Dynamic Registration: Joint Ego Motion Estimation and 3D Moving Object Detection in Dynamic Environment

Wenyu Li\textsuperscript{1}, Xinyu Zhang\textsuperscript{2,*}, Zijun Wang\textsuperscript{1}, Shichun Guo\textsuperscript{2}, Nan Qiu\textsuperscript{2} and Jun Li\textsuperscript{2}

Abstract—Localization in a dynamic environment suffers from moving objects. Removing dynamic object is crucial in this situation but become tricky when ego-motion is coupled. In this paper, instead of proposing a new slam framework, we aim at a more general strategy for a localization scenario. In that case, Dynamic Registration is available for integrating with any lidar slam system. We utilize 3D object detection to obtain potential moving objects and remove them temporarily. Then we proposed Dynamic Registration, to iteratively estimate ego-motion and segment moving objects until no static object generates. Static objects are merged with the environment. Finally, we successfully segment dynamic objects, static environments with static objects, and ego-motion estimation in a dynamic environment. We evaluate the performance of our proposed method on KITTI Tracking datasets. Results show stable and consistent improvements based on other classical registration algorithms.

I. INTRODUCTION

Simultaneous Localization and Mapping (SLAM) estimating the robot pose while mapping its surroundings of an unknown environment. Most SLAM systems assume static environments. However, the real world contains dynamic objects. Localization or ego-motion estimation in a dynamic environment is important for applications such as robot navigation and autonomous driving tasks. SLAM algorithms can easily fail in dynamic environments, so moving objects need to be detected and removed.

Most articles \cite{[1]}–\cite{[3]} viewed localization in a dynamic environment from two different perspectives: either as a robust SLAM or as an extension of SLAM. The former regards dynamic elements as outliers and remove them from the estimated process \cite{[4]}, whereas the latter extends SLAM to detect and track the dynamic objects \cite{[5]}.

Robot systems require object information of the surroundings for decision making and motion planning. In autonomous driving scenarios, the car not only needs to localize itself but to perceive other cars or pedestrians. Dynamic scenes are crucial in real environments. Thus it’s necessary to pay attention to all objects and their motion state, to facilitate segmenting and tracking these instances, rather than focusing on segmenting and inferencing points that are probably in moving.

![Fig. 1: Removing objects using 3D object detection in different stages. We use red, green and blue represent cars, cyclists and pedestrians respectively. Image in Fig. 1(d) is only utilized for visibility.](image)

These are the start points of the paper: localize in dynamic environments and detect full objects. The intuition is to directly sense dynamic objects and remove them. But when ego-robot moves, motion segmentation become implicit. However, this effect has little influence on object detection. When obtained objects previously, ego-motion estimation and moving objects detection in dynamic environments seem feasible.

In consideration of above all, we propose Dynamic Registration. The remainder of this paper is organized as follows. In Section II, we review literature in related research areas. In Section III, we describe the proposed Dynamic Registration. In Section IV, we introduce the experimental details and evaluated results. Finally, we conclude this work and suggest future extensions in Section V.

II. RELATED WORK

A. Moving Object Detection

Moving object detection or segmentation has been widely researched in the computer vision community in the past \cite{[6]}. Whereas here we only focus on lidar-based methods. Generally, it’s hard for MOD methods to segment complete moving objects in dynamic scenes due to the existing coupling of ego-motion, which threatens complete geometric representations of objects.

Map based: \cite{[7]} builds an occupancy map as a filter to remove dynamic points. \cite{[8]} performs map points classification based on visibility assumptions for the sake of lifelong map
Motion based: Motion based methods detect moving objects without prior informations, but may fail to segment all moving objects limited by merely leveraging motion clues. And they are unable to detect stationary objects that have the potential to move [9]. End to end: Chen et al. [10] projects point cloud to range view and then generates residual images as inputs of a segmentation network. But it simply divided the point clouds into moving and static. [11] segments points using two secutive frames semantically.

However, these methods miss the concept of instance. They tend to segment a point with its state of motion and semantic label rather than to detect a car or a pedestrian with its size, location, orientation and, its motion state. That is to say, we may get some moving points representing a half car, which makes it hard to decide the motion state of the object.

