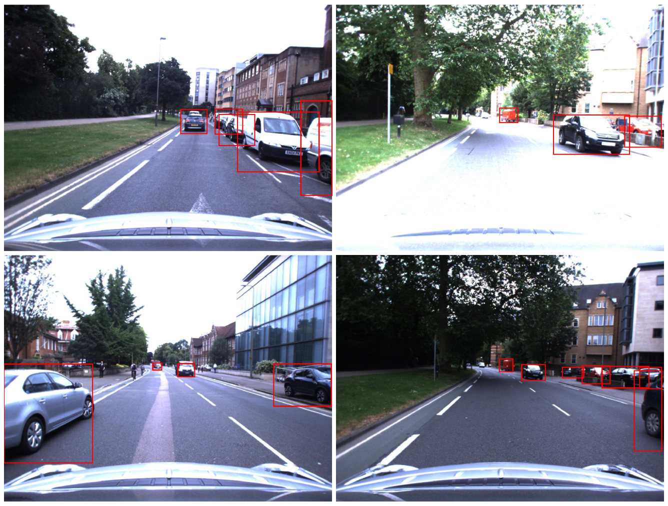

Fig. 5. Sample vehicle detection results from our YVDN on Oxford RobotCar Dataset. Red boxes: detected vehicles. As we can see, the detector is robust to occlusion and illumination.

and robustness. Second, the key-points are tracked using our K-Frames forward-backward bounding tracking algorithm. We choose $K = 5$. Third, after we get the pair of corresponding points between different images, we estimate the affine transformation between the matched point pairs using RANSAC algorithm [29]. Thus, we successfully track the bounding boxes from previous image to current image. Finally, we remove bounding boxes if the overlapping ratio is large than a threshold. We set the ratio threshold 0.7 in our experiment. Some tracking results can be seen from Fig. 6. The results show that our tracking algorithm can successfully track the missing detected bounding boxes.

C. *Precise Vehicle-Free 3D Point Cloud Mapping*

We filter each laser scan by removing vehicles from the point cloud before reconstructing the point cloud. For each laser scan, we first project the 3D points to its nearest image frame according to the time stamp. Then we remove the points whose projection to the image falling into the detected and tracked bounding boxes. Thus, we get vehicle-free laser scan. Fig. 7 shows two examples of vehicle removal from laser scan.

For vehicle-free 3D point cloud mapping, we first interpolate the INS data to get the global pose for each laser scan according to their time stamps. Next we register each vehicle-free laser scan to the global coordinate using geometric transformations. We select around 200-meters path from the whole route and reconstruct the 3D map with and without vehicles. Fig. 8 shows the full point cloud model of the scene before and after vehicle removal. We can see that the vehicles can be successfully removed from the 3D map. It validates the feasibility and robustness of our proposed algorithm.

V. Conclusion

In this paper, we propose a general on-vehicle system for precise vehicle-free 3D point cloud mapping using a sequence of laser scans from a LiDAR sensor and corresponding RGB images from a camera. The vehicle-free 3D map only contains points from static background without points from moving objects, i.e. vehicles in our experiment. It is supposed to make contribution on more robust and accurate LiDAR-based vehicle localization. Our proposed YVDN and K-Frames forward-backward object tracking algorithm is able to detect vehicles robustly. The results on the Oxford RobotCar Dataset indicates that our proposed on-vehicle system can remove points belonging to vehicles and build a precise vehicle-free 3D point cloud map. In our future work, we will continue working on the LiDAR-based localization using the vehicle-free 3D point cloud map or converted 2D grid map.

VI. Acknowledgment

This research was supported by the National Research Foundation (NRF) Singapore through the Singapore-MIT Alliance for Research and Technology's (FM IRG) research programme. We are grateful for the support.

