Motion Planning In Complex Urban Environments: An Industrial Application on Autonomous Last-Mile Delivery Vehicles*

Haiming Wang[1], Liangliang Zhang[1,†], Qi Kong[1], Weicheng Zhu[1], Jie Zheng[2], Li Zhuang[1] and Xin Xu[2]

Keywords: Path planning, vehicle robot, autonomous navigation, Optimization

***Abstract*— In this article, a motion planning framework for autonomous driving is explored to achieve unmanned last-mile delivery vehicle application in complicated urban scenario. This approach can dramatically improve the intelligence, driving safety, driving robustness, and scalability of autonomous vehicles. In this framework, a specific High-Definition (HD) map representation was proposed for last-mile delivery applications, and a Route Planning layer generates a kinematically feasible reference line with smoothness condition based on Routing and High-Definition (HD) Map components. Then a Scenario Planning layer considers routing results and both static and dynamic obstacles into account to select corresponding scenario to execute, such as cruise on road scenario, cross intersection scenario, parking scenario, etc. Finally, a Trajectory Planning layer which is classified with trajectory generation and optimization modules is described. In trajectory generation part, rough path-speed profile and corresponding decisions, such as nudge, stop, yield, etc, are generated. Then, the rough path-speed profile is post-processed by optimization algorithms in trajectory optimization part. Based on the real road test results from thousands of times for JD.com's real autonomous delivery vehicle operations and approximate 115,475 kilometers (km) of fully autonomous driving mode in urban scenario, the proposed motion planning framework demonstrates the efficiency of autonomous driving, improves the driving quality and reduces the manual intervention.**

I. INTRODUCTION

This paper is focused on the motion planning framework for an autonomous driving delivery application via a novel route-scenario-trajectory planning architecture. Autonomous driving vehicles have a great capability to improve transportation efficiency and reduce traffic accidents caused by human drivers' mistakes. Additionally, they can also save human resources and considerable time for the whole society. Fortunately, the autonomous driving technology has been paid attention and rapidly developed from robotics research community, such as the Defense Advanced Research Projects Agency (DARPA) Grand and Urban Challenge (Thrun et al., 2006; Buehler, Iagnemma, & Singh, 2007, 2009; Urmson et al., 2008). Moreover, an advanced driving assistance systems (ADAS) such as adaptive cruise control, lane keeping feature, and self-parking systems have been widely implemented in commercial luxury cars. These ADAS functions can efficiently reduce accidents and exhibit great autonomous driving performance. However, these systems only perform on very simple driving scenarios and still need human intervention constantly. Thus, the technology is still far away from true level-4 autonomous driving.

In the meanwhile, a lot of research platforms and companies' projects have demonstrated the great potential for autonomous driving technology on public transportations. However, it is still too early to claim that autonomous driving technology can fully replace human drivers in passenger vehicles. Vehicle reliability, passenger safety, and cost in sensors and technology are still challenges in practical applications. For example, based on 2020 Autonomous Vehicle Disengagement Reports issued by California's Department of Motor Vehicles, the best number of miles per a disengagement, which is 29,944 from Waymo, is still way below the number that 165,000 miles on average per an accident caused by human driver. Therefore, significant challenges in terms of technology still remain for the commercialization of autonomous passenger vehicles(Paden, Čáp, Yong, Yershov, & Frazzoli, 2016; Buehler et al., 2009).

Last-mile delivery, however, is a very promising application scenario where commercialization of autonomous driving technology is possible within the foreseeable future. As the continuous growth of e-commerce, last-mile delivery plays a significant role in the e-commerce experience. However, the urban last-mile delivery is the most expensive part of the whole supply chain, so autonomous driving technology is expected to reduce the cost and improve efficiency. Compared with passenger vehicles, there are three significant differences for delivery vehicles. The last-mile delivery vehicle is generally operated at relatively low speeds, typically within 20 miles per hour (mph), compared with passenger vehicles' speed from 35 mph to 70 mph on average. A slow vehicle needs shorter perception distance and shorter brake distance. Secondly, the last-mile delivery vehicle is typically smaller and lighter than the passenger vehicles, thus, this further decreases the risk and fatality of traffic accidents. Finally, a delivery vehicle is free of passengers, therefore, the safety and driving requirements are much lower, compared with passenger vehicles. Especially, in extreme conditions, a delivery vehicle can sacrifice itself to protect other vehicles sharing the road.

Although the last-mile delivery vehicles is very promising for commercial application, there are still challenges. Firstly, compared with passenger vehicles application, last-mile delivery vehicles often operate at more complicated urban environments (Li et al., 2020). For passenger vehicles

*This work was supported by JD.com, Beijing, China.

[1]Haiming Wang, Liangliang Zhang, Qi Kong, Weicheng Zhu, and Li Zhuang are with the Autonomous Driving Devision, JD.com American Technologies Corporation, Mountain View, CA, US 94043. {haiming.wang, liangliang.zhang, qi.kong, weicheng.zhu, li.zhuang}@jd.com

[2]Jie Zheng and Xin Xu are with the Autonomous Driving Devision, JD.com, Beijing, China. {zhengjie5, xuxin178}@jd.com

† Corresponding authors.

based Robotaxi and high way transportation applications, they all have well-defined scenarios, structured space, and clear traffic regulations. Last-mile delivery vehicles, however, operate under irregular situations in a significant of amount portion of its operation time. With the rapid growth of world's urban population, solving last mile delivery problem becomes a lot more complicated. Especially in China, there is about 830 million people lived in urban regions. Consequently, such a large amount of urban population induces three major challenges in last-mile delivery. Firstly, high population density makes urban residents live in apartment or condominium rather than single family house. As a result, when last-mile delivery vehicle approaches the destination, unexpected situations may happen in this unstructured environment due to lack of traffic rules, such as, parking spot occupied by unknown objects, pedestrians walking around, interference from passenger vehicles, etc. Secondly, even though in structured local roads, bicycles, motorcycles, and different types of automobiles share the road and have different kinematic feature, so that, it is difficult to interact with this type of complex scenario. Finally, since last-mile delivery vehicle has a relatively slow speed, then its behavior becomes more complicated when crossing the intersections or making left/U turn with traffic lights. Therefore, the development of autonomous last-mile delivery vehicle requires advanced technologies in perception and planning (Kümmerle, Ruhnke, Steder, Stachniss, & Burgard, 2015). In particular, in order to handle with special needs for last-mile delivery, advanced motion planning algorithm needs to be developed to determine the behavior of the vehicles.

Research on motion planning can primarily be classified in graph search based approach, sampling based approach, interpolating based approach, and optimization based approach (González, Pérez, Milanés, & Nashashibi, 2015). Firstly, graph search based planning, such as Dijkstra algorithm (Bacha et al., 2008), $A^\star$ algorithm (Ziegler, Werling, & Schroder, 2008), State Lattice algorithm (Howard & Kelly, 2007), are frequently implemented in finding a global and local path while avoiding obstacles in surrounding environments. However, the resolution of the grid and lattice will compromise between the optimality of a path and the efficiency of computation load. On the other hand, it is difficult to generate a kinematically feasible path by using graph search based approach only. Sampling based planning solves the planning problems in high dimensional spaces. This approach generates a collision-free path by sampling configuration space of the vehicle. In sampling based planning, the most commonly used method Rapidly-exploring Random Tree (RRT) (Karaman & Frazzoli, 2011) has been extensively tested for automated vehicle. The shortcoming of this method is that the resulting path sometimes is not optimal, not smooth and not curvature continuous. Interpolating based approach uses pre-known a set of waypoints obtained from a map to generate a new set of data (path) which obeys trajectory continuity, obstacle avoidance, and vehicle constraints. The interpolating based planning implements different techniques for trajectory generation and smoothing, such as clothoid curves (Brezak & Petrović, 2013), polynomial curves (Glaser, Vanholme, Mammar, Gruyer, & Nouveliere, 2010), spline curves (Berglund, Brodnik, Jonsson, Staffanson, & Soderkvist, 2009), etc. The interpolating planner is very easy to implement (Ferguson & Stentz, 2007), however, it highly relies on global planning or global waypoints, so that, it is inflexible during on-road planning. Finally, optimization based approach has been successfully demonstrated in autonomous driving in DARPA challenges. In this scheme, optimal paths are generated by minimizing a cost function subjecting to different constrains, such as station, velocity, acceleration, jerk, and road boundary. Moreover, it is often used to smooth previously computed collision-free trajectories (Ferguson, Howard, & Likhachev, 2008; Fan et al., 2018). For motion planning in autonomous driving area, optimization based algorithms are mainly developed in Frenet frame (Werling, Ziegler, Kammel, & Thrun, 2010). Generally, these algorithms are divided into direct optimization methods and path-speed decoupling methods. Direct optimization methods (McNaughton, Urmson, Dolan, & Lee, 2011) solve the optimal trajectory directly by searching. The challenge of this approach is computation load greatly increases as search resolution increasing. Thus, the computed trajectory may not be optimal due to the time consumption requirements. On the other hand, path-speed decoupling methods (Fan et al., 2018; Gu, Atwood, Dong, Dolan, & Lee, 2015) optimize path and speed separately by various constrains. This philosophy can design and optimize path and speed independently, then synthesize them into desired trajectory. Therefore, this approach can improve the robustness and flexibility of optimization.

In this paper, the proposed approach builds on the existing work discussed above. Particularly, our approach uses a combined route planning, scenario planning, and trajectory planning architecture to address the unique issues in last-mile delivery application. Before introducing the proposed motion planning framework, a novel multi-layer High-Definition (HD) map design philosophy based on special last-mile delivery application is described. After that, we focus on describing the proposed motion planning architecture, firstly, a single and multiple destinations based route planning is executed, by using a traditional graph search algorithm. Secondly, a route smoothing is completed by a nonlinear optimization method. Then, the scenario planning module is used to choose a scenario based on current vehicle driving status gained from route planning and a set of environmental features from perception, prediction, and localization. Moreover, some special scenarios are specifically designed for last-mile delivery. Finally, once a scenario is selected, such as cruise on-road scenario, motion planner continuously executes all tasks sequentially. These tasks can be classified with behavior decision task, trajectory generation task, and trajectory optimization task. Our contributions for motion planning in last mile delivery is in terms of: (1) HD map's specific design for last-mile delivery application; (2) Multiple destinations based route planning for complex last-mile delivery scenario; (3) Additional scenario design for last-mile

delivery; (4) Novel cost functions' establishment for path-speed generation and optimization tasks.