B. Localization in Dynamic Environment

Most SLAM systems designed for dynamic environments are visual SLAM [12], [13]. 2D object detection is usually implemented in a robot SLAM system [14]. One important reason is, it’s easier to deploy and accelerate a visual detection model than in lidar point clouds. However, recent advanced technics in 3D object detection make it possible to meet the need for both speed and accuracy in point cloud [15], [16]. We introduce them in Section III-A.1.

Wang et al. [17] first proposed a Bayesian-based theory of Simultaneous Localization, Mapping and Moving Object Tracking(SLAMMOT). From then on, SLAM and the Detection And Tracking of Moving Objects (DATMO) can be combined and have been proved to be beneficial to each other. Paralleling SLAM and MOT could estimate both ego-motion and object motion Simultaneously. [18] implement a joint estimation of localization and tracking. To emphasize the capability of tracking, they adopt segmentation rather than detection. [19] project 3D point cloud into a 2D plane, and adopt a FCN network to segment objects. But classification and clustering need to be taken as post-processing.

Our Dynamic Registration system consists of three modules, as illustrated in Fig. 2. The first module performs a 3D object detection neural network, generating 3D bounding boxes for all instances. The second module temporarily removes all objects from the input point cloud. And then, an initial pose can be drawn by classical scan match methods. In the third module, we transform objects in current frames into previous frames in Birds Eye View(BEV) by the estimated poses. And a motion state estimation module is employed to segment static and dynamic objects. Points of static objects and environments merge to generate a new environment. Then the merged static environment is used to estimate the ego-pose and state of objects iteratively until no objects are considered static. Finally, the static environment, ego-motion, and dynamic objects, these three coupling individuals, are calculated and estimated successfully.

A. Object Detection And Removal

1) 3D Object Detection: We aim to remove moving object points from a registration pipeline and segment these points as instance-level objects. Previous works in segmenting or detecting moving elements use either a semantic segmentation or a binary moving segmentation since the heavy computations and complexities of instance segmentation in a 3D point cloud.

These 2D segmented methods are not enough to describe objects in 3D space. Further post-processing like clustering needs to be implemented in terms of instances.

The balance of real-time and accuracy need to be considered. 3D Object Detection predicts 3D bounding boxes for objects in a single frame point cloud. However, compared with 3D instance segmentation [20], 3D object detection
achieves almost the same instance-level performances due to the sparsity of point clouds.

VoxelNet [21] achieve end-to-end detection using Voxel Feature Extractor, but introduced 3D convolutions. SECOND [15] speeds up 3D convolutions via sparse optimization. PointPillars [16] proposes pillar representation to improve the efficiency. In order to implement real-time detections in scenes, we adopt PointPillars to realize 3D Object Detection. The output of 3D object detection at time \( t \) with \( n \) objects is defined as

\[
D_t \triangleq \{ d_0, d_1, \ldots, d_n \}
\]

\[
d_n \triangleq \{ x, y, z, l, w, h, \theta, \text{label}, \text{score} \}
\]

where \( x, y, z \) the center point coordinate of a cuboid; \( l, w, h \) the length, width, height of the cuboid, both in meters, respectively; \( \theta \) the heading angle around z-axis in degrees. In addition to these cuboid parameters, \( \text{label} \) represents three classes including car, cyclist, and pedestrian, and \( \text{score} \) denotes a confidence score for detection result.

\[
T_0 = \text{Registration}(M_{t-1}, M_t)
\]

where \( T_0 \in SE(3) \).

2) Removing Points In Cuboid: Supposing a cuboid can be formulated as three vectors, as shown in Fig. 3, which are \( AC, AB, AD \), respectively. We use the dot product to decide whether a point is located in the cuboid.

