Lightweight Map-Enhanced 3D Object Detection and Tracking for Autonomous Driving

Lei Gong, Shunhong Wang, Yu Zhang*, Yanyong Zhang, Jianmin Ji
School of Computer Science and Technology
University of Science and Technology of China
Hefei, China
Email:{gleisa19,wangshunhong}@mail.ustc.edu.cn,*yuzhang@ustc.edu.cn

ABSTRACT

3D object detection and tracking are crucial to the real-time and accurate perception of the surrounding environment for autonomous driving. Recent approaches on 3D object detection and tracking have made great progress, thanks to the rapid development of deep learning models. Even though these models have achieved superior performance on specific datasets, the actual self-driving systems still cannot deal with real-world driving situations properly, especially in complicated scenarios like road intersections. With the development of vehicle-infrastructure cooperation technology, scene information such as map is considered to have great potential in alleviating these problems. In this paper, we explore the potential of solving corner cases in real driving scenarios through the cooperation between autonomous vehicles and map information. We propose a holistic approach that integrates and utilizes the map information in system following the tracking-by-detection paradigm. In order to ensure that the use of map information does not bring much overhead to detection and tracking, we propose a representation method for concise information extracted from rich map. We show that our framework can improve the detection and tracking accuracy with mild or no increase of latency. Specifically, in some cases, our results demonstrate a MOTA improvement of nearly 2% .

KEYWORDS

tracking-by-detection; multi-object tracking (MOT); autonomous driving; map information; unified framework

ACM Reference Format:
Lei Gong, Shunhong Wang, Yu Zhang*, Yanyong Zhang, Jianmin Ji. 2020. Lightweight Map-Enhanced 3D Object Detection and Tracking for Autonomous Driving . In *12th Asia-Pacific Symposium on Internetware (Internetware'20), May 12–14, 2021, Singapore, Singapore.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3457913.3457941

1 INTRODUCTION

Due to the technological progress and the development of deep learning, automated driving is under intensified development by industry and academia since the last decade. As an important part of the intelligent vehicle, environmental perception system mainly refers to the detection of the surrounding environment of the vehicle through on-board sensors. Object detection is used to detect the static or dynamic targets in environment, while tracking is responsible for generating tracklets of objects. The estimated trajectory and the kinematic information of targets are particularly vital for predicting the future movement [14].

Tracking-by-detection is one of the most popular multi-object tracking (MOT) paradigm. A common tracking-by-detection system, for example the AB3DMOT [23], usually relies on an object detector for getting the bounding box of objects in the surrounding environment. Filter models such as Kalman Filter [12] can be used to predict and update the state of tracked targets, while association methods such as Hungarian algorithm can be used to find the correspondence between a collection of new measurements and preexisting tracks [6]. Although various detection [10, 18] and tracking frameworks [17, 23, 27] have been proposed and achieve great performance in real world driving scenarios, there are still many cases that are tough for a single autonomous vehicle system. For example, roadside cars and other static objects may be misidentified, or the behaviour of vehicles, cyclists, pedestrians is hard to predict.

With the development of V2X (Vehicle-to-Anything) and communication technologies, researchers explore solving corner cases by the cooperation between the vehicle and the roadside intelligent infrastructure. Specifications for road infrastructure have been formulated to support autonomous driving in recent years [3]. These specifications emphasize the huge role of scene information in the future development of autonomous driving, and the map is the basic scene information provided by road infrastructure.

However, we have noticed that almost none of the detection and tracking frameworks on the Open Driving datasets leaderboard consider the use of scene information like map, even though some of these datasets contain map data. A few works explore the preliminary use of lane centerlines or road geometric information and show the potential of map information in enhancing the performance and robustness of the perception systems. For example, Chang *et al.* realize the importance of map information and build Argoverse [4] dataset with rich map. Then they design a specific 'baseline' tracker which demonstrates that utilization of map information brings a better tracking performance. Therefore, whether additional information such as maps is needed to help improve the performance of the original tracking framework is a question worth exploring. Furthermore, maps vary from place to place, making it hard to standardise the representation. How to extract concise information

Internetware'20, May 12–14, 2021, Singapore, Singapore

ACM ISBN 978-1-4503-8819-1/20/11…$15.00
https://doi.org/10.1145/3457913.3457941

from different original data to improve perception accuracy and reduce time performance loss is also crucial.

To address the above challenge, in this paper, we explore the effective fusion method of combining map information with **general** tracking-by-detection framework. We propose a *Semantic-Geometric model* defining the concise information extracted from the map, which only contains necessary map information. We further design a universal fusion method based on this model, which makes it applicable to such MOT systems with few modifications.

The main contributions of this paper are as follows:

- We propose the method to integrate map information into the existing tracking-by-detection system to improve its performance.
- We propose a concise information representation method, which can be extracted from different map resources and contains the necessary semantic and geometric data required for driving scenarios.
- We propose a general algorithm that uses map information to improve accuracy. This algorithm can be used before or after the detection process, and only a few modifications to the original system are required.
- We prototype the proposed method and make a case study on typical 3D object tracking scenarios in the KITTI dataset [7]. Experimental results show that our method can reduce false positives in detection and achieve a MOTA (Multiple Object Tracking Accuracy) improvement.

2 BACKGROUND AND RELATED WORK

In this section, we introduce the related work on object detection and tracking in autonomous driving, as well as the work using map information in tracking systems.

2.1 3D Object Detection and Tracking

Autonomous vehicles nowadays are large and complex systems equipped with multiple sensors. The most common sensor, the camera, can provide rich appearance features but lack depth information to estimate the distance to other objects well, which is fatal to cars driving on highways. LiDAR, a relatively newer technology, can measure distance by using an array of light pulses, and can provide 3D point clouds for autonomous vehicles. It has become a mainstream method that doing the detection and tracking task based on 3D LiDAR point cloud.