In this article, we focus on the description of the motion planning for autonomous last-mile delivery vehicles in a practical way, with experiments rather than theoretical improvement of the motion planning algorithm. The proposed route, scenario, and trajectory planning framework was applied to the autonomous last-mile delivery vehicle ROVER 5.0 of JD Logistics, which was invented by JD.com, Beijing, China. ROVER 5.0 delivery vehicle ran daily delivery operations in multiple communities and campuses of six big cities which are Suzhou, Beijing, Wuhan, Xi'an, Xianyang, and Guangzhou in China since 2020. Since 2020 until now, ROVER 5.0 drove over 100,000 kilometers (km) in those cities, and successfully delivered 350,574 orders. In particular, when the COVID-19 outbroke in Wuhan, Hubei, China during January to March, 2020, JD.com deployed ROVER 5.0 autonomous delivery vehicles in Wuhan to deliver medical supplies and living material from JD's local distribution center to Wuhan's ninth hospital and three large communities, as shown in Fig. 1. These unmanned autonomous delivery largely eliminated human contact and protect customers not to be exposed in pandemic disease situation. During three months' autonomous delivery operations in Wuhan, ROVER 5.0 drove around 200 km and successfully delivered more than 1,600 orders.

Fig. 1. JD.com's autonomous vehicle ROVER5 is shipping medical supplies to Wuhan Ninth Hospital during COVID-19 pandemic.

The remainder of this paper is organized as follows. The autonomous last-mile delivery vehicle ROVER 5.0 is described in Section II. The system architecture and structure of the motion planning algorithm are briefly introduced in Section III. A description for details, which illustrates the proposed motion planning approach, is given as following: HD map special design for last-mile delivery application in Section IV; Route planning is described in Section V; Scenario planning is described in Section VI; Trajectory planning is described in Section VII. In Section VIII, a simulation platform, which is support of motion planning development and validation of autonomous driving systems, is descried. In Section IX, we provide a case study with real delivery scenario, and demonstrate the efficacy of the the proposed approach. This article is ultimately concluded in Section X.

II. ROVER 5.0 VEHICLE

ROVER 5.0 vehicle, as shown in Fig. 2, is designed by JD.com, Beijing, China. The system of ROVER 5.0 mainly consists of a chassis, a power system, a computing unit, sensors, and accessories. ROVER 5.0 has a four-wheel motor drive, ackerman-steering, and an electronic hydraulic brake system. To protect the vehicle from environment impact, ROVER 5.0 installs front and rear bumpers and an emergency brake button for emergencies. Since ROVER 5.0 is designed for last-mile delivery applications, so there are multiple containers on the vehicle. For the power supply, ROVER 5.0 has a chargeable lithimion battery with 48V, 80AH which can provide all power for the autonomy hardware, and the maximum mileage for single-charge battery range is about 50Km.

Fig. 2. ROVER 5.0 last-mile delivery vehicle.

The vehicle states related data, sensors data, and map data are communicated to the computer system through USB interfaces. ROVER 5.0 has a Human Machine Interface (HMI) system. A LED touch screen is installed on the rear part of vehicle, that customers can communicate with vehicle for goods pickup. Furthermore, ROVER 5.0 vehicle installs a vehicle monitoring system which can monitor the road environments around the ego vehicle and publish video streaming on the authorized Internet.

For computation, ROVER 5.0 uses two Nvidia Jetson AGX Xaviers which integrates with 8-core NVIDIA Carmel Armv8.2 64-bit CPU and 512 NVIDIA CUDA cores and 64 Tensor cores GPU. Jetson AGX Xavier also delivers up to 32 TOPs of AI performance. Moreover, a 1T solid state drive (SSD) is mounted for the data logging. The accessories mainly consist of LED display that customers can interface with vehicle for goods pickup and remote controller which can manually control the vehicle remotely.

ROVER 5.0 uses a number of sensors to navigate in complex urban environments. The sensors used in ROVER

5.0, can be seen in Table. I. For a measure of the global position of the vehicle, a single point Global Positioning Systems (GPS) was installed on the upper rear part of the vehicle. A combined laser-map and visual-map matching algorithm was employed to estimate the vehicle's position and orientation, by integrating GPS data with multi-line LIDAR, High Dynamic Range (HDR) camera, Inertia Measurement Unit (IMU), and Odometry data. For obstacle detection and tracking, one 16-line LIDAR was installed on the roof of ROVER 5.0, and the other 16-line LIDAR was mounted on the front bumper. The maximum range of all three LIDARs is 100 m. Additionally, four mono cameras were also installed on the roof to detect the surrounding obstacles. For protection of vehicle's back up behavior, another 1-line LIDAR was installed on the rear bumper, and its maximum range is 10 m. For traffic light detection, a HDR camera was mounted on the upper portion of vehicle. Furthermore, 12 ultrasonic receptors are mounted on the lower portion of vehicle to prevent collisions from obstacles and road elements.

TABLE I
SENSOR DESCRIPTION INTO ROVER 5.0

Sensor	Characteristics
RAC-C1 GPS	Positioning accuracy within 20-60cm dynamically and 1.5m statically, speed accuracy within 0.1m/s
ADIS16470 IMU	A triaxial gyroscope ±2000°/sec dynamic range and a triaxial accelerometer with ±40g dynamic range
Velodyne VLP-16 LIDAR	360° horizontal FOV with 0.1° resolution, 30° vertical FOV with 2° resolution, 100m maximum range
Sick TIM551 LIDAR	270° FOV with 1° resolution, 0.05m ∼ 10m working range
BFS-U3-16S2C Mono Camera	CMOS, Color, 226FPS with 1440×1080 Pixels
BFS-U3-23S3C HDR Camera	CMOS, Color, 163FPS with 1920×1200 Pixels

III. SYSTEM ARCHITECTURE

A. System Framework of ROVER 5.0

The autonomous driving system of ROVER 5.0 has a layered architecture, as shown in Fig. 3, which consists of an application layer, a component layer, a system layer, and a hardware and driver layer. Additionally, data platform, human machine interface (HMI) and visualization tool are also utilized by the layered architecture.

The application layer is comprised with both on-line and off-line platforms. ROVER 5.0's on-line platforms contain with the delivery service system, real-time monitoring system, and business scheduling system among different application scenarios. Additionally, the roles of off-line platform are mainly composed of off-line data labeling, training, simulation, and validation which are subject to simulation platform and machine learning platform.

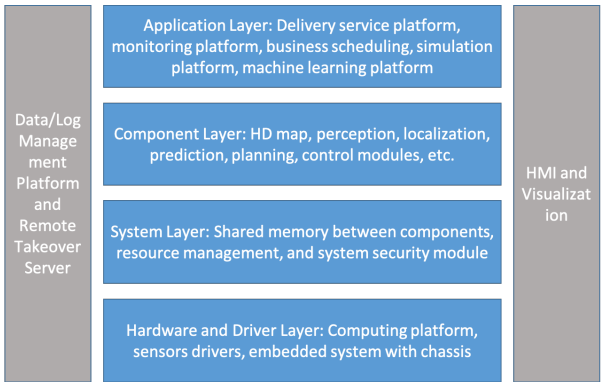

Fig. 3. System architecture of ROVER 5.0 last-mile delivery vehicle.

The component layer is mainly composed by the individual low-level algorithm module, such as perception, localization, prediction, planning, and control modules, etc. The relationship of these modules are as shown in Fig. 4. The HD map module provides a high definition global map as a benchmark. From the sensor reading collected by various advanced sensors, the localization and perception modules estimate the vehicle's driving states and perceive dynamic surrounding environment. The predictive environment (such as obstacles) variation can be obtained by prediction algorithm. Ultimately, all the information is gathered to motion planning module, which makes the driving behavior decision and generate a collision-free and kinematically/dynamically feasible trajectory applying to control module.

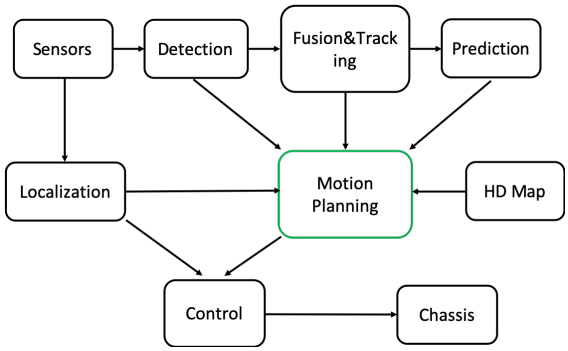

Fig. 4. The architecture of autonomous driving components.

In an autonomous driving system, the system layer is a middle-level connecting component layer and hardware/sensor layer. The system layer consists of sharing memory among components, synchronizing the data for algorithms usage based on sensors time, and security module.

At last, the hardware and sensor driver layer consists of computing hardware platform, embedded system hardware platform, sensor drivers, and corresponding protocol/software connecting them. The details has been described as in Sec. II.

B. Architecture of Motion Planning

To enable autonomous vehicles to complete the given missions and tasks, a hierarchical architecture is widely used (McNaughton et al., 2008). In this architecture, each mission is decomposed into sub-missions to complete hierarchically. However, this type of layered architecture has some performance problems. The major shortcoming of this framework is that the higher-level decision layer sometimes doesn't have enough information to perfectly guide the lower-level trajectory planning execution. On the other hand, a parallel architecture is also existed in various autonomous driving systems. Compared with the hierarchical framework, modules in this system are relatively independent and work in parallel. For example, lane-merge behavior, car-following behavior, and lane-keeping behavior are independently worked. In some complicated cases needing cooperation, this framework may not perform well.