Let \( P \) be a point in 3D space, if

\[
0 \leq \frac{\overrightarrow{AP} \cdot \overrightarrow{AB}}{||\overrightarrow{AB}||} \leq \frac{\overrightarrow{AB} \cdot \overrightarrow{AB}}{||\overrightarrow{AB}||}
\]

then \( P \) is fixed at \( \pm 90^\circ \) around \( \overrightarrow{AB} \) axis, and its length is within \( ||\overrightarrow{AB}|| \). In the same way, the point \( P \) can be limited in the cuboid. This procedure can be expressed as:

\[
0 \leq \frac{\overrightarrow{AP} \cdot \overrightarrow{AB}}{||\overrightarrow{AB}||} \leq \frac{\overrightarrow{AB} \cdot \overrightarrow{AB}}{||\overrightarrow{AB}||}
\]

\[
0 \leq \frac{\overrightarrow{AD} \cdot \overrightarrow{AD}}{||\overrightarrow{AD}||} \leq \frac{\overrightarrow{AD} \cdot \overrightarrow{AD}}{||\overrightarrow{AD}||}
\]

\[
0 \leq \frac{\overrightarrow{AC} \cdot \overrightarrow{AC}}{||\overrightarrow{AC}||} \leq \frac{\overrightarrow{AC} \cdot \overrightarrow{AC}}{||\overrightarrow{AC}||}
\]

We first remove all potential moving objects temporarily because static objects will be added to the environment after motion segmentation. Fig. 1(a) and Fig. 1(b) show the removing results.

B. Object Reprojection

1) Initial Pose Estimation: We first remove all points in the detected cuboids without checking their motion state. After that, static and dynamic objects are both removed. Then an initial pose can be estimated by a registration algorithm using the clean point clouds.

2) Reprojection: Detection results \( D_t \) at time \( t \) is projected to time \( t-1 \) using estimated transformation matrix \( T_0 \).

\[
D_t^{t-1} = T_0 D_t
\]

Objects are detected differently in two frames. It theoretically remains the same detections in such a short time interval. However, affected by the uncertainty of observations and inferences, it may occur noisy detections, false positives, false negatives, and occlusions which result in different results for each frame.
C. Dynamic Registration

1) Motion segmentation: Object reprojection errors can be drawn subsequently from associated detections according to the initial pose. Here the reprojection errors represent moving distances of objects in two frames, so detected objects can be segmented by a threshold. Then dynamic environment can be divided into moving objects and static objects.

**Algorithm 1: Dynamic Registration**

**Input:** Input Point Clouds: \( M_{t-1}, M_t \)

**Output:** Dynamic Registration Pose: \( T_{tDR} \)

C.1) Segmented Dynamic Objects: \( D_{t-1}^d, D_t^d \)

1. \( (D_{t-1}, D_t) \leftarrow 3D \) Object Detection \( (M_{t-1}, M_t) \)

2. \( (M_{RA}^R, M_{RA}^A) \leftarrow \) RemoveAll \( (M_{t-1}, D_{t-1}, M_t, D_t) \)

3. \( T_0^S \leftarrow \) Registration \( (M_{RA}^R, M_{RA}^A) \)

4. \( D_{t-1}^S \leftarrow \) Object Reprojection \( (T_0^S, D_{t-1}, D_t) \)

5. \( (D_{t-1}, D_{t-1}^S, D_t^S) \leftarrow \) MotionSegmentation \( (D_{t-1}, D_{t-1}^S) \)

6. \( (M_{RD}^R, M_{RD}^D) \leftarrow \) MergeAndConcatenate \( (D_{t-1}, D_t^S, M_{RA}^R, M_{RA}^A) \)

7. \( T_{tDR} \leftarrow \) Registration \( (M_{tDR}^R, M_{tDR}^D) \)

2) Iterative Dynamic Registration: In most cases, removing all objects from point clouds helps improve registration accuracy. However, static objects always occur in highly changeling real-world environments, e.g., parked cars. Static objects also provide rich features. There is no direct intuition for keeping them from the registration pipeline.

Since static objects have been segmented, they can be added to the environment. When newly stationary points are merging and concatenating, a registration algorithm, as mentioned in Section III-B.1, is employed again for a more reliable transformation matrix. This dynamic registration process is iteratively conducted until no static objects are generated. After that, static environment, dynamic objects, and ego-motion are all fully segmented out. Any mapping, tracking, or localization algorithm may be utilized for post-processing.