3D Object Detection from Point Clouds. In recent years, many point-cloud-based object detection methods have emerged, which are usually divided into two types: voxelization based and projection based, according to how point cloud features are represented. Voxelization based methods such as SA-SSD [10] and Part A^2 [20], utilize a voxel grid to encode features. These methods suffer from the sparsity of point clouds and the huge computational cost of 3D convolution. Projection based methods such as PIXOR [26] and PointPillars [15], attempt to project the point cloud into a perspective view (*e.g.,* a bird's eye view) and then apply image-based feature extraction techniques. These methods have better time performance than voxel-based methods, but their projected features are usually insufficient for accurate object detection, especially for small targets.

3D Multi-Object Tracking (MOT). Existing 3D MOT approaches are mostly implemented based on tracking-by-detection paradigm, which makes a clear distinction between the detection and tracking of objects. For each frame, the general idea is to first localize objects of interest using an object detector, then associate detected objects between frames. For the tracking task, a common methodology is to split the tracking into two phases: prediction of object location, as well as matching of detections and predictions. AB3DMOT [23] uses Kalman filter [12] for motion prediction and Hungarian algorithm for data association to achieve real-time 3D multi-target tracking. JRMOT [17] optimally integrates information from 2D RGB images and 3D point clouds, then fuses them into a multi-modal feature to perform data association through re-identification. FanTrack [1] proposes an online MOT formulation using CNN for data association. GNN3DMOT [24] improves the performance of data association by introducing graph neural network into the new feature interaction mechanism.

2.2 Issues in MOT

False Detection. When using the tracking-by-detection framework, the precision of MOT is largely related to the accuracy of the detector. If the result given by the detector contains some wrong objects (which should not exist), the tracker would obviously give a wrong result with these wrong objects. These wrong objects are called *false positives* (FP). When the driving scenario is complicated enough, false positives will inevitably occur. For example, the laser reflected by the trees provides the detector with confusing point cloud data, which may cause detector mistakes. In order to reduce the impact caused by the false positives, previous works use tracking management to delay the output of an object by several frames. Only when the object exists in these frames, the object would appear in the tracking result. But this approach brings new problems.

False Negatives Related to the Tracking Management. The above tracking management attempts to lower the influence of false positives by delaying the output of the object by several frames. And lots of methods use this approach. For example, AB3DMOT uses a value called Bir_{min} to define the minimum matching frames before an object is output as a *truth positive* (TP), while both JRMOT and GNN3DMOT use a value called n_{init} to do the same thing. However, the tracking management would bring *false negative* (FN) to every object that really exists. Actually, the tracking management is based on the distrust of the detector results, and uses some false positives to reduce a large number of false negatives. This approach only works in scenarios with a few objects. For complicated driving scenarios, this approach may not be worth it. Anyway, false positives will bring additional losses to the precision of MOT.

Uncertainty Caused by the Driving Route. For many tracking approaches, it is a common way to use a motion model to predict the location of an object in the next frame. But if the object just appears for one frame, the tracker will get nothing helpful to predict the velocity of the object. Problems would arise when the scene changes rapidly. Without velocity information at the first detected frame, it can be predicted that the object would be at the same position in the next frame. When the object moves fast, the tracker will not

	Level	Name	Description	Digital information provided to AVs			
				Digital map with static road signs	VMS, warnings, incidents, weather	Microscopic traffic situation	Guidance: speed, gap, lane advice
Conventional infrastructure	E	Conventional infrastructure / no AV support	Conventional infrastructure without digital information. AVs need to recognise road geometry and road signs.				
	D	Static digital information / Map support	Digital map data is available with static road signs. Map data could be complemented by physical reference points (landmarks signs). Traffic lights, short term road works and VMS need to be recognized by AVs	X			
Digital infrastructure	C	Dynamic digital information	All dynamic and static infrastructure information is available in digital form and can be provided to AVs.	X	X		
	B	Cooperative perception	Infrastructure is capable of perceiving microscopic traffic situations and providing this data to AVs in realtime.	X	X	X	
	A	Cooperative driving	Based on the real-time information on vehicle movements, the infrastructure is able to guide AVs (groups of vehicles or single vehicles) in order to optimize the overall traffic flow.	X	X	X	X

Table 1: Levels of the Infrastructure Support for Automated Driving (ISA Levels)

be able to associate the same object in two adjacent frames based on motion model, which has a great impact on the performance of the MOT task. When we talk about this problem with autonomous vehicles, it's a heuristic idea that we can give every object an initial velocity. But due to the uncertainty caused by the driving route, the velocity of objects (mostly cars) also has great uncertainty. Therefore, approaches based on the motion model will not perform well when an object suddenly appears.

2.3 MOT with Map Information

Limited by vision field and height of sensors, there are always conner cases that cannot be handled by a single intelligent car.

Map can provide information contain geographic, geometric and semantic priors, that are useful for many tasks in autonomous vehicles [3, 11, 13]. Chang *et al.* build a dataset called Argoverse to support autonomous vehicle perception tasks including 3D tracking and motion forecasting [4]. This dataset contains a vector map of lane centerlines with attributes, a rasterized map of ground height, and a rasterized map of the drivable area and the region of interest. Then they construct a baseline tracking pipeline using these map attributes including the drivable area, ground removal, and lane direction. Their experimental results show that map information leads to slightly better detection and tracking performance.

Danzer *et al.* propose an approach adapting a multi-object tracking system to model interaction between vehicles and the current road geometry [5]. Due to the interaction between objects and the road geometry, the assumption of the Constant Turn Rate and Velocity (CTRV) model used to describe the vehicle dynamics may not meet reality. For this reason, they integrate the interaction between objects using the Intelligent Driver Model (IDM) [22] and road geometry information to improve the prediction step of the GM-LMB filter. Their main idea is to correct the wrong assumption of the dynamic model, *i.e.,* velocity and turn rate, adapting the prediction step of a GM-LMB filter.

In addition to map information, there are lots of other scene information. As shown in Table 1, the Infrastructure Support levels for Automated Driving (ISAD) is recently proposed by the INFRAMIX project [3]. They define five levels of infrastructure support for automated driving (ISA Levels) in order to categorize the various means of infrastructure elements and capabilities that support automated vehicles. The static digital information (map support) is the most basic factor defined in D to A. Starting from level C, the Digital infrastructure provides not only digital map, but also VMS (Variable Message Signs), weather, Microscopic traffic situation and so on.