As discussed above, there are shortcomings of both the current hierarchical and parallel planning architectures, In this paper, we proposed a novel planning framework that combines the strengths of the hierarchical and parallel architectures. The proposed motion planning architecture is comprised with route planning, scenario planning, and trajectory planning. Specially, scenario planning is based on parallel architecture and trajectory planning is for hierarchical architecture. In parallel framework, scenarios in this system are relatively independent and work in parallel. According to any specific scenario, a hierarchical framework is widely used to decompose the mission into multiple tasks and execute these trajectory related tasks sequentially. Thus, many complicated problems become solvable. Additionally, a HD map module designed specifically for last-mile delivery application is used as an input for proposed motion planning architecture. The proposed framework is shown in Fig. 5.

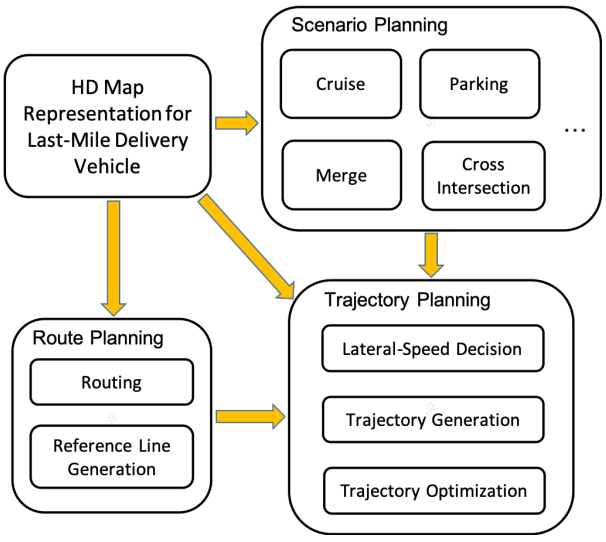

Fig. 5. The proposed motion planning framework.

Based on HD map design, the data from road network is used to create a node-edge graph. The waypoints in road are defined as nodes and the link between any two waypoints is the directional edge. These edges are also assigned costs like time and distance related value functions. The route planning is generated by classic graph search method. However, this routing result doesn't consider any motion requirements, so that, an nonlinear optimization method is required to optimize this routing result. Thus, the aims of route planning are to generate a feasible reference line by the map based graph search and then optimize the reference line for smoothness requirement.

The scenario planning is based on the concept of dividing the planning into a set of scenarios. The benefits of this architecture could be: first, increasing independence and resource sharing among scenarios; second, improving the efficiency for industrial level developments. Essentially, the scenario planning is used to choose a scenario based on current vehicle driving status gained from HD map, perception and localization results. To independently plan in different scenarios and meet unique requirements in last-mile delivery, the scenarios are customized designed by different traffic and driving conditions.

In term of last-mile delivery application, we designed a list of scenarios as follows: (1) crossing intersections scenario is designed for the query of all intersection elements including in stop lines, lane-road IDs, safety islands, pillars, etc; (2) traffic lights scenario aims at addressing the issue that last-mile delivery vehicles don't have enough time to pass through the intersections, due to their low speed; (3) left turn scenario is specifically designed for last-mile delivery vehicles. Since the driving speed is relatively low for last-mile delivery vehicles, so the vehicles usually keep driving on bicycle lane on the right hand side of the road. To prepare for a left turn in an intersection area, the vehicles have to move from bicycle lane to the far left hand side of the road. After that, vehicles need to adjust their orientations heading on the traffic lights; (4) backup on narrow road scenario handles with the situation that delivery vehicles drive on bicycle lane or unstructured area. If perceived obstacles are too close to plan a feasible trajectory for obstacle avoidance, then vehicles activate a backup scheme to produce a space to re-plan a reasonable trajectory; (5) zone parking scenario is similar as passenger vehicles' zone navigation. it deals with the point-to-point planning problem in unstructured environments like parking lots; (6) merging from off-road to routing scenario is designed for the switching process from parking or pull over status to the on-road status; (7) remote command from monitoring scenario is designed for manual intervention when the ego car is in stuck situation.

For example, we design a cruise scenario when the ego car is on-lane, design a cross intersection scenario when the ego car goes straight through the intersection with traffic regulations, and design a left turn and U turn scenario when the ego car's intention is left or U turn in front of the traffic lights, etc. Once a scenario is selected, a set of corresponding trajectory planning related tasks will be executed to complete this scenario. For instance, when the vehicle is driving under the cruise scenario, all behavior decision tasks including

path-speed based decision, obstacle based decision, traffic regulation based decision, and trajectory optimization tasks including path-speed optimization need to be executed sequentially within a motion planning cycle.

A set of tasks corresponding different scenarios can be classified with behavior decision tasks, trajectory generation tasks, and trajectory optimization tasks. The behavior decider focused on dealing with on-road traffic related decision, such as traffic lights decision, crosswalk area decision, stop sign decision, intersection decision, etc, and obstacles based decision. For obstacles based decision, both static and dynamic obstacles are considered into account to make lateral decisions like nudge and lane change and speed decisions like stop, yield, follow, etc. In addition, behavior decider not only makes general decisions, but generates a rough path and speed profile based on the trajectory-based decisions. After a rough path and speed profile for autonomous vehicle is generated, the trajectory optimizer applies optimization method to produce a smoother and human-like path and speed with consideration of the road boundary and kinematics/dynamics constraints of the autonomous vehicle.

IV. HD MAP DESIGN

In unpredictable real environments, autonomous vehicles should drive safely by detecting obstacles and recognizing existing landmarks on road. From the perspective of the motion planning, the map representation is significant, because a motion planner has to utilize a HD map which consists of detection and recognition results to generate optimal trajectories.

In order to efficiently handle the on-road detection and recognition results for the specific last-mile delivery case, we proposed a HD map representation by combining 16 layers to present both obstacles and static landmarks in urban environments. For the common purpose, this 16 layered map is designed geometrically and semantically. The geometric map representation in each map layer is expressed as fundamental elements like point, pose, polyline, polygon, respectively. On the other hand, the semantic expression of road elements is composed by identification (ID) and type. Therefore, all on-road detection and recognition results can be represented by both geometric and semantic expressions.

Our HD map design contains the definition of road geometries and properties, which is represented by 16 layers to provide both dynamic and static information of the environment. Moreover, Each layer has own geometric and semantic expressions. Like passenger vehicles, our last-mile delivery vehicle has many similar map representations for road elements, such as, the boundaries and type of each lane and road, the intersections with their semantic expression like crosswalks, traffic lights, speed bumps, etc, and the parking lot map representation with lane ID and geometric shape. However, our map handling has some differences with that of passenger vehicles. In addition to these common road elements expression, the last-mile delivery vehicle application, especially the operations in China, has some special road elements representation. For example, we designed a barrier gate layer representing an element to access distribution centers and gated communities; We designed a pillars layer, where these pillars are commonly used to prohibit motor vehicles' access and permit bicycles access only. Fig. 6(a) shows a typical map representation of an intersection in China, the blue dots in the map denote pillars which are used to prohibit vehicles to access; Moreover, we also designed a safety island layer. The safety islands shown in Fig. 6(a) are located in a large intersection area where pedestrians, bicycles, and our delivery vehicles could wait there for the next green light (compared with passenger vehicles, they don't have to consider this safety islands condition). For intuitive comparison, Fig. 6(b) shows a real safety island in a intersection.

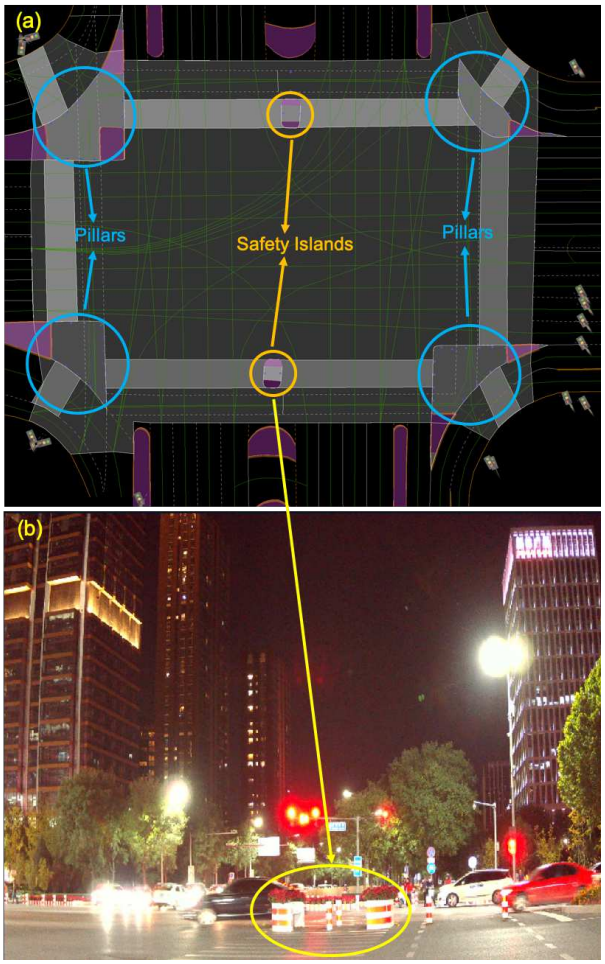

Fig. 6. ROVER 5.0's map representation for intersections: (a) is a map representation of an intersection with pillars and safety islands; (b) is a typical intersection photo in China.

In addition to traditional lanes and roads representation in HD map design, we also developed a novel lane group representation as an intermediate level between lane and road for better lane associations at large intersections. At large intersections, a lane's exit might correspond multiple other lanes' entrance, so it is necessary to define lane group to distinguish the group of lanes. As shown in Fig. 7, the frames

in red show multiple lanes in different roads are synthesized into a lane group. The following routing method is based on this lane group level configuration.