We formulate the iterative dynamic registration process as Algorithm 1: Dynamic Registration, Algorithm 2: Iterative Dynamic Registration, respectively. Lower left \( t \) represent time stamp. Upper left 0 and * represent initial results and optimized results. Upper right symbols \( RA, RD, s, d \) represent Remove All objects, Remove Dynamic objects, static and dynamic respectively.

IV. EXPERIMENTS AND RESULTS

To better evaluate the performance of our proposed dynamic registration algorithm, we adopt KITTI Tracking datasets [24] as a benchmark for the experiment since it provides both ego and object information.

**Algorithm 2: Iterative Dynamic Registration**

**Input:** Point Clouds: \( M_{t-1}^{RA}, M_t^{RA} \)

Detection Results: \( D_{t-1}, D_t \)

Initial Registration Pose: \( T_0^{RD} \)

Dynamic Registration Pose: \( T_{tDR}^{RD} \)

**Output:** Iterated Registration Pose: \( T_t^{RD} \)

C.2) Iterated Dynamic Objects: \( D_{t-1}^{s+}, D_t^{d+} \)

C.3) Iterated Static Objects: \( D_{t-1}^{s*}, D_t^{d*} \)

1. \( 0 D_{t-1}^{s*} \leftarrow \) Object Reprojection \( (T_0^{RD}, D_{t-1}, D_t) \)

2. \( (0 D_{t-1}^{d*}, 0 D_t^{d*}, 0 D_{t-1}^{s*}, 0 D_t^{s*}) \leftarrow \) Motion Segmentation \( (D_{t-1}^{s*}, 0 D_{t-1}^{d*}) \)

3. \( D_{t-1}^{s*} \leftarrow \) Object Reprojection \( (T_{tDR}^{RD}, D_{t-1}, D_t) \)

4. \( (* D_{t-1}^{d*}, * D_t^{d*}, * D_{t-1}^{s*}, * D_t^{s*}) \leftarrow \) Motion Segmentation \( (D_{t-1}^{s*}, * D_{t-1}^{d*}) \)

5. while \( * D_{t-1}^{d*} \neq D_{t-1}^{d*} \) and \( * D_t^{d*} \neq D_t^{d*} \) do

6. \( (* M_{RD}^{R}, * M_{RD}^{D}) \leftarrow \)

7. \( T_t^{RD} \leftarrow \) Registration \( (* M_{RD}^{R}, * M_{RD}^{D}) \)

8. \( * D_{t-1}^{d*} \leftarrow * D_{t-1}^{d*} \) and \( * D_t^{d*} \leftarrow * D_t^{d*} \)

9. \( * D_{t-1}^{s*} \leftarrow \) Object Reprojection \( (T_t^{RD}, D_{t-1}, D_t) \)

10. \( (* D_{t-1}^{s*}, * D_t^{s*}, * D_{t-1}^{d*}, * D_t^{d*}) \leftarrow \) MotionSegmentation \( (D_{t-1}^{s*}, * D_{t-1}^{d*}) \)

end

A. Implementation Details

As discussed in Section III-A.1, we leverage a pillars-based 3D object detection network, PointPillars, for detecting objects in KITTI Tracking sequences.

We directly use a version provided by MMDetection3D [25], without any fine-tuning or editions. The model is pre-trained on KITTI 3D detection benchmark. Detailed settings and the pre-trained model can be found in their repository.

3D object detection on KITTI dataset traditionally follows a range of \([-0.7,0,-0.4,0.4,-3.1]\)). However, it doesn’t perform well in a registration scenario. Thus we limit the point cloud range to \([-0.4,0.4,-3.3,0,-3.1]\)) in \(x, y, z\) axis respectively. Subsequently, results out of range are filtered out. We also select cuboids with a confidence threshold higher than 0.5, in order to grab quality detections.

Additionally, to improve the efficiency and accuracy of the registration algorithm, we downsample each input point cloud with a 0.3m voxel.