2.4 Open Source Autonomous Driving Platforms

Autoware[1] and Baidu Apollo[2] are two prime open-source autonomous driving platforms designed to accelerate the development, testing, and deployment of autonomous vehicles. The complex architecture can be divided into several core sub-modules such as perception, localization, planning, and navigation connected by a middleware framework such as ROS (Robot Operating System) in Autoware.

The utilization of map is one of the core functions. Autoware uses the Vector Map provided by a commercial corporation before Version 1.12, since version 1.13, they replace it with Lanelet2 format [16]. However, as an open driving platform with complete

[1]https://github.com/Autoware-AI/autoware.ai
[2]https://github.com/ApolloAuto/apollo

functions, Autoware adopts a design closely coupled with the specific map format. As a result, developers have to make a lot of changes to the original relevant code. As for Baidu Apollo, it also only supports Apollo OpenDrive HDMap specification modified and extended from the standard OpenDrive[3]. Therefore, this tightly coupled design is not conducive for us to design a universal method to use multiple maps in tracking framework and integrate them into various driving platforms.

3 METHODOLOGY AND FRAMEWORK

Our goal is to design a universal fusion method to use map information in various tracking frameworks to improve performance such as accuracy. This method should meet a basic requirement, that is, it can be applied to different existing tracking systems, as well as different map formats and driving systems through lightweight modification.

We first introduce the detailed design goals of the methodology, and then present the design considerations.

3.1 Design Goals

According to our research questions and the core design requirement, we finally propose a concise map information representation model and its application method.

The detailed goals motivating the map-enhanced method design include:

G1. *Real-time*. Autonomous vehicle systems have high requirements for real-time processing. Therefore, this method should not bring more overhead to the original tracking system.

G2. *Universality*. This method of using map information should be applied to many different detection and tracking systems, and also adaptable to different map formats and driving systems.

G3. *Lightweight*. When this method is applied to a detection and tracking system, only a few modifications to the original system are required.

G4. *Pluggability*. This method should have lower coupling with the original tracking framework. It should be independent of the development of tracking frameworks.

G5. *Extensibility*. Considering the huge role of other information play in future work, this method should also be extensible to support more scene information in tracking systems.

3.2 Tracking-by-Detection MOT Framework

As can be seen from Section 2.1, the tracking-by-detection paradigm makes a clear distinction between detection and tracking. Besides, the tracking loop can generally be divided into several sub-modules. So we can easily add processing to improve accuracy using map information between these modules.

A tracking-by-detection framework generally contains Detection, Association, State Update, Track Management and Prediction modules as shown in Fig. 1. The function of each module is described as follows:

Detection. This module is responsible for detecting objects in the surrounding environment and providing bounding boxes of the desired objects.

Data Association. After receiving the new measurements, the Data Association module will find the most likely correspondence between the predicted state of the tracked targets and these new measurements.

State Update. Once the correspondences are calculated, the State Update module is performed to update the current state of the object.

Tracking Management. This tracking management module is used to determine when to create new tracks and when to delete obsolete tracks.

Prediction. This module is just used for tracking methods based on motion state. By using the motion state of an object in the previous frame, it predicts the possible state of the object in the current frame, then uses it to help the data association process.

Common tracking-by-detection systems usually have a similar framework with the above sub-modules. We position to support a large class of existing and future systems with such framework features, and choose the joint between the modules in the framework to insert the map-enhanced processing, so as to achieve the goals of universality (G2) and pluggability (G4) mentioned in Section 3.1.

In order to increase the extensibility (G5) of our method and adapt our method to different map formats and driving systems (G2), next we will discuss the designed concise scene information integrated on such a tracking-by-detection framework.

3.3 Scene Information to be Integrated

In order to achieve the real-time (G1) and universality (G2) goal in Section 3.1, we propose a concise representation of scene information and a universal method to access and use this information quickly. In this paper, considering the original map data may have various formats and redundant information we don't need, we gather and organize the necessary semantic and geometric data, such as road geometry, centerlines, traffic sign and so on. For example, different areas in the map could be represented by sequences of points that form the polygons of specific area. A line can also be a polyline segment with the coefficient of fitting curve. A traffic sign can be represented as a point. For each kind of map information, we will explore the uniform and most appropriate representation to reduce storage and processing overhead.

Different map information can be used in different modules. For the detection module, its performance has a great influence on the whole tracking system. We can consider using information such as no-driving zone, to filter FPs in the detection results or the input point cloud to enhance the performance of the original tracking system. For the association and prediction modules, the accuracy depends on the definition of motion model and prediction of behavior patterns of cars or pedestrians. For example, the velocity and turn rate of the car change rapidly in curve lane scenarios and the assumption of constant velocity model may not be satisfied and cause failure of tracking. The interaction between targets, the curvature of road lanes and the priors of human behavior can add constraints to the prediction range and improve the prediction performance.

In this paper, we mainly discuss the use of no-driving zones to reduce FPs of the detector, see Section 4 for details, and verify the effectiveness of the proposed method through this case study.

[3]https://www.asam.net/standards/detail/opendrive/

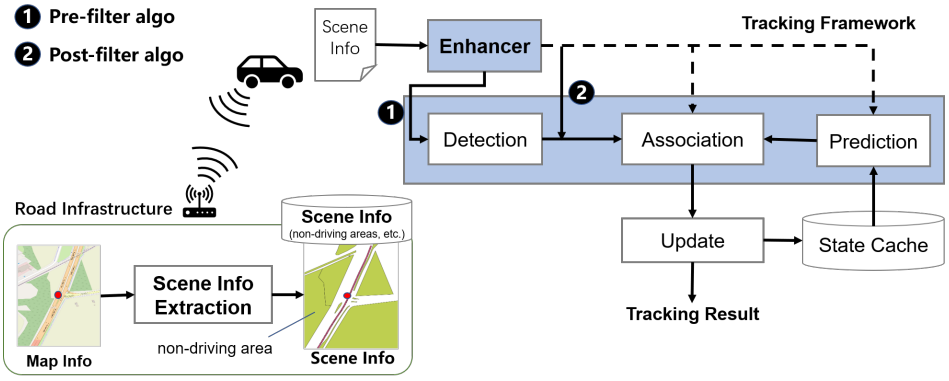

Figure 1: The Map-Enhanced tracking framework in our paper. We integrate map information into original tracking system and make use of different key information in different modules. The scene info including map information is concise representation of extracted semantic and geometric information from various data resources. The pre- and post- filter algorithm is designed for utilizing map information to improve detection performance.