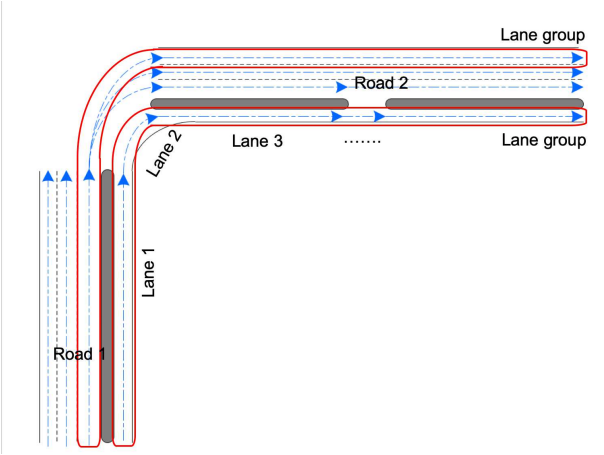

Fig. 7. Lane group representation in map design

V. ROUTE PLANNING

In autonomous driving, the aim of route planning is to generate an optimal route by map from starting point to destination. To produce the route plans, the data provided by road network from map are used to create a node-edge graph. In the proposed route planning method, the lanes in road network are considered as nodes, the connectivity between lanes or lane groups as shown in Fig. 7, are conceived as edges, and the cost of edges is defined by a combination of several factors, such as traverse time of the edges, distance between edges, complexity of the edge conditions, etc. Therefore, the route planning is to generate a feasible route by the map based graph search and smooth the route as a reference line for the following trajectory planning.

A. Routing

After completing the special HD map design for last-mile delivery application, the road network's representation described as Sec. IV is pretty well-defined. Then, the process of routing is to implement a graph search approach like $A^\star$ to find a shortest delivery path in given road network. Especially for last-mile delivery, the routing consists of single task routing, multiple tasks routing, and a lane based forbidden list design for routing cost. Where, single task in last-mile delivery means all orders belong to one customer and are sent to one destination, and multiple tasks denote that there exist multiple destinations to arrive.

1) Single Task Routing: When a single task is distributed to last-mile delivery vehicles, the routing strategy is to find a shortest route along a road network from vehicle's origin to a designated destination. As discussed in Sec. V, a road network is designed as a graph $G = (N, E)$ consisting of an indexed set of nodes N with $n = |N|$ nodes and a spanning set of directed edges E with $m = |E|$ edges. Each edge is represented as an ordered pair of nodes $E(i, j)$. The value $C(i, j)$ associated with each edge represents the cost incurred by traversing the edge. In this paper, lanes or lane groups are designed as nodes $N(i)$, the connectivity between lanes are considered as edges $E(i, j)$, and traverse time, distance between edges, complexity of the edge conditions are defined as cost $C(i, j)$.

Based on the above road network's definition and notation, a graph search based $A^\star$ method (Zeng & Church, 2009) can be readily used to generate the shortest single task route.

2) Multiple Tasks Routing: Compared with single task routing, multiple tasks operation widely exists in last-mile delivery applications. As shown in Fig. 8, a typical daily operation for last-mile delivery vehicles is that a delivery vehicle starts from a distribution center, and then stops by several transit stations, finally arrives to a designated destination.

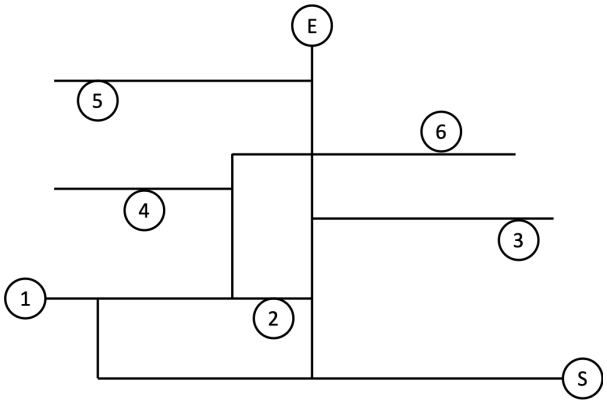

Fig. 8. Multiple destinations' routing for last-mile delivery

To generate an optimal multiple tasks route, we consider the multiple destinations delivery problem as a generalized asymmetric traveling salesman problem (Applegate, Bixby, Chvatal, & Cook, 2006). With no loss of generality, we consider starting point, transit stations, and final destination as a set of nodes N_i for $i = 0, 1, \cdots, n-1$ in road network, where $n = |N|$. Specifically, N_0 denotes the starting point, N_{n-1} denotes the final destination. Thus, the optimal route can be solved as following steps:

Step 1: Calculating the shortest path for any arbitrary two delivery stations by $A^\star$ approach, as discussed in Sec. V-A.1. For example, $\mathcal{D}_{i,j}$ for $i, j \in [0,\ n-1]$ denotes the shortest distance between the i^{th} delivery station to the j^{th} delivery station.

Step 2: Summarizing orders information (cost) $\mathcal{O}_i$ for $i = 0, 1, \cdots, n-1$, where the orders need to be sent to each delivery station. Typically, in last-mile delivery application, we want to minimize the vehicle's load during the delivery operations, which means delivering a large quantity orders in high priority. Additionally, some urgent goods is also delivered to specific station in high priority, such as fresh food, medical suppliers, etc. Thus, the order related cost can

be expressed as,

$$\mathcal{O}_i = w_0 \left(\frac{1}{N_i}\right)^2 + \sum_{j=1}^{N_i^u} w_j \left(\frac{1}{N_i^j}\right)^2 \tag{1}$$

where, N_i is the total number of orders sending to the i^{th} delivery station, N_i^u is the number of category for urgent goods in the i^{th} delivery station, and N_i^j is the number of orders for the j^{th} urgent goods category sending to the i^{th} delivery station. w_0 is weighting corresponding to the total number of orders, $w_1, w_2, \cdots, w_{N_i^u}$ are weightings corresponding to the number of urgent goods.

Step 3: Computing the total traverse cost from the i^{th} delivery station to the j^{th} delivery station as,

$$C_j^i = \mathcal{D}_{i,j} + \mathcal{O}_i + \mathcal{O}_j \tag{2}$$

All traverse costs between any arbitrary two nodes (assuming the number of nodes are equal to n) can be converted to the following matrix format,

$$C_M = \begin{bmatrix} 0 & C_1^0 & \cdots & C_{n-1}^0 \\ C_0^1 & 0 & \cdots & C_{n-1}^1 \\ \vdots & \vdots & \ddots & \vdots \\ C_0^{n-1} & C_1^{n-1} & \cdots & 0 \end{bmatrix} \tag{3}$$

Step 4: By using dynamic programming search, we can find minimum cost based Hamiltonian cycle in the cost matrix C_M from Eq. (3), so that the optimal delivery sequence for multiple tasks can be readily solved.

3) Forbidden List Design: For route planning in last-mile delivery application, we also proposed a unique forbidden list design to address the problem of local lane's variations. Compared with passenger vehicles, last-mile delivery vehicles often operate in irregular traffic environments. For example, ROVER 5.0 sometimes drives in the bicycle lanes, sidewalks, campus, or residential communities. These irregular traffic environments, however, often varies very frequently. Thus, the path produced by route planning based on HD map might be infeasible for driving.

In order to generate feasible routes under irregular traffic environments, we designed a forbidden list inserting all impassable lane ID due to traffic environment changes. By maximizing the weights of lanes in the forbidden list, the routing method automatically bypasses all lanes in forbidden list, then generate optimal routes under current traffic environments.

In Fig. 9, we draw a flowchart describing how route planning algorithms should interact with the dynamic changing environment like road networks and adapt the best routes assigned to a vehicle according to the updates (i.e. change in forbidden list, congestions level, incidents, etc.) received from the traffic management systems.

B. Reference Line Smoothing

In last-mile delivery application, routing result is considered as a reference line for the subsequent trajectory planning and control. As mentioned as Sec. V, reference

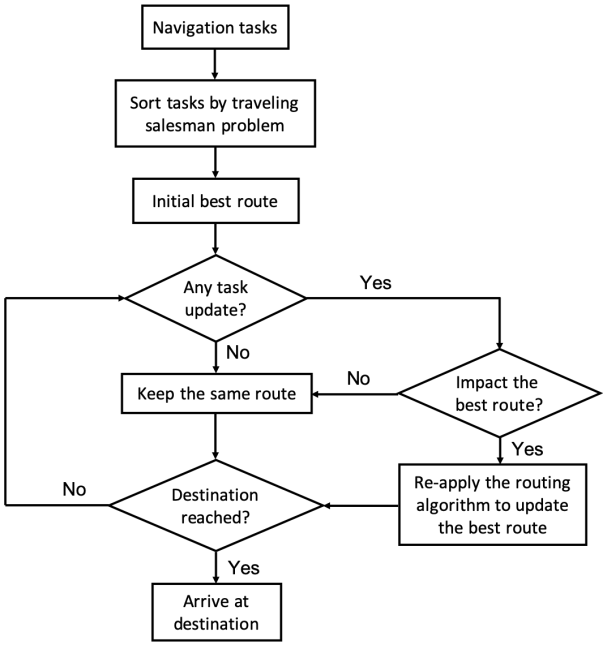

Fig. 9. Flowchart of the best route update during environment changing

line is the foundation of the whole planning module. In each planning cycle, a reference line is generated at first, then such like obstacle projection, traffic rule logic, path-speed based decision and optimization are produced by the reference line. However, the raw reference line, that is routing, is from lane represented by waypoints in HD map, so these points from reference line are not smooth enough to the smoothness requirements for the trajectory planning and control modules. Consequently, reference line smoothing is required for planning and control.

The routing results generated by Sec. V-A are discrete waypoints and not smooth. In these discrete waypoints, each two adjacent points can determine a straight line, where the equation of this line is represented as,

$$\mathcal{L}_s : A_i^s x + B_i^s y + C_i^s = 0 \tag{4}$$

In the meanwhile, a perpendicular line passing through the midpoint of this straight line as shown in Fig. 10,

$$\mathcal{L}_p : A_i^p x + B_i^p y + C_i^p = 0 \tag{5}$$

where, A_i^s, B_i^s, C_i^s are the coefficients of the straight line $\mathcal{L}_s$ given by the $i-1^{th}$ and i^{th} waypoints, A_i^p, B_i^p, C_i^p are the coefficients of the corresponding perpendicular line $\mathcal{L}_p$ passing through the midpoint of the straight line $\mathcal{L}_s$, and x, y are defined in Eq. (6).