B. Quantitative Experiments

1) Evaluation Metrics: We adopt relative pose error (RPE) to test the performances of our proposed registration algorithm. Absolute trajectory error (ATE) may not be suitable in a scan matching scenario since a slight error in a transformation matrix at the beginning of a sequence leads to a large gap in two trajectories subsequently. For simplicity, here we only consider the translational part of RPE:

\[
RPE_{trans} = \sqrt{\frac{1}{N-1} \sum_{i=1}^{N-1} \| \text{trans} \left( T_{gt}^{-1} T_{est} \right) \|_2} \tag{6}
\]
where $T_{gt}$ and $T_{est}$ are ground truth pose and estimated pose between two frames, respectively.

2) Results: The results are shown in Table I, which details comparisons of our method’s performance against other traditional registration algorithms. We build our Dynamic Registration based on two classical methods, NDT and ICP. Results match intuitive expectations on average. Dynamic Registration performs better in almost all sequences, since removing dynamic objects follows the static world assumption.

Results show our approach outperforms other registration algorithms, even better than removing all potential moving objects. Removing all objects without checking their motion state also works when objects all keep moving. However, this coarse strategy increases localization error because fewer static points are engaged to the registration pipeline. Adding static points to the environment decreases localization uncertainty in a dynamic and complex environment.

C. Qualitative Evaluation

1) Odometry And Mapping: We also implement a pairwise structureless odometry based on continuous point cloud registration simply on visualization grounds. Since other factor makes it inexplicit to evaluate the effects of removing dynamic objects. Results are shown in Fig. 5. Here Dynamic Registration is build on NDT, corresponding to NDT-RMD in Table I. Dynamic Registration Odometry is closer to the ground truth trajectory than NDT odometry.

Mapping results indicate that our approach successfully removes dynamic objects. Red boxes enlarge objects’ trajectories. Obvious ghostings are cleared, but fragment remains(purple box in Fig. 5(c)), mainly due to false negatives of the detector. However, false positives matter little, since they are possibly segmented as static objects and then added to the environment.

2) Moving Object Detection: In most cases, moving objects can be segmented successfully. But motion segmentation in dynamic registration relates to 3D object detection, ego-motion estimation, and the preset segmentation threshold. Low-quality detection results occur when objects are too far from the ego car, which leads to false positives or false negatives. These “wrong detections” are filtered out by data association, as mentioned in Section III-B.2. But data association merely solved the issue partially, since false negatives remain until state estimation and prediction are conducted. Detect objects in point cloud sequences (also regarded as multi object tracking) is beyond the discussion scope of this paper.

V. Conclusions

In this paper, we proposed dynamic registration, combining ego motion estimation and moving object detection in a point cloud registration pipeline. 3D object detection is utilized to detect objects previously and removed temporally. Then it begins to iteratively estimate ego motion and add static objects until no static objects are segmented. We evaluate our proposed method on KITTI benchmark and compare it with different registration algorithms, which demonstrates its effectiveness. We’ll future extend dynamic registration to a dynamic slam system.