References

[1] W. Maddern, G. Pascoe, C. Linegar, and P. Newman, "1 year, 1000 km: The oxford robotcar dataset," *The International Journal of Robotics Research*, vol. 36, no. 1, pp. 3–15, 2017.
[2] F. Dellaert, D. Fox, W. Burgard, and S. Thrun, "Monte carlo localization for mobile robots," in *IEEE International Conference on Robotics and Automation*, May 1999.
[3] D. Fox, "Kld-sampling: Adaptive particle filters," in *Advances in neural information processing systems*, 2002, pp. 713–720.
[4] J. M. Bedkowski and T. Röhling, "Online 3d lidar monte carlo localization with gpu acceleration," *Industrial Robot: An International Journal*, vol. 44, no. 4, pp. 442–456, 2017.
[5] D. Nistr, O. Naroditsky, and J. Bergen, "Visual odometry," in *Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 1. Ieee, 2004, pp. I–I.
[6] D. Maturana and S. Scherer, "Voxnet: A 3d convolutional neural network for real-time object recognition," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2015, pp. 922–928.
[7] H. Su, S. Maji, E. Kalogerakis, and E. Learned-Miller, "Multi-view convolutional neural networks for 3d shape recognition," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 945–953.
[8] X. Chen, H. Ma, J. Wan, B. Li, and T. Xia, "Multi-view 3d object detection network for autonomous driving," in *The IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, no. 2, 2017, p. 3.
[9] J. Redmon and A. Farhadi, "Yolo9000: Better, faster, stronger," in *IEEE Conference on Computer Vision and Pattern Recognition*, July 2017.
[10] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "Ssd: Single shot multibox detector," in *European conference on computer vision*. Springer, 2016, pp. 21–37.
[11] C. R. Qi, W. Liu, C. Wu, H. Su, and L. J. Guibas, "Frustum pointnets for 3d object detection from rgb-d data," *arXiv preprint arXiv:1711.08488*, 2017.
[12] F. Moosmann, O. Pink, and C. Stiller, "Segmentation of 3d lidar data in non-flat urban environments using a local convexity criterion," in *Intelligent Vehicles Symposium, 2009 IEEE*. IEEE, 2009, pp. 215–220.
[13] J. Behley, V. Steinhage, and A. B. Cremers, "Laser-based segment classification using a mixture of bag-of-words," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*. IEEE, 2013, pp. 4195–4200.
[14] D. Z. Wang and I. Posner, "Voting for voting in online point cloud object detection." in *Robotics: Science and Systems*, vol. 1, 2015, p. 5.
[15] D. Maturana and S. Scherer, "3d convolutional neural networks for landing zone detection from lidar," in *Robotics and Automation (ICRA), 2015 IEEE International Conference on*. IEEE, 2015, pp. 3471–3478.
[16] B. Li, T. Zhang, and T. Xia, "Vehicle detection from 3d lidar using fully convolutional network," in *Proceedings of Robotics: Science and Systems*, AnnArbor, Michigan, June 2016.

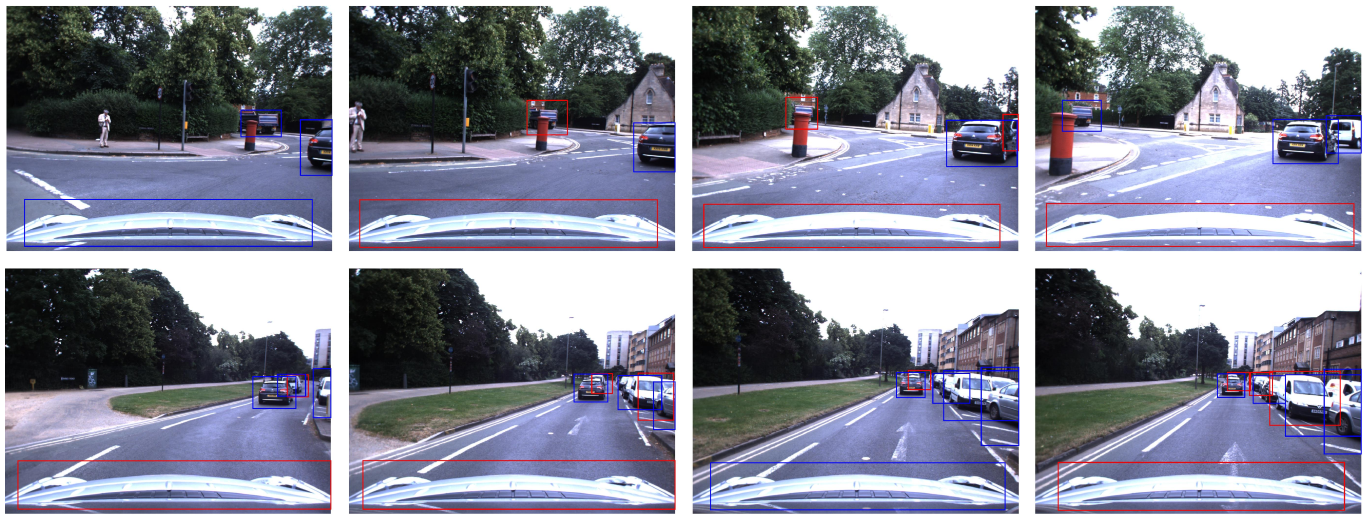

Fig. 6. Sample vehicle tracking results: each row indicates the tracking results in continuous frames. Blue boxes: detected vehicles. Red boxes: tracked vehicles. As can be seen, missing detection can be successfully tracked.