To smooth the routing results, the optimized states X are designed as,

$$X = [x \quad y \quad \theta \quad \phi]^T \tag{6}$$

where, x and y are the coordinates of the center of mass in an inertial frame, θ is the inertial heading, ϕ is the front steering angle. Then, the objective function can be designed as a linear combination of smoothness cost and

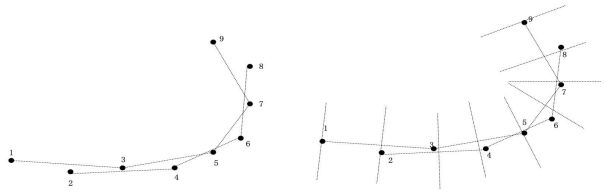

Fig. 10. Expression of lon and lat line segments

the cost of deviation from raw routing result. By setting weighting functions, the costs above can be minimized and compromised. Thus, mathematically, the objective function can be formulated as an optimization problem,

$$\min_{X\in\Re^N} J(X), \text{ where}$$
$$J(X) \triangleq \sum_{i=1}^{N} w_0(\phi_i - \phi_{i-1})^2 + w_1 \frac{(A_i^s x_i + B_i^s y_i + C_i^s)^2}{A_i^{s2} + B_i^{s2}} \tag{7}$$

where, the first item of the objective function is the change of front steering angle, which reflects the smoothness, and the second item of the objective function is the distance from the optimized coordinates (x, y) at each sampling point to the corresponding straight line $\mathcal{L}_s$ in Eq. (4). w_0 and w_1 are the weightings.

The constraints for reference line smoothing are comprised with vehicle kinematics constraints, road related constraints, and vehicle physical limitation constraints. At relatively low speed and fixed planning frequency, we want reference line can succinctly capture the motion of the vehicle using geometry. In this paper, we considers a kinematics bicycle model (Rajamani, 2011), then all optimized states meeting vehicle kinematics constraint are described as,

$$\begin{aligned} x_{i+1} &= x_i + v_i \cos\theta_i T \\ y_{i+1} &= y_i + v_i \sin\theta_i T \\ \theta_{i+1} &= \theta_i + \frac{v_i \tan\phi_i}{l} T \end{aligned} \tag{8}$$

where, the definitions of x, y, θ, ϕ are referred as Eq. (6). v_i is the linear velocity at the i^{th} sampling point, T is the sampling time. Additionally, the road related constraints are as following. The optimized reference line is required to within the road width, and the optimized states (x, y) coordinates must lie on each perpendicular line $\mathcal{L}_p$ between two adjacent row waypoints.

$$\begin{aligned} -d < \frac{A_i x_i + B_i y_i + C_i}{\sqrt{A_i^2 + B_i^2}} < d \\ A_i^p x_i + B_i^p y_i + C_i^p = 0 \end{aligned} \tag{9}$$

where, d denotes the maximum deviation from row routing results. Then, the vehicle physical limitation constraints consist of velocity constraint and front wheel steering angle constraint as following,

$$\begin{aligned} v_{low} < v_i < v_{high} \\ \phi_{low} < \phi_i < \phi_{high} \end{aligned} \tag{10}$$

After applying all constraints, the reference line optimization problem's solution can be readily obtained by some nonlinear optimization solver. For comparison, the quadratic programming (QP) based quintic polynomial fitting method was also applied to design and smooth the raw reference line. Readers are referred to (*Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving*, n.d.) for the details of the QP based polynomial optimization design. In this experiment, a raw routing result is shown in Fig. 11 (a), and the total length of the routing result is around 2,200.00m. Smoothing results at two different methods were evaluated in this road test. We truncated two different portions of the road test which are compared in Fig. 11 (b), (c), where blue path is the raw reference line, green path is the optimized path obtained by using QP based polynomial approach, and pink path is the optimized path obtained by the proposed approach. Moreover, the curvature of the smoothed reference line is compared in Fig. 11 (d), (e), where blue line denotes the curvature obtained by QP based polynomial approach, and red line denotes the curvature obtained by the proposed approach.

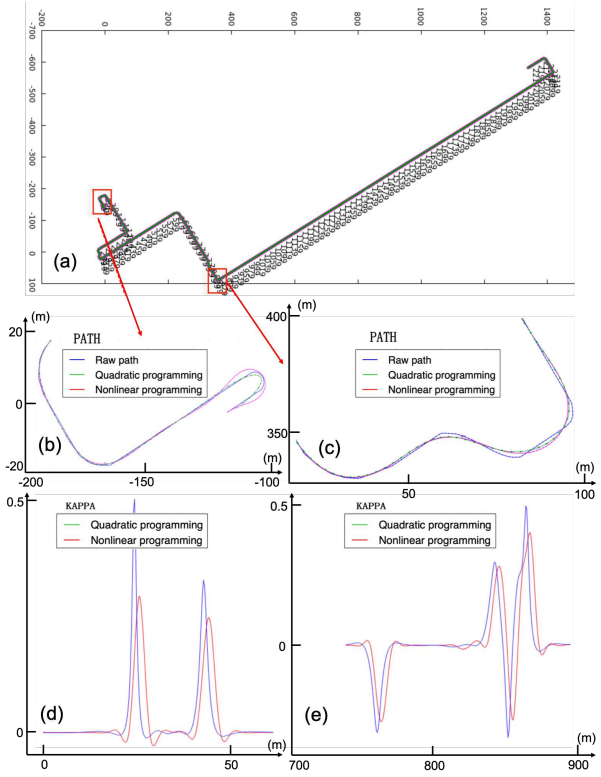

Fig. 11. A 2,200m's long raw reference line is shown at (a), comparison of the smoothing results from two truncated portions obtained by using quadratic programming and nonlinear programming at (b) and (c), and comparison of the corresponding curvatures at (d) and (e), respectively (maximum curvature can be dramatically reduced by using our proposed approach).

The road test result show that by using the proposed nonlinear optimization technique, both outstanding smoothness and satisfied vehicle kinematics requirements can be maintained throughout the entire reference line. Whereas by using

the QP based polynomial method, although good smoothness requirement can be achieved, the smoothing results don't meet the vehicle kinematics requirements, especially while making turns. Such sharp turns were dramatically reduced by using the proposed technique, as shown in Fig. 11 (b), (c). The optimized results' curvature comparison in Fig. 11 (d), (e) also show that by using the proposed nonlinear optimization technique, the curvature of the smoothing reference line were dramatically reduced (maximum curvature is reduced from 0.5 to 0.3). Therefore, the road results demonstrate the efficacy of the proposed technique for reference line smoothing.

VI. SCENARIO PLANNING

The scenario planning architecture is used to choose a scenario based on current vehicle driving status gained from route planning and a set of environmental features. The scenarios are customized designed by different traffic and driving conditions. For example, In autonomous driving practice, we design a cruise scenario when the ego car is tracking on-lane, design a cross intersection scenario when the ego car passes through the intersection with traffic regulations, etc. Once a scenario is selected, a set of corresponding tasks will be executed to complete this scenario. Additionally, in high level design, the corresponding special scenarios are, respectively, left turn based on traffic light, backup on narrow road, parking in a zone, and merging from off-road to route in last-mile delivery applications. When the vehicle is driving under the specific scenario, all behavior decision tasks including path-speed based decision, obstacle based decision, traffic regulation based decision, and trajectory optimization tasks are executed sequentially within a motion planning cycle.

A. Parking scenario

Unlike trajectory planning on road discussed in the following Sec. VII, to efficiently generate a smooth trajectory to a parking goal pose in unstructured zones existing obstacles, we used a well known hybrid $A^\star$ algorithm (Dolgov, Thrun, Montemerlo, & Diebel, 2008) applied to the kinematic state space of the vehicle in the first stage. Compared with traditional $A^\star$ which only allows visiting centers of cells, the hybrid $A^\star$ associates with each grid cell a continuous states of the vehicle. As known, the hybrid $A^\star$ is not guaranteed to find the optimal solution, because of the continuous states constraints. The paths produced by hybrid $A^\star$ are often still sub-optimal and not guaranteed to be drivable. In order to generate drivable paths, we used the conjugate gradient descent method (Dolgov et al., 2008) to locally improve the paths, which is locally optimal, and usually attains the global optimum as well.

Fig. 12 illustrates the motion planning in a typical unstructured environment. Especially, this figure shows a parking task when last-mile delivery vehicle came back to the distribution center. The red arrow denote the designated parking goal pose for our vehicle's parking task. The red line is the open space path generation toward parking spot, and the green line is the desired trajectory generated and optimized by hybrid $A^\star$ and conjugate gradient descent methods.

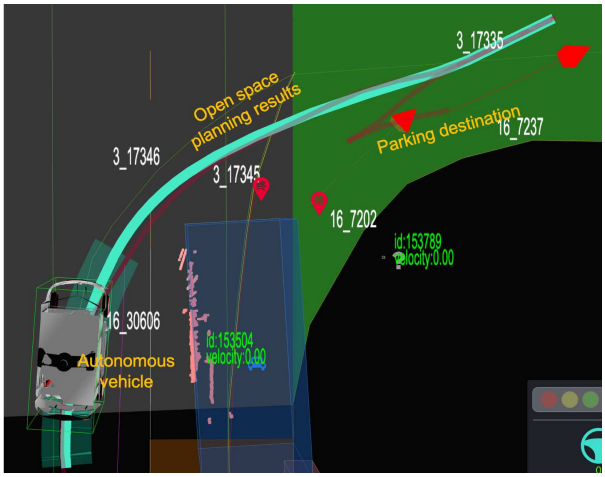

Fig. 12. An example of the motion planning for a typical parking scenario.

B. Merge from off-road to routing scenario

As mentioned in Sec. VI-A, parking and resuming to drive on roads are frequent behaviors in last-mile delivery. The merging behavior from parking spots to roads is discussed in this section.

The merging behavior can be comprised with two stages. The first stage is to find a goal pose based on current vehicle position and reference line, and the second stage is to generate a trajectory from current vehicle position toward the goal pose. For the first stage, a provisional goal point can be computed by projecting the current vehicle's position to the target reference line. If the provisional goal point is not occupied by obstacles, then we set it as final goal point, otherwise, we can search appropriate goal point along with the reference line with a fixed step.