| seq | Registration | Dynamic Registration |
|-----|--------------|----------------------|
|     | NDT | ICP | NDT-RMA | ICP-RMA | NDT-RMD | ICP-RMD |
| 0000 | 0.2583 | 0.3641 | 0.2646 | 0.3887 | 0.2101 | 0.3444 |
| 0001 | 0.3453 | 0.4550 | 0.3506 | 0.4793 | 0.2689 | 0.4050 |
| 0002 | 0.4560 | 0.5723 | 0.4718 | 0.5861 | 0.4479 | 0.5696 |
| 0003 | 0.7364 | 0.7952 | 0.7523 | 0.7869 | 0.7151 | 0.7769 |
| 0004 | 0.8349 | 0.7200 | 0.8509 | 0.7353 | 0.8127 | 0.6874 |
| 0005 | 0.8709 | 0.8537 | 0.8773 | 0.8643 | 0.8545 | 0.8301 |
| 0006 | 0.3247 | 0.4664 | 0.3190 | 0.4651 | 0.2912 | 0.4496 |
| 0007 | 0.3650 | 0.4428 | 0.3566 | 0.4601 | 0.3195 | 0.4095 |
| 0008 | 0.7415 | 0.7842 | 0.7361 | 0.7859 | 0.7102 | 0.7589 |
| 0009 | 0.4218 | 0.5042 | 0.4432 | 0.5230 | 0.3979 | 0.4613 |
| 0010 | 0.9602 | 0.8938 | 0.9185 | 0.8978 | 0.9407 | 0.8667 |
| 0011 | 0.3978 | 0.5209 | 0.4137 | 0.5629 | 0.3722 | 0.5111 |
| 0012 | 0.1161 | 0.1432 | 0.0856 | 0.1490 | 0.0955 | 0.1473 |
| 0013 | 0.3404 | 0.4283 | 0.3488 | 0.4262 | 0.2861 | 0.3735 |
| 0014 | 0.1874 | 0.4625 | 0.1939 | 0.5032 | 0.1876 | 0.4653 |
| 0015 | 0.3067 | 0.3625 | 0.3161 | 0.3705 | 0.2894 | 0.3571 |
| 0016 | 0.1917 | 0.1136 | 0.1341 | 0.1145 | 0.1102 | 0.1159 |
| 0017 | 0.1259 | 0.1598 | 0.1514 | 0.1658 | 0.0801 | 0.1617 |
| 0018 | 0.2791 | 0.3746 | 0.2759 | 0.3857 | 0.2788 | 0.3074 |
| 0019 | 0.2154 | 0.2977 | 0.2162 | 0.3054 | 0.1954 | 0.2797 |
| 0020 | 0.1874 | 0.7300 | 0.5098 | 0.7938 | 0.4526 | 0.7138 |

mean | 0.4268 | 0.4974 | 0.4279 | 0.5120 | 0.3960 | 0.4758 |

REFERENCES

[1] M. R. U. Saputra, A. Markham, and N. Trigoni, “Visual SLAM and Structure from Motion in Dynamic Environments: A Survey,” ACM Computing Surveys, vol. 51, no. 2, pp. 1–36, Jun. 2018.
[2] B. Bescos, C. Campos, J. D. Tardós, and J. Neira, “DynaSLAM II: Tightly-coupled multi-object tracking and SLAM,” IEEE Robotics and Automation Letters, vol. 6, no. 3, pp. 5191–5198, 2021.
[3] J. Zhang, M. Henein, R. Mahony, and V. Ila, “Vdo-slam: A visual dynamic object-aware slam system,” arXiv preprint arXiv:2005.11052, 2020.
of the environment.

Registration. In Fig. 5(b) and Fig. 5(c), red box zoom-in details of NDT, and the red ones are trajectories of proposed Dynamic Registration results of Dynamic Registration. In Fig. 5(a), dashed curves represent ground truth poses. Blue solid lines are trajectories of 0013 among KITTI Tracking Datasets. In Fig. 5(a), dashed curves of traj.

(c) Odometry and mapping results of Dynamic Registration

Fig. 5: Comparisons of the global trajectories for sample sequence 0013 among KITTI Tracking Datasets. In Fig. 5(a), dashed curves represent ground truth poses. Blue solid lines are trajectories of NDT, and the red ones are trajectories of proposed Dynamic Registration. In Fig. 5(b) and Fig. 5(c), red box zoom-in details of the environment.

[4] R. Mur-Artal and J. D. Tardós, “Orb-slam2: An open-source slam system for monocular, stereo, and rgb-d cameras,” IEEE Transactions on Robotics, vol. 33, no. 5, pp. 1255–1262, 2017.

[5] C.-C. Wang, C. Thorpe, and S. Thrun, “Online simultaneous localization and mapping with detection and tracking of moving objects: Theory and results from a ground vehicle in crowded urban areas,” in 2003 IEEE International Conference on Robotics and Automation (ICRA), vol. 1, 2003, pp. 842–849.

[6] M. Siam, H. Mahgoub, M. Zahran, S. Yogamani, M. Jagersand, and A. El-Sallab, “MODNet: Motion and Appearance based Moving Object Detection Network for Autonomous Driving,” in 2018 21st International Conference on Intelligent Transportation Systems (ITSC), Nov. 2018, pp. 2859–2864.