3.4 Semantic-Geometric Model

We define the semantic representation for the concise information such as no-driving zones used in our fusion method, and introduce a ***Semantic-Geometric Map*** $\mathbb{M}:\mathbb{L} \rightarrow \mathbb{V}$. It is a mapping from semantic label class $\mathbb{L}$ to geometric value set class $\mathbb{V}$. The semantic label class $\mathbb{L}$ currently contains at least two labels, *i.e.,* $\mathsf{L}_{\mathsf{ndrv}}$ and $\mathsf{L}_{\mathsf{curve}}$, representing two kinds of no-driving zone and curve, respectively. The geometric value set class $\mathbb{V}$ contains the corresponding geometric definition, which is essentially a labeled sum type [9], *i.e.,*

$$\mathbb{V} = \mathsf{V}_{\mathsf{ndrv}} + \mathsf{V}_{\mathsf{curve}},$$

where each component of the sum type represents a specific geometric value type corresponding to a semantic label, *e.g.,* $\mathsf{V}_{\mathsf{ndrv}}$ for $\mathsf{L}_{\mathsf{ndrv}}$ and $\mathsf{V}_{\mathsf{curve}}$ for $\mathsf{L}_{\mathsf{curve}}$.

The geometric value set of the no-driving zone, $\mathsf{V}_{\mathsf{ndrv}}$, is a power set of the polygon set, *i.e.,* $\mathbb{P}$. We have

$$\mathbb{M}\left(\mathsf{L}_{\mathsf{ndrv}}\right) = \{P_i\} \in \mathsf{V}_{\mathsf{ndrv}}$$

where each element $P_i = (p_0, ..., p_n, p_0)$ is a finite sequence of points, enclosing a no-driving polygon, and each $p_j = (x_j, y_j)$ represents the coordinates of the j−th point.

Similarly, the geometric value set of the curve, $\mathsf{V}_{\mathsf{curve}}$, is also a curve set, *i.e.,* $\mathbb{C}$. We have

$$\mathbb{M}\left(\mathsf{L}_{\mathsf{curve}}\right) = \{C_i\} \in \mathsf{V}_{\mathsf{curve}}$$

where each element $C_i = (\check{P}_i, A_i)$ represents a curve. $\check{P}_i \in \check{\mathbb{P}}$ and A_i respectively represent the polyline segment corresponding to the i−th curve and coefficients of quadratic polynomial fitted by the points of the polyline segment. Both $\check{\mathbb{P}}$ and $\mathbb{P}$ are sets of point sequences, but the point sequence of the former is a polyline segment, while the point sequence of the latter forms a polygon. $A_i = (c_i, b_i, a_i)$ represents the constant coefficients of quadratic polynomial $F(x) = c_i + b_i x + a_i x^2$, which are obtained by fitting $(p_0, ..., p_n)$ by least square method, and $c_i = p_0(y)$.

Although we only define two kinds of scene information at present, the abstract representation of this semantic-geometric map can be extended to support more scene information according to actual needs in the future. Furthermore, the above map can be extracted offline from real map and stored on the roadside or edge side for real-time acquisition by the on-board system, as the Scene Info Extraction step shown in Figure 1.

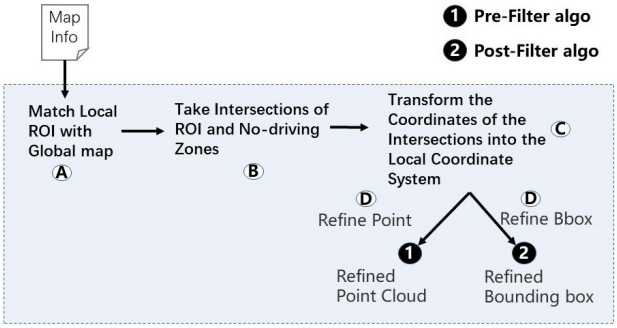

Figure 2: The filter algorithm in our prototype framework as shown in Fig. 1. This algorithm is designed for reducing the false positives by filtering the point cloud or bounding boxes within the no-driving zones provided in the map information.

3.5 Method Design

After giving the definition of scene information, we'd like to talk about how to use the map information defined by semantic-geometric model to satisfy the design goals listed in Section 3.1.

Our method is an incremental modification to the original tracking-by-detection framework. In order to make our method lightweight (G3) and pluggable (G4) to adapt different models or approaches with little modification, we abstract the operations as interfaces.

For each component of V corresponding to a semantic label, we define the corresponding operator set $Op(V)$, where each operator set represents a series of data processing operations.

For no-driving zone $\mathsf{V}_{\mathsf{ndrv}}$ defined in Section 3.4, we have

$$Op(\mathsf{V}_{\mathsf{ndrv}}) = \{Filter_{pre}, Filter_{post}\}$$

$Filter_{pre}$ and $Filter_{post}$ represent the filter algorithms illustrated in Fig. 2. The filter algorithms are defined to reduce false positives of the detection module. We abstract and summarize operations of each algorithm into four steps. Each step represents an abstract process, which does not depend on a specific implementation. After Localization, the first step is matching the local view with global map, which represents a fusion of local view map and global map. Steps B and C represent the processes to get the no-driving zone in local view. The final step D is to refine the points of point cloud or bounding boxes base on the no-driving zone information. The pre-filter algorithm is used to remove the points of the point cloud in the no-driving zones. The original tracking systems only need to call the points filter in data preprocessing progress. The post-filter algorithm has a similar process to the pre-filter algorithm, except that the last step is used to remove the bounding box of the objects in the no-driving zones.