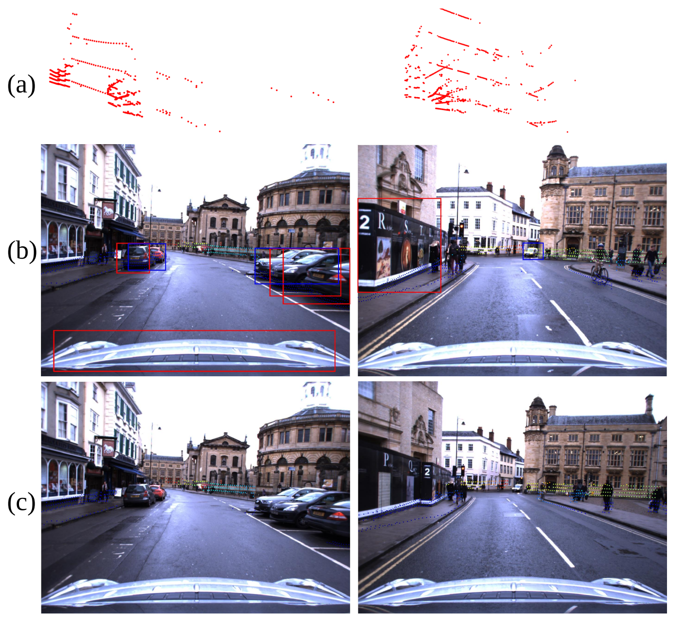

Fig. 7. Examples of vehicle removal from each laser scan. Each column represents an example vehicle removal pipeline. (a) Single laser scan frame; (b) Laser points projection to image and vehicle detection and tracking on image; (c) Vehicle points removal for vehicle-free laser scan.

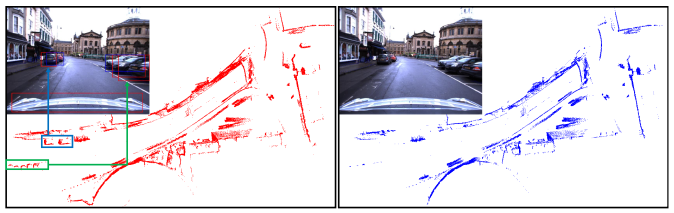

Fig. 8. Sample 3D mapping results with/without vehicles. The left image shows a full point cloud map. Green and blue boxes represent vehicles in the corner image. The right image shows the vehicle-free point cloud map.

[17] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Computer Vision and Pattern Recognition*, 2014.

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

[19] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," in *Advances in neural information processing systems*, 2015, pp. 91–99.

[20] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788.

[21] J. Shi *et al.*, "Good features to track," in *IEEE Conference on Computer Vision and Pattern Recognition*. IEEE, 1994, pp. 593–600.

[22] B. Song, T.-Y. Jeng, E. Staudt, and A. K. Roy-Chowdhury, "A stochastic graph evolution framework for robust multi-target tracking," in *European Conference on Computer Vision*. Springer, 2010, pp. 605–619.

[23] Z. Kalal, K. Mikolajczyk, and J. Matas, "Tracking-learning-detection," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 7, pp. 1409–1422, 2012.

[24] Y. Xiang, A. Alahi, and S. Savarese, "Learning to track: Online multi-object tracking by decision making," in *2015 IEEE international conference on computer vision (ICCV)*, no. EPFL-CONF-230283. IEEE, 2015, pp. 4705–4713.

[25] M. Keuper, S. Tang, Y. Zhongjie, B. Andres, T. Brox, and B. Schiele, "A multi-cut formulation for joint segmentation and tracking of multiple objects," *arXiv preprint arXiv:1607.06317*, 2016.

[26] A. Sadeghian, A. Alahi, and S. Savarese, "Tracking the untrackable: Learning to track multiple cues with long-term dependencies," in *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017.

[27] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International Journal of Computer Vision*, vol. 115, no. 3, pp. 211–252, 2015.

[28] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in *European conference on computer vision*. Springer, 2006, pp. 430–443.

[29] M. A. Fischler and R. C. Bolles, "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography," in *Readings in computer vision*. Elsevier, 1987, pp. 726–740.