[7] S. Pagad, D. Agarwal, S. Narayanan, K. Rangan, H. Kim, and G. Valla, “Robust Method for Removing Dynamic Objects from Point Clouds,” in 2020 IEEE International Conference on Robotics and Automation (ICRA), May 2020, pp. 10 765–10 771.

[8] F. Pomerleau, P. Krúsi, F. Colas, P. Furgale, and R. Siegwart, “Long-term 3D map maintenance in dynamic environments,” in 2014 IEEE International Conference on Robotics and Automation (ICRA). IEEE, 2014, pp. 3712–3719.

[9] A. Dewan, T. Caselitz, G. D. Tipaldi, and W. Burgard, “Motion-based detection and tracking in 3D LiDAR scans,” in 2016 IEEE International Conference on Robotics and Automation (ICRA), May 2016, pp. 4508–4513.

[10] X. Chen, S. Li, B. Mersch, L. Wiesmann, J. Gall, J. Behley, and C. Stachniss, “Moving Object Segmentation in 3D LiDAR Data: A Learning-Based Approach Exploiting Sequential Data,” IEEE Robotics and Automation Letters, vol. 6, no. 4, pp. 6529–6536, Oct. 2021.

[11] Y. Sun, W. Zuo, H. Huang, P. Cai, and M. Liu, “PointMoSeg: Sparse Tensor-Based End-to-End Moving-Obstacle Segmentation in 3-D Lidar Point Clouds for Autonomous Driving,” IEEE Robotics and Automation Letters, vol. 6, no. 2, pp. 510–517, Apr. 2021.

[12] B. Bescos, J. M. Fácil, J. Civera, and J. Neira, “DynaSLAM: Tracking, mapping, and inpainting in dynamic scenes,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4076–4083, 2018.

[13] C. Yu, Z. Liu, X.-J. Liu, F. Xie, Y. Yang, Q. Wei, and Q. Fei, “DS-SLAM: A semantic visual SLAM towards dynamic environments,” in 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, pp. 1168–1174.

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

[15] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “PointPillars: Fast Encoders for Object Detection From Point Clouds,” in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). Jun. 2019, pp. 12 689–12 697.

[16] C.-C. Wang, C. Thorpe, S. Thrun, M. Hebert, and H. Durrant-Whyte, “Simultaneous Localization, Mapping and Moving Object Tracking,” The International Journal of Robotics Research, vol. 26, no. 9, pp. 889–916, Sep. 2007.

[17] F. Moosmann and C. Stiller, “Joint self-localization and tracking of generic objects in 3D range data,” in 2013 IEEE International Conference on Robotics and Automation (ICRA), May 2013, pp. 1146–1152.

[18] Z. Wang, W. Li, Y. Shen, and B. Cai, “4-D SLAM: An Efficient Dynamic Bayes Network-Based Approach for Dynamic Scene Understanding,” IEEE Access, vol. 8, pp. 219 996–220 014, 2020.

[19] D. Zhou, J. Fang, X. Song, L. Liu, J. Yin, Y. Dai, H. Li, and R. Yang, “Joint 3D Instance Segmentation and Object Detection for Autonomous Driving,” in 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2020, pp. 1836–1846.

[20] Y. Zhou and O. Tuzel, “VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection,” in 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition, Jun. 2018, pp. 4490–4499.

[21] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in 2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016, pp. 3464–3468.

[22] X. Weng, J. Wang, D. Held, and K. Kitani, “3d multi-object tracking: A baseline and new evaluation metrics,” in 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Jun. 2019, pp. 12 689–12 697.

[23] A. Bewley, Z. Ge, L. Ott, F. Ramos, and B. Upcroft, “Simple online and realtime tracking,” in 2016 IEEE International Conference on Image Processing (ICIP). IEEE, 2016, pp. 3464–3468.

[24] B. Bescos, J. M. Facíl, J. Civera, and J. Neira, “DynaSLAM: Tracking, mapping and inpainting in dynamic scenes,” IEEE Robotics and Automation Letters, vol. 3, no. 4, pp. 4076–4083, 2018.

[25] M. Contributors, “MMDetection3D: OpenMMLab next-generation platform for general 3D object detection,” 2020.