Once the goal point is determined, the merge trajectory from off-road to reference line can be generated by zone trajectory planning, referred as in Sec. VI-A or on-road trajectory planning, described details in Sec. VII. When the angle difference between the current vehicle's heading and target reference line's heading is too large, zone trajectory planning is applied for merge trajectory, such as the behavior from parking lot back to the road. On the other hand, when the current vehicle's heading is aligned to target reference line, on-road trajectory planning method is used to produce the merge trajectory, such as, the vehicle which is pulled over roadside tries to be back to the road.

C. Crossing intersections scenario

Within the geometric map, an intersection contains intersection ID, geometric shape, lane and road ID binding with the intersection, as shown in Fig. 6. In each motion planning cycle, the system keeps getting the current intersection information of interest, and based on the route planning results and traffic lights status, determining whether the vehicles are

going to cross the intersection. Once the crossing intersection scenario is determined, all intersection elements, such as stop line, lane-road IDs, speed limits in intersection, pillars, safety islands, discussed in Sec. IV, are obtained based on the intersection shape. These data are known in advance of arrival at the intersection, and are completely static.

In contrast to the information gained from the intersection elements, the moving obstacles data received periodically by perception are highly dynamic. However, the obstacles tracking and prediction through an intersection need additional intersection elements constraints. During the process of intersection crossing, the trajectory planning in the following Sec. VII is determined by both intersection elements and moving obstacles.

D. Left turn scenario

The aim of left turn scenario is to adjust last-mile vehicle's orientation heading to traffic lights at intersections. Logically, left turn scenario is transited from crossing intersections scenario. When vehicles complete to adjust orientation in left turn scenario, the scenario will be switched back to crossing intersections scenario for passing through intersections. Compared with the passenger vehicle, the last-mile delivery vehicle has much slower driving speed, so a delivery vehicle keeps driving on the lane together with pedestrians and bicycles. When route planning produces a route for the trend to turn the vehicle left through a intersection, the vehicle needs to cross an intersection and waits in front of the stop line for traffic lights. After this transition, the vehicle might not perceive traffic lights due to its orientation. Therefore, we designed the left turn scenario to adjust vehicle's orientation in front of the stop line. Fig. 13 can show the special left turn scenario for delivery vehicles. Firstly, the red arrow is the goal pose, and its optimal position is calculated based on stop line's position and obstacles' cost, and the goal's orientation is headed by the goal position to the traffic lights position; Secondly, once a goal is determined, hybrid $A^\star$ algorithm can generate a feasible path to drive vehicles to that goal, where can perceive the traffic lights; Thirdly, when traffic light is green, the left turn scenario will change back to cross intersection scenario, then vehicles finish to go through the intersection.

E. Traffic lights scenario

As mentioned in Sec. VI-D, the primary property of last-mile delivery vehicle is low speed. This slow speed property leads to vehicles not have enough time to pass through the intersections straightly or by left turn, when traffic light turns from green to red. Thus, we need a different strategy when vehicles position in front of traffic lights at road intersections.

It is obvious that vehicles stop on stop-line at the road intersections when traffic light is red. However, there exists two options (keep stopping or drive) when traffic light becomes green. To avoid the situation that vehicles still drive in the middle of intersections even though traffic light turns to red, we designed a traffic lights based scenario decision. In our ROVER 5.0's operations, the HDR camera

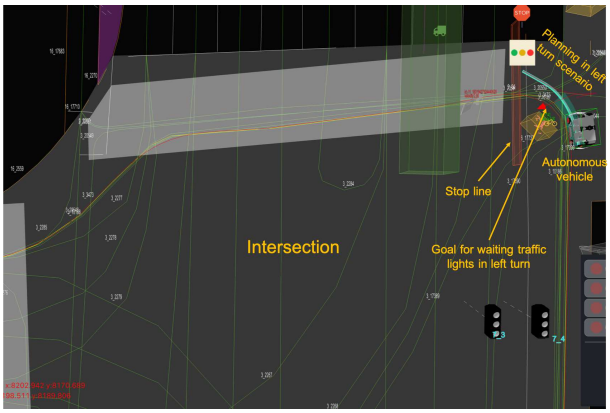

Fig. 13. Left turn scenario for the last-mile delivery vehicles

listed in Table. I perceives traffic lights of interest. Once the green light is detected, then the timer is initiated to record the duration until the vehicle arrives at stop-line. When the duration exceeds some pre-defined time threshold, the vehicle will stop at stop-line to wait for the next green light.

F. Back up scenario

Essentially, back up scenario is designed for the error recovery of vehicles on roads or intersections. The most commonly encountered recovery situations occur when the vehicle is blocked because of sudden perceived obstacles, or the vehicle stops beyond the stop line in intersections.

1) On-Road Back Up: For back up scenario triggered by obstacles, when the planning trajectory could not drive vehicles bypass obstacles, due to trajectory planning problems or obstacles perception issues, the planning status is switched from cruise scenario to back up scenario. For example, various situation induced by transient irregular obstacles can trigger this scenario, e.g., (1) traffic cones, bicycles, which are detected as wrong sizes, induce incorrect planning results; (2) some obstacles like barriers, other cars, or overhanging branch that are detected too late to bring the ego care to stuck.

When the ego car is stuck by obstacles on road, the algorithm for on-road back up behavior selects an initial forward goal along the reference line with some distance forward $s_{initial}$ from ego car's position. If the initial forward goal is occupied by other obstacles, then the alternative goal can be searched by adding some incremental distance $s_{incremental}$, until some maximum distance s_{max}. Similarly, the algorithm also chooses a back up distance s_{back} along with the reference line behind the ego car. Empirically, $s_{initial} = 10$m, $s_{incremental} = 1$m, $s_{max} = 15$m, and $s_{back} = 6$m worked well for last-mile delivery vehicles.

To obtain a back up trajectory, we initially used the current reference line with some offset obtained by current vehicle's lateral distance from reference line. If the back up trajectory based on reference line is infeasible, then the algorithm can generate a new back up trajectory using the same method as discussed in Sec. VI-A. In the meantime,

trajectory planning module continues to plan the feasible forward trajectory to bypass obstacles based on a given forward goal. When planned trajectory is constantly not overlapped with obstacle's bounding box at some number of planning cycles, the system will switch the back up scenario to cruise scenario for the normal operations.

2) On-Intersection Back Up: Similar with On-Road back up scenario in Sec. VI-F.1, when ego car crosses the stop line on intersection scenario and traffic lights show red signal, the system automatically switches scenario from intersection to back up. All back up configurations are the same as those of on-road back up. Once traffic lights turn to green or the ego car doesn't cross the stop line, then the system quits back up scenario.

VII. TRAJECTORY PLANNING

The proposed motion planner is depicted in Frenet frame that the reference frame is given by the tangential and normal vector at each path point based on the reference line (Werling et al., 2010). The reference line here is roughly the road center or desired path from routing module. Rather than generating trajectory directly in Cartesian frame, we map the obstacles and the ego car states with position and heading $(x,\ y,\ \theta)$ to the Frenet frame $(s,\ l,\ l^{(1)},\ l^{(2)},\ l^{(3)},\ l^{(4)})$ with respect with time t, which represent longitudinal station, lateral displacement, and its lateral derivatives. In Frenet frame, the trajectory of ego car and obstacle can be described in SL $(l(s))$ and ST $(s(t))$ graphs. In motion planning, we can evaluate the static obstacles, estimated dynamic obstacles from prediction module, and ego cars' positions at each time instant. Then, the obstacles and ego car at each time instant are projected on the reference line in Frenet frame. Thus, the relationship between ego car and obstacles are pretty clear in SL and ST graphs (Werling et al., 2010).

The proposed motion planning approach comprises two main parts as following.

A. Trajectory Generation

1) Path Generation and Decision: The lateral decision not only produces the obstacles based vehicle lateral decision, but generates a rough path profile by a dynamic programming search. As shown in Fig. 14, the lateral decision module mainly comprises the following four steps.

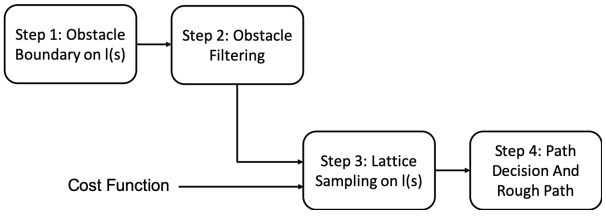

Fig. 14. Flow chart of the lateral decision technique.

Step 1 (Obstacles Boundary On $l(s)$): Based on smooth reference line as discussed in Sec. V-B, the obstacles and ego cars described with location and heading are mapped to the Frenet frame coordinates, which can produce $l(s)$ boundaries represented as a lateral displacement l with respect to station s. For example, assuming there is a smooth reference line R composed of n points, which can be partitioned into $n-1$ line segments as follows,

$$seg_i = \overrightarrow{R_i R_{i+1}},\ i = 1,\ 2,\ \cdots,\ n-1 \tag{11}$$

where, R_i denotes the i^{th} reference point, and seg_i denotes the i^{th} line segment. Without loss of generality, to map an obstacle from Cartesian coordinates (x, y) to Frenet coordinates (s, l), as shown in Fig. 15, firstly we need to find the nearest line segment (assuming the k^{th} line segment) from all line segments based on the obstacle's position $P(x, y)$ in Cartesian coordinates. Secondly, connecting the k^{th} line segment (seg_k)'s start point R_k with a terminal point P constructs a vector $\overrightarrow{R_k P}$. Similarly, the k^{th} line segment (seg_k) is denoted by $\overrightarrow{R_k R_{k+1}}$ in Eq. (11). By vector projection formula, the scalar projection s and scalar rejection l can be readily obtained by,

$$\begin{aligned} s &= \frac{\overrightarrow{R_k P} \cdot \overrightarrow{R_k R_{k+1}}}{\left\|\overrightarrow{R_k R_{k+1}}\right\|} \\ l &= \frac{\left\|\overrightarrow{R_k P} \times \overrightarrow{R_k R_{k+1}}\right\|}{\left\|\overrightarrow{R_k R_{k+1}}\right\|} \end{aligned} \tag{12}$$

where, '$\cdot$' denotes inner product and '$\times$' denotes cross product. Thus, given obstacles and ego car's positions in Cartesian coordinates and a smooth reference line, we can compute all obstacles' SL boundaries in Frenet coordinates. Building $l(s)$ boundaries is very important, because it illustrates the relationship between obstacles and ego cars, and provides evidence for the following lateral decision and path generation.