Our modification to the original code is very lightweight, We hide the detailed implementations behind the simple interfaces, making our method more convenient to use and easier to expand.

In this paper, we focus on reducing false positives in the tracking results with the help of map information. To reflect the effect of our method, we did a case study and elaborated the detailed implementation behind the interface in Section 4.

4 CASE STUDY

In this section, we present our preliminary experiments on typical driving scenario and study the potential of map information in MOT.

Take log "2011_09_26_drive_0032" of the KITTI datasets as an example, it is a typical highway scenario, which means that the autonomous vehicle does not need to consider objects outside the road. However, there are lots of trees beside the road and our experiments show that the detector of tracking-by-detection MOT framework may wrongly recognize the tree as an object, thereby increasing the false positives. Therefore, we extract the semantic and geometric information from the OpenStreetMap [8] based on the GPS of car trajectory. OpenStreetMap is an open world map that provides free geographic data to everyone. And we hope to reduce false positives by the no-driving zone information in our framework. We will detail the process in the following subsections.

4.1 Map Information Extraction

The map information extraction step is an additional work independent of the whole framework. We extract all information we need from the real map data and build a concise map. This will be the prior knowledge of the MOT process, which means the construction of this information will not influence the MOT time.

For every single log in KITTI datasets, the GPS trajectory of the ego vehicle is stored in the "oxts" directory. We first get the GPS trajectory of ego vehicle from the file and extract the corresponding area's map data from the OpenStreetMap. The map data is stored in XML file format.

We refine the original data and build a concise representation which contains both semantic and geometric information according to the definition in Section 3.4. In this case, the concise representation contains the coordinates and semantic label like "forest", "grass", "building" of all polygons representing the no-driving zone. The real map and the corresponding concise map are shown in Fig. **??**.

Finally, we organize these information into a single JSON file and provide it to the pre- and post-filter.

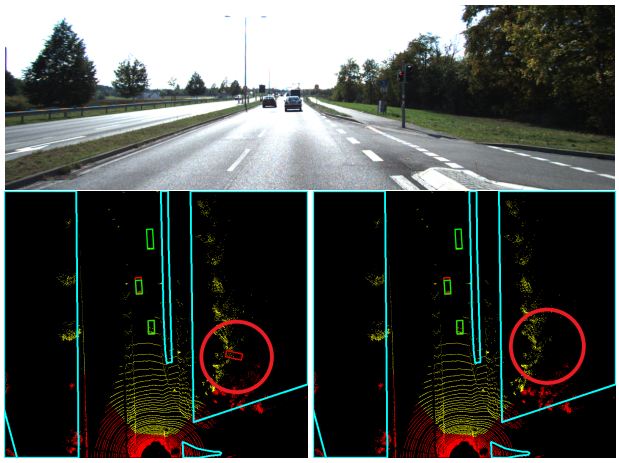

Figure 3: The new detection result after remove the FP

4.2 Pre- and Post-Filter

After getting the no-driving zones extracted from the real map information, we can make a simple judgment on each point in the point cloud or each object produced by the detector. This judgment is based on an assumption that a car should not appear at the no-driving zones which could be forest, building, greenbelt, etc. So that, for the pre-filter, we want to remove all the points in the no-driving zones in the point cloud. In this way, we can reduce the number of points in the point cloud, which may theoretically accelerate the inference time of the detection model. And we will talk about it according to the result of the experiment. For the post-filter, we regard the cars located at the no-driving zones as false positives, so we want to remove them to get a better detection result. To elaborate on this process, we split it into steps as below.

Step 1. Get the ROI in the Global Coordinate System. In KITTI dataset, we can get the LiDAR data in a range of limitation. Generally, we use the LiDAR data within 40m on the left side and the right side, and within 70.4m in the front. And we call the region of the LiDAR data the ROI. In order to get the information of no-driving zones within the ROI, we should transform the ROI that is in the local coordinate system to the global coordinate system.

Simply, we express the ROI as a rectangle, which is represented by 4 points (p_1, p_2, p_3, p_4). The coordinates of the points take the car as the center point. Based on the relationship between the KITTI's several coordinate systems, first of all, rotate these points to the reference camera coordinate system; then transform them to the velodyne coordinate system; finally transform them to the IMU coordinate system. And we can get the global location and yaw angle from the car's GPS information, then perform a simple coordinate conversion on each point of the rectangle. Thus, we

could get the rectangle with global coordinates, which is the ROI in the global coordinate system. By the way, the ROI is in meters, so we use the car's coordinates in the UTM system for convenience, which is also in meters. Because there are no UTM coordinates in the KITTI dataset, we should transform the GPS coordinates into the UTM coordinates.

It should be noted that the reason why we transform the ROI rectangle to the global coordinate system, rather than transform the no-driving polygons to the local coordinate is we think this could reduce the amount of calculation. For the ROI rectangle, we just need to transform 4 points. But for the no-driving polygons, we may need to transform a lot of points, because there maybe a lot of polygons, and some of them could be really complex (containing lots of points).

Step 2. Take Intersections of ROI and No-driving Zones. The no-driving zones are represented by polygons. In order to get the no-driving zones in the ROI, we should take intersections of ROI rectangle and no-driving polygons.

This problem could be abstracted as that taking intersection of a polygon and a convex polygon (here is the ROI rectangle). Fortunately, the Sutherland-Hodgman polygon clipping algorithm [21] finds the polygon that is the intersection between an arbitrary polygon (the "subject polygon") and a convex polygon (the "clip polygon"). And this algorithm is just designed for the problem what we need to solve here.

By using the Sutherland-Hodgman polygon clipping algorithm, we can easily get the no-driving zones in the ROI.

Step 3. Transform the Coordinates of the Intersections into the Local Coordinate System. This step is just opposite to the first step "Get the ROI in the Global Coordinate System". This step aims to transform the coordinate of non-driving zones in the ROI into the coordinate in the local coordinate system.

Due to the different definition between the GPS coordinate system and camera coordinate system in the setup of KITTI dataset, we need to take several additional transformations to the points of no-driving zones. The transform matrix is already provided by the dataset.