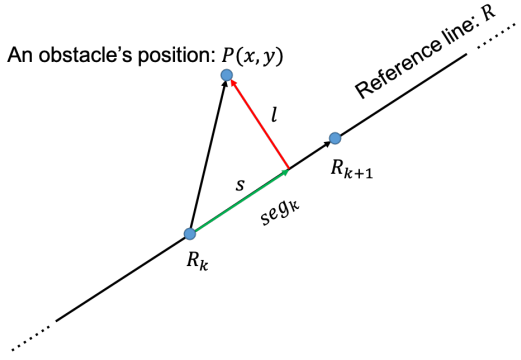

Fig. 15. (x, y) in Cartesian coordinates transforms to (s, l) in Frenet coordinates.

Note: Obstacles projection to Frenet frame highly depends on the smoothness of reference line. If the curvature of the reference line is too large, such as U turn shape of reference line, the obstacles mapping may produce ambiguous outcomings, that is, a lateral displacement l respects to multiple different stations s in Frenet frame. To address this type of

ambiguity, we partition the entire reference line with subsets according with U turns as,

$$ref(\cdot) = \cup_k ref_{p_k}(\cdot) \tag{13}$$

where $ref_{p_k}(\cdot)$ is defined as the k^{th} sub-reference line, $k = 1,\ 2,\ \cdots,\ n$, and n is the number of partitions. Assuming there exist m obstacles and $n-1$ U turns in reference line, which partitions the entire reference line into n sub-reference lines. Then, we can project the all m obstacles onto the n sub-reference lines to produce $m \times n$ SL boundaries. Thus, these $m \times n$ SL boundaries are considered independently to the following lateral decision making and rough path generation.

Step 2 (Obstacles Filtering): For any time instant, we assume that all obstacle elements are constructed to a set $\mathcal{O}$, which is given by

$$\mathcal{O}(t) = \{obj_{k,t} \mid k = 1,\ 2,\ \cdots,\ N;\ t \in \Re\} \tag{14}$$

where N is the total number of obstacle elements perceived by perception component.

In order to ignore some irrelevant obstacles that don't impact planning results, we define a planning region of interest (ROI) by some criterion. In this article, a s-l representation based region relative with reference line in Frenet frame can be considered as the ROI, as shown in Fig. 16. Without loss of generality, we assume the reference line is

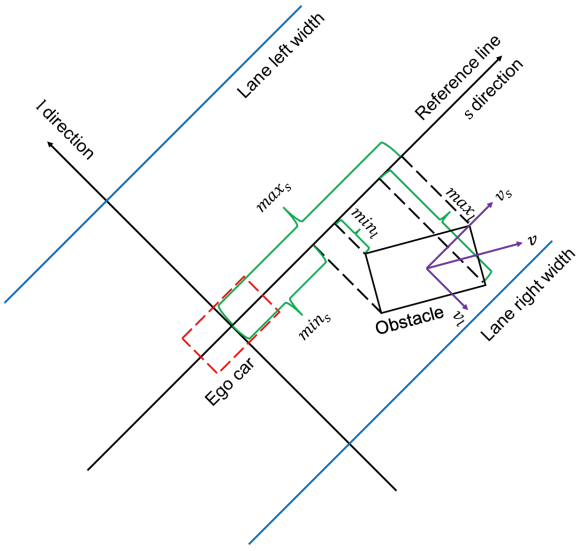

Fig. 16. A schematic diagram for obstacle filtering.

middle line of the lane, the planning length in each planning cycle is defined as L_p, and any obstacle is represented as a rectangular bounding box. From the Fig. 16, four vertexes of an obstacle are mapped onto reference line to produce min_s and max_s relative with ego car, and produce min_l and max_l relative with reference line. On the other hand, if the obstacle is dynamic, then the velocity of obstacle v from prediction module can be orthogonally decomposed as v_s and v_l. Thus, the criterion of obstacles filtering are: (1) Obstacle's s-directional boundaries satisfy the conditions in Eq. (15), that means if the obstacles are far away from planning distance or locate in the behind of ego car, we can ignore them for planning.

$$\begin{aligned} min_s &> P_s + L_p + \epsilon_f \\ max_s &< P_s - \epsilon_r \end{aligned} \tag{15}$$

where, P_s denotes the ego car's s-coordinate along with reference line, ϵ_f and ϵ_r are the additionally longitudinal distance buffers in front and behind of ego car. (2) Obstacle's l-directional boundaries satisfy the conditions in Eq. (16), that means if the obstacles go beyond lane's boundaries, we can ignore them for planning.

$$\begin{aligned} min_l &> max\left\{\mathbf{W}_l(min_s), \cdots, \mathbf{W}_l(max_s)\right\} - \epsilon_l \\ max_l &< max\left\{\mathbf{W}_r(min_s), \cdots, \mathbf{W}_r(max_s)\right\} + \epsilon_r \end{aligned} \tag{16}$$

where, $\mathbf{W}_l$ and $\mathbf{W}_r$ are functions of lane's left and right boundary, ϵ_l and ϵ_r are buffers for obstacles filtering in l direction. (3) Additionally, the prediction of dynamic obstacles are also considered here. At any time instant, an obstacle's velocity $\overrightarrow{v}$ can be estimated by prediction, as shown in Fig. 16. Based on orthogonal decomposition method, obstacle's velocity $\overrightarrow{v}$ is decomposed as vectors $\overrightarrow{v_s}$ and $\overrightarrow{v_l}$ in s-l frame. When the direction of $\overrightarrow{v_s}$ is along with reference line, and the magnitude of $\overrightarrow{v_s}$ is larger than maximum speed of the delivery vehicle, then we can ignore the corresponding obstacles. On the other hand, for $\overrightarrow{v_l}$, when the magnitude of $\overrightarrow{v_l}$ is larger than a lateral velocity threshold (empirically, 0.4m/s for ROVER 5.0 vehicle), the corresponding obstacles can be filtered out.

As a result, those irrelevant obstacles for motion planning are filtered and the processing time could be significantly reduced. The filtered obstacles subset is given by

$$\mathcal{O}_f(t) = \{obj_{k,t} \mid k = 1,\ 2,\ \cdots,\ N_f;\ t \in \Re\} \tag{17}$$

where N_f is the total number of obstacle elements of interest after filtering, and $\mathcal{O}_f(t) \subset \mathcal{O}(t)$.

Step 3 (Sampling On SL Graph): For lattice sampling, multiple rows of points are firstly sampled in front of ego car under Frenet frame. As shown in Fig. 17, the colors of sampling points denote the corresponding costs. Red end of the color spectrum denotes the higher cost, and violet end denotes the lower cost. The bounds of sampling depend on the current passable areas where the vehicles drive on. For each row's sampling, the choice of resolution is based on lane boundaries and obstacles' position. Sampling points between rows are smoothly interpolated by polynomial edges. Notice that the resolution of the sampling between rows highly depends on vehicle speed and driving scenarios, so it can be customized by different application cases. For instance, when the ego car tries to go through a very narrow road junction, the sampling resolution could be set higher than normal.

Step 4 (Decision and Rough Path Generation): When all lattice sampling based nodes and edges are constructed, the rough path result can be generated by a connection of

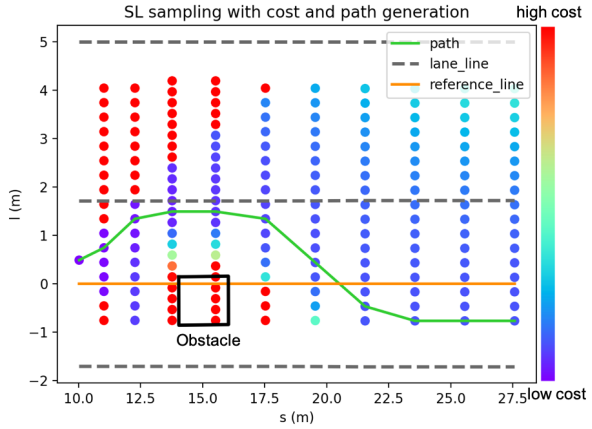

Fig. 17. An illustration of sampling on SL graph and path generation. Note: the red end of the color spectrum denotes the higher cost, and the violet end denotes the lower cost.