Step 4. The Filter. Up to now, we get the no-driving zone in the local coordinate system, then we will use it to filter out the points we want. And the only difference between the pre-filter and the post-filter is here.

For the pre-filter, we want to remove the points in the no-driving zones. For the post-filter, we want to remove the objects. They can be attributed to the same problem, but with different inputs.

Similarly, this problem could be abstracted as that determining whether a point is in some polygons. Because the region of interest has a certain size, we regard the ROI as an image and the points as pixels by rounding their coordinates in centimeters. Then we draw a mask of these polygons on the image, which can judge whether a point is in some polygons by addressing with its coordinates.

For the pre-filter, we remove all the points which are located in the no-driving zones and use the remaining points as the input of the detector.

For the post-filter, as shown in Fig. 3, if the center point of an object is in some polygons, which means the object corresponded with the center point is located in the no-driving zones, we regard the object as a false positive. Then we remove it from the detection result, so as not to be tracked in the following steps.

4.3 Tracking Management

In previous works, in order to exclude some false positives, when the object first appears, the object will be regarded as a false positive. So the information of an object needs to be stored for several frames before tracking the object as a real object. This causes a lot of false negatives in the start frames of every object. Since we already remove some obvious false positives in the original detection result, we could employ a more radical strategy to manage the tracking process. In short, with great confidence, we could track the object from the first frame that the object has been detected by the detector.

4.4 Evaluation

4.4.1 *Experimental Environment Settings.*

Environment. The experimental environment is a server with 2 Intel Xeon CPU E5-2690 v3 (each with 12 physical cores, hyper-threading enabled), 4 Geforce RTX 2080 Ti GPUs (each with 4352 cores, 12GB of video memory, actually only one GPU is used in the experiment), and 256GB of memory. Server software configuration: the operating system is Ubuntu 18.04, Python version is 3.7.7, and CUDA version is 10.2.

Detector and Tracker. Our work is based on the 3D tracking-by-detection framework. We choose the detector and tracker off the shelf on the leaderboard of the KITTI object detection and tracking task. For the tracker, we choose AB3DMOT [23] because it is the state of the art (SOTA) open source 3D MOT framework following tracking-by-detection paradigm on KITTI tracking leaderboard. For the detecter, we chosse the SA-SSD[10] and some detectors supported by OpenPCDet[4] like PV-RCNN[18], SECOND[25], and PointRCNN[19].

For AB3DMOT, we set Bir_{min} to 1, 2, 3 to show how this parameter affects the results and why our method can lower this parameter to get much better results. And because of no influence to our method, we set Age_{max} to 2, where Age_{max} is the max number of frames to hold the tracklet that is unmatched.

Evaluation Metrics. For precision, we use standard CLEAR metrics [2] including MOTA, MOTP, FP, FN. The MOTA considers all the object matching errors in all frames in tracking, so it can reflect the effect after using our method. And the MOTP can let us know whether the precision of detector has been influenced after using our method. The FP is what we are focusing on, and our method is trying to lower it. The last one, FN, is influenced by the Bit_{min}, and we want to show this effect in quantity. For KITTI 2D MOT evaluation, we use the official KITTI 2D MOT evaluation tool [7]. For 3D MOT evaluation, we use the evaluation tool proposed by [23].

Time Statistics. We count the time for detection, tracking, and the whole running time. Further, we count the inference time of the detection model.

[4] https://github.com/open-mmlab/OpenPCDet#model-zoo

4.4.2 Precision. The results on KITTI 3D and 2D evaluation tools are shown at Tables 2 and 3, respectively.

Table 2: Results on the KITTI 3D MOT evaluation tool proposed by [23]

Detctor & Tracker	Cfg	B_m	Metrics			
			$MOTA\uparrow$	$MOTP\uparrow$	$FP\downarrow$	$FN\downarrow$
SA-SSD AB3DMOT	–	1	0.7302	0.7371	28	244
	Pre	1	**0.7500**	0.7370	**8**	244
	Post	1	**0.7500**	0.7372	**8**	244
PV-RCNN AB3DMOT	–	1	0.8214	0.738	11	169
	Pre	1	0.8065	0.7439	7	188
	Post	1	**0.8234**	0.738	**9**	169
SECOND AB3DMOT	–	3	0.8373	0.7319	4	160
	Pre	1	0.8502	0.7281	7	144
	Post	1	**0.8512**	0.7291	4	**146**
PointRCNN AB3DMOT	–	1	0.8006	0.7227	18	182
	Pre	1	**0.8135**	0.7265	**9**	178
	Post	3	0.7946	0.7162	3	204

B_m represents Bir_{min} in this paper.
– in column Cfg means the original framework
Pre in column Cfg means the utilization of pre-filter.
Post in column Cgf means the utilization of post-filter.

In Table 2, we show our results on the KITTI 3D MOT evaluation tool proposed by [23]. For each combination of detector and tracker, we test the original framework, the framework using pre-filter and using post-filter, respectively. For each of them, we present the best MOTA and the corresponding Bir_{min} in this Table. As we can see from the table, the framework using pre- or post-filter method can achieve nearly 2% MOTA improvement. We also notice that the MOTA may decrease in some cases, *e.g.*, the result of PV-RCNN with pre-filter. The main reason is that the evaluation tool proposed by [23] would automatically try different thresholds of the score of objects to find out the best final result. When we use our filter method, the total number of tracking result decreases and therefore result in the difference of best threshold. A higher threshold means more tracking results will be ignored, including some True Positives with a low score. As a result, the FN increases and MOTA decreases.

In order to eliminate the impact of threshold changes introduced by [23] on final results and explore the potential of filter methods, we use the official KITTI 2D MOT evaluation tool with no filter threshold. We find that all combinations of detector and tracker will get better performance with both pre- and post-filter methods. We present the best MOTA results and corresponding Bir_{min} in Table 3. As we can see, with the same parameters, the FN of three kinds of the framework is almost the same. The improvement is from the reduction of the FPs. The result of SECOND and AB3DMOT with post-filter even have a MOTA improvement of 11% and a FP reducement of 111 .

In original method, decreasing Bir_{min} can reduce the FN , but result in a rapid increase in FP, so it cannot bring significant performance improvement sometimes. With our approach, we can choose to use lower Bir_{min} and get lower FN and FP simultaneously.

Table 3: Results on the official KITTI 2D MOT evaluation tool

Detctor & Tracker	Cfg	B_m	Metrics			
			$MOTA\uparrow$	$MOTP\uparrow$	$FP\downarrow$	$FN\downarrow$
SA-SSD AB3DMOT	–	1	0.6905	0.8286	71	239
	Pre	1	0.7341	0.8290	27	239
	Post	1	**0.7421**	0.8286	**19**	239
PV-RCNN AB3DMOT	–	3	0.6865	0.8443	147	169
	Pre	3	0.6894	0.8471	141	172
	Post	3	**0.7123**	0.8442	**121**	169
SECOND AB3DMOT	–	3	0.6310	0.8343	237	135
	Pre	3	0.6845	0.8333	183	135
	Post	3	**0.7410**	0.8343	**126**	135
PointRCNN AB3DMOT	–	3	0.7470	0.8307	69	186
	Pre	3	**0.7718**	0.8347	**56**	174
	Post	3	0.7708	0.8282	49	182

From the metric MOTP, we can find that our method will not influence the precision of the detector, whether it is the pre-filter or post-filter. The MOTP metric is defined as:

$$MOTP = \frac{\sum_{t,i} d_{t,i}}{\sum_t c_t} \tag{1}$$

where c_t denotes the number of matches in frame t and $d_{t,i}$ is the bounding box overlap of target i with its assigned ground truth object. With lower Bir_{min}, more detected objects are considered as true positive and some of them may have low confidence and therefore $d_{t,i}$ is small. Since MOTP gives the average overlap between all correctly matched hypotheses, most of the filtered FP by our method may only appear in one frame and have no matches, therefore they will not be calculated in metric MOTP.

4.4.3 Time Consumption. The results of the time consumption are shown in Table 4. We present the corresponding time consumption of the best performance configuration in Table 2. The results of T_{infer} and T_{trac} are just as we expected. For tracking framework using pre-filter, The filter method reduces the data volume, which results in an inference time decrease. For both pre- and post-filter methods, the total number of tracking objects is reduced, so we can observe that both methods have a lower tracking time.

As for the T_{det}, the detection time of framework using pre-filter method include the inference time and filter task time, so it should not be larger than the sum of original detection time and filter time. And the detection time of framework using post-filter method is expected as almost the same as original framework. All the experimental results performed in line with expectations except for the SA-SSD with post-filter. It has a larger detection time increase than we expected. We think it's probably our framework design problem, and we will continuously optimize it in the future work.

Table 4: Time Consumption.

Detctor & Tracker	Cfg	B_m	Times(ms)				
			T_{infer}	T_{trac}	T_{det}	T_{run}	T_{fl}
SA-SSD AB3DMOT	–	1	49.6	8.7	90.9	99.8	-
	Pre	1	**48.3**	**5.2**	**90.7**	**96.0**	7.5
	Post	1	52.1	8.1	96.8	115.4	10.0
PV-RCNN AB3DMOT	–	1	136.8	16.9	181.7	198.8	-
	Pre	1	**121.0**	**13.9**	**173.8**	**187.7**	9.8
	Post	1	136.4	15.2	173.9	200.0	10.9
SECOND AB3DMOT	–	1	55.6	21.8	105.3	127.2	-
	Pre	1	**52.7**	16.3	111.2	127.7	11.0
	Post	1	55.4	**15.9**	105.1	132.0	10.8
PointRCNN AB3DMOT	–	1	95.0	6.6	118.1	124.8	-
	Pre	1	**93.9**	**6.0**	123.5	129.5	7.1
	Post	3	95.3	6.2	119.7	133.8	7.9

T_{det} contains the time for preprocessing and inference.
T_{infer} is the time for inference.
T_{trac} is the time for tracking
T_{fl} is the time for the filter task.

5 DISCUSSION

In this section, we make a discussion of our method and analyze the remaining issues.

We have to point out that in Autoware and Baidu Apollo systems, there are similar functions to use map information to filter the point or detected objects as our proposed pre- and post-filter algorithm. As we emphasized in Section 2, these implementations are tightly coupled with specific frameworks and map formats, while our design goal is to make it adaptable to various existing or future tracking frameworks and systems. Some algorithms like pre-filter algorithm can also be applied to other tracking frameworks. As we discussed in Section 3, our method is not just to integrate map information into tracking systems. The semantic-geometric map we proposed in Section 3.4 is designed to model scene information, which makes this method extensible to support more scene information. As shown in Table 1, different levels of infrastructure support for automated driving have different levels of underlying exchanged information. The information of level A digital infrastructure is far more than just maps that we mainly use in our experimental framework. Integrating more different scene information into our tracking systems is important future work.

An important performance influencing factor in our method is the localization precision of the ego-vehicle. Global localization requires the autonomous vehicle to match the local map with global map and correcting its position. In real autonomous driving situation, the localization algorithm is also one of the most important research topics. The primary experiments in this paper are designed for publicly available datasets commonly used in autonomous driving. Thus, the semantic maps in our framework are build on an approximation map we extracted from OpenStreetMap based on GPS data. We assume the positioning error of GPS and IMU data in dataset is acceptable for our framework. Since we use concise map designed in our framework to represent map information, we can extract semantic and geometric information from different resources to build it . In the future, we will extend it to scene information and focus on building a more accurate and concise representation.

6 CONCLUSION

In this paper, We propose the methodology for utilizing map information in general tracking-by-detection framework to improve tracking performance. In order to achieve the real-time and universality goal, we propose a semantic-geometric map to model the concise scene information extracted from original data. This method can be applied to various tracking-by-detection systems with few modifications. Based on our method, we design a prototype tracking system and use no-driving zones as map information to improve performance. We design an algorithm on utilizing no-driving zone information to reduce the false positives of the detection module. This algorithm can be used before or after the detection process. Experiments show that the prototype framework can achieve better performance than original tracking systems without bringing overhead.

In the future, we will integrate more scene information into our framework, for example, the curves of the road lane, the spatial and temporal distribution of agents and so on. We believe that with the development of communication technology and roadside intelligent infrastructure, the scene information will become the key to breaking through the bottleneck of MOT.

ACKNOWLEDGMENTS

This work was partially funded by the the National Key Research and Development Program of China (No. 2018AAA0100500), the National Natural Science Foundation of China (No. 61772487) and Anhui Provincial Development and Reform Commission 2020 New Energy Vehicle Industry Innovation Development Project "Key System Research and Vehicle Development for Mass Production Oriented Highly Autonomous Driving".

REFERENCES

[1] E. Baser, V. Balasubramanian, P. Bhattacharyya, and K. Czarnecki. 2019. FANTrack: 3D Multi-Object Tracking with Feature Association Network. In *2019 IEEE Intelligent Vehicles Symposium (IV)*. 1426–1433. https://doi.org/10.1109/IVS.2019.8813779

[2] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics. *J. Image Video Process.* 2008, Article 1 (Jan. 2008), 10 pages. https://doi.org/10.1155/2008/246309

[3] Anna Carreras, Xavier Daura, Jacqueline Erhart, and Stefan Ruehrup. 2018. Road infrastructure support levels for automated driving. In *Proceedings of the 25th ITS World Congress, Copenhagen, Denmark*. 17–21.

[4] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. 2019. Argoverse: 3D tracking and forecasting with rich maps. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 8748–8757.

[5] Andreas Danzer, Fabian Gies, and Klaus Dietmayer. 2018. Multi-Object Tracking with Interacting Vehicles and Road Map Information. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 589–595.

[6] Patrick Emami, Panos M Pardalos, Lily Elefteriadou, and Sanjay Ranka. 2020. Machine Learning Methods for Data Association in Multi-Object Tracking. *ACM Computing Surveys (CSUR)* 53, 4 (2020), 1–34.

[7] Andreas Geiger. 2012. Are We Ready for Autonomous Driving? The KITTI Vision Benchmark Suite. In *Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (CVPR '12)*. IEEE Computer Society, USA, 3354–3361.

[8] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing* 7, 4 (2008), 12–18.

[9] Robert Harper. 2016. *Practical foundations for programming languages*. Cambridge University Press.

[10] Chenhang He, Hui Zeng, Jianqiang Huang, Xian-Sheng Hua, and Lei Zhang. 2020. Structure Aware Single-stage 3D Object Detection from Point Cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 11873–11882.

[11] Florian Jomrich. 2020. *Dynamic Maps for Highly Automated Driving-Generation, Distribution and Provision.* Ph.D. Dissertation. Technische Universität.
[12] Rudolph Emil Kalman. 1960. A new approach to linear filtering and prediction problems. (1960).
[13] Nikola Karamanov, Desislav Andreev, Martin Pfeifle, Hendrik Bock, Mathias Otto, and Matthias Schulze. 2018. Map line interface for autonomous driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 26–33.
[14] Sebastian Krebs, Bharanidhar Duraisamy, and Fabian Flohr. 2017. A survey on leveraging deep neural networks for object tracking. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 411–418.
[15] Alex H. Lang, Sourabh Vora, Holger Caesar, Lubing Zhou, Jiong Yang, and Oscar Beijbom. 2019. PointPillars: Fast Encoders for Object Detection From Point Clouds. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 12697–12705.
[16] Fabian Poggenhans, Jan-Hendrik Pauls, Johannes Janosovits, Stefan Orf, Maximilian Naumann, Florian Kuhnt, and Matthias Mayr. 2018. Lanelet2: A high-definition map framework for the future of automated driving. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 1672–1679.
[17] Abhijeet Shenoi, Mihir Patel, JunYoung Gwak, Patrick Goebel, Amir Sadeghian, Hamid Rezatofighi, Roberto Martín-Martín, and Silvio Savarese. 2020. JRMOT: A Real-Time 3D Multi-Object Tracker and a New Large-Scale Dataset. arXiv:2002.08397 [cs.CV]
[18] Shaoshuai Shi, Chaoxu Guo, Li Jiang, Zhe Wang, Jianping Shi, Xiaogang Wang, and Hongsheng Li. 2020. Pv-rcnn: Point-voxel feature set abstraction for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10529–10538.
[19] Shaoshuai Shi, Xiaogang Wang, and Hongsheng Li. 2019. Pointrcnn: 3d object proposal generation and detection from point cloud. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 770–779.
[20] S. Shi, Z. Wang, J. Shi, X. Wang, and H. Li. 2020. From Points to Parts: 3D Object Detection from Point Cloud with Part-aware and Part-aggregation Network. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020), 1–1. https://doi.org/10.1109/TPAMI.2020.2977026
[21] Ivan E. Sutherland and Gary W. Hodgman. 1974. Reentrant Polygon Clipping. *Commun. ACM* 17, 1 (Jan. 1974), 32–42. https://doi.org/10.1145/360767.360802
[22] Martin Treiber, Ansgar Hennecke, and Dirk Helbing. 2000. Congested traffic states in empirical observations and microscopic simulations. *Physical review E* 62, 2 (2000), 1805.
[23] Xinshuo Weng, Jianren Wang, David Held, and Kris Kitani. 2020. 3D Multi-Object Tracking: A Baseline and New Evaluation Metrics. *IROS* (2020).
[24] Xinshuo Weng, Yongxin Wang, Yunze Man, and Kris Kitani. 2020. GNN3DMOT: Graph Neural Network for 3D Multi-Object Tracking with 2D-3D Multi-Feature Learning. *CVPR* (2020).
[25] Yan Yan, Yuxing Mao, and Bo Li. 2018. Second: Sparsely embedded convolutional detection. *Sensors* 18, 10 (2018), 3337.
[26] Bin Yang, Wenjie Luo, and Raquel Urtasun. 2018. PIXOR: Real-Time 3D Object Detection From Point Clouds. In *2018 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, UT, USA, June 18-22, 2018*. IEEE Computer Society, 7652–7660. https://doi.org/10.1109/CVPR.2018.00798
[27] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking objects as points. (2020), 474–490.