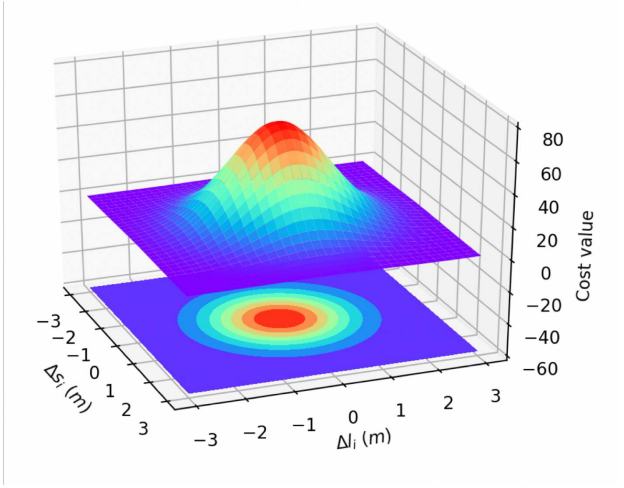

Fig. 18. Schematic diagram of obstacle based cost function.

selected edges which are evaluated to have minimal cost functions. Each node's cost function is a linear combination of the obstacle avoidance cost and path cost including smoothness cost and deviation from reference line cost.

$$J(s,l) = J_{obj}(s,l) + J_{path}(s,l) \tag{18}$$

For the obstacles impact for path based decision and generation, we only consider the obstacles which locate in the range of interests (ROI). For example, the obstacles' S-L boundaries that are within some ROI boundaries are considered in path generation. Then, the obstacle avoidance cost $J_{obj}(s,l)$ is given by,

$$J_{obj}(s,l) \triangleq \sum_{i=0}^{N} w_c \mathcal{G}(\mu_{\Delta s}, \mu_{\Delta l}, \sigma_{\Delta s}, \sigma_{\Delta l}, \Delta s_i, \Delta l_i) \tag{19}$$

where, N denotes the number of nodes, w_c is the weighting of obstacle avoidance cost, $\mathcal{G}(\mu_{\Delta s}, \mu_{\Delta l}, \sigma_{\Delta s}, \sigma_{\Delta l}, \Delta s_i, \Delta l_i)$ is the two-dimentional Gaussian function,

$$\begin{aligned} &\mathcal{G}(\mu_{\Delta s}, \mu_{\Delta l}, \sigma_{\Delta s}, \sigma_{\Delta l}, \Delta s_i, \Delta l_i) \\ &= \frac{1}{\sigma_{\Delta s}\sigma_{\Delta l}\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{(\Delta s_i - \mu_{\Delta s})^2}{\sigma_{\Delta s}^2} + \frac{(\Delta l_i - \mu_{\Delta l})^2}{\sigma_{\Delta l}^2}\right)} \end{aligned} \tag{20}$$

where Δs_i is the longitudinal distance between the i^{th} node and the obstacle's bounding box; Δl_i is the lateral distance between the i^{th} node and the obstacle; $\mu_{\Delta s}$ and $\mu_{\Delta l}$ are the expectation values of Δs and Δl; $\sigma_{\Delta s}$ and $\sigma_{\Delta l}$ are the standard deviations of Δs and Δl.

By Eq. (20), we can see 2D Gaussian function is a characteristic symmetric "cone" shape, as shown in Fig. 18. Thus, the corresponding obstacle cost function J_{obj} is a monotonically decreasing function, along with longitudinal and lateral distance between ego car and obstacles increasing.

In addition to the obstacle avoidance cost $J_{obj}(l(s))$, the path cost $J_{path}(l(s))$ is given by,

$$\begin{aligned} J_{path}(s,l) \triangleq \sum_{i=0}^{N} &w_0(l(s_i) - r(s_i))^2 + w_1(l^{(1)}(s_i))^2 \\ &+ w_2(l^{(2)}(s_i))^2 + w_3(l^{(3)}(s_i))^2 \end{aligned} \tag{21}$$

where, $[l(s_i),\ l^{(1)}(s_i),\ l^{(2)}(s_i),\ l^{(3)}(s_i)]$ are the lateral displacement and its corresponding derivatives relative with the i^{th} station s_i, $r(s_i)$ is the lateral displacement of reference line, and w_i for $i = 0,\ 1,\ 2,\ 3$ are the weightings.

Thus, the node cost $J(s,l)$ can be obtained by a combination of path related cost and obstacle related cost by Eq. (18). Thus, the final rough path can be synthesized by these corresponding edges with the minimal cost through a dynamic programming (DP) search. Note: (1) In addition to consider the node cost $J(s,l)$ by DP search, the curvature limitation also needs to be considered. In practical DP search, the nodes connection in adjacent level is restricted by curvature limitation; (2) If all of the space are blocked by obstacles and no room to pass through for delivery vehicles, the DP search still can find a solution to generate an infeasible path across the obstacles. The following decisions can be made by speed planning described in Sec. VII-A.2, such as stop, yield, following, etc.

As shown in Fig. 17, the black rectangle denotes an obstacle on lane, and the colored dots denote the sampling points in front of an ego car. Each sampling point's cost is computed by Eqs. (18) (19) (21) and the value of cost corresponds with different color. Here, red end of the color spectrum denotes the higher cost, and violet end denotes the lower cost. The computation of cost is a trade-off among obstacle avoidance, minimum deviation from path to reference line, path smoothness, and curvature limitations. From Fig. 17 we can see the color of sampling points which is near by an obstacle, far away from reference line, or making path curvature larger are red. Thus, the final optimal path is generated by connecting minimum cost of points in

each row. In the meanwhile, according to the relationship between the path points and obstacle positions, the path also makes the obstacle decisions.

2) Speed Profile Generation and Decision: The speed decision module produces a decision for speed relevant behavior, as well as generates a rough speed profile represented as a station s function with respect to time t, that is, $s(t)$ representation. Similar as lateral decision algorithm described in VII-A.1, the speed decision also consists of four steps, as shown in Fig. 19.

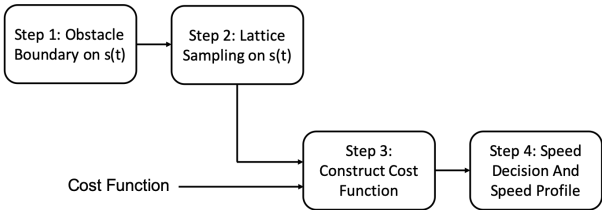

Fig. 19. Flow chart of the speed decision technique.

Step 1 (Obstacles Boundary On $s(t)$): Firstly, the obstacles' prediction trajectories are projected on planned rough path in Frenet frame, which can produce $s(t)$ boundaries which are represented as a station s with respect to time t for all obstacles. If the obstacles' trajectories have interactions with planned path with some threshold (such as ego car's width), then the corresponding blocking intervals will be generated on $s(t)$ frame.

Without loss of generality, we assign all blocking sections T_k for $k \in \mathbb{N}$ ($\mathbb{N}$: the set of natural numbers), to be closed,

$$T_k = [t_{k,i},\ t_{k,f}], \quad k \in \mathbb{N} \tag{22}$$

where $t_{k,i}$ and $t_{k,f}$ are defined as the initial instant and the final instant for the k^{th} blocking section, respectively. Thus, the entire obstacles based blocking intervals in Frenet frame is a well-defined function of time, and is partitioned into multiple intervals $s_{blk}(\cdot)$, i.e.,

$$s_{blk}(\cdot) = \cup_k s_{blk,k}(\cdot) \tag{23}$$

where $s_{blk,k}(\cdot)$ are defined for the k^{th} blocking section T_k, and $k = 1, 2, \cdots$.

However, computing obstacles ST boundaries is a very time-consuming task, especially for dynamic obstacles. As shown in Fig. 20, to calculate dynamic obstacles ST boundaries, for each prediction trajectory point, the minimum distance between each vertex of an obstacle polygon and the path points should be computed. The time complexity is up to $O(kn^2)$, where k denotes the number of vertex for the obstacle polygon. In order to save more time in each planning computation cycle, a k-d tree space-partitioning data structure was proposed to address the time-consuming problem for obstacles ST boundaries computation. In practice, it is nearly always used to support search on multi-dimensional coordinates, such as 2D space in autonomous driving. In this paper, we construct k-d tree based on path points data. Because we store path points data organized by

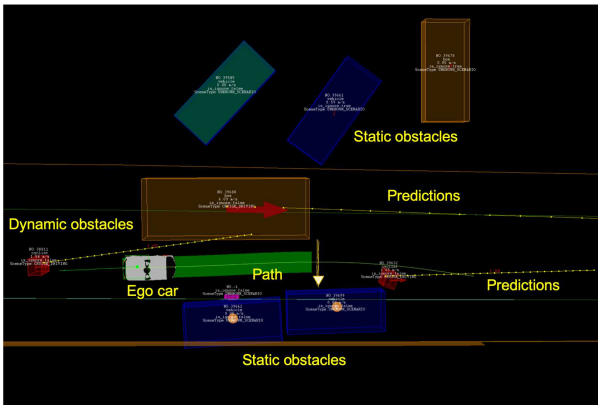

Fig. 20. Example of the autonomous driving scenario with dynamic obstacles ST boundaries computation.

xy-coordinates, in this case, k is 2, with the x-coordinate field arbitrarily designated key 0, and the y-coordinate field designated key 1. At each level, the split direction alternates between x and y. Thus, a xy-coordinates node at level 0 (x direction split) would have in its left subtree only nodes whose x values are less than that of root. The right subtree would contain nodes whose x values are greater than that of root. a xy-coordinates node at level 1 (y direction split) can be done in the same manner. Fig. 21 shows an example of how a collection of two-dimensional path points would be stored in a k-d tree.

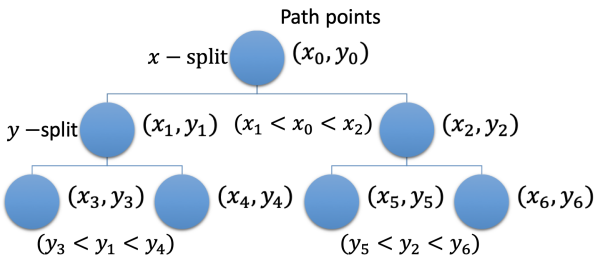

Fig. 21. Example of path points stored in a k-d tree.

To evaluate and demonstrate the effect of obstacles ST boundaries computation, the computation time of obstacles ST boundaries with k-d tree was investigated in the tests. Specially, the planned path in the tests includes 10^4 numbers of points, and the prediction time of two obstacles last 6s with 0.1s resolution. For comparison, the same tests configuration was also evaluated using the traditional "for" loops. The test results showed that using k-d tree data structure, the total computational time of obstacles ST boundaries was only 5.8ms, whereas the computational time using traditional "for" loops was dramatically increased to 62.2ms. Therefore, the test results demonstrate the superior time-saving performance of k-d tree assisted with obstacles ST boundaries computation, particularly for large number of path points.

Step 2 (Sampling On ST Graph): Considering the current

ego car's states as the initial condition, a serials of time instant t_k is sampled from current initial time instant in Frenet frame, such that,

$$t_k \in [0,\ T_p], \quad k \in \mathbb{N} \tag{24}$$

where T_p is defined as the total planning time for speed planning, and $k = 1, 2, \cdots, N_k$. Correspondingly, we sample accelerations $a_{t_k,i}$ at each time instant t_k,

$$a_{t_k,i} \in [d_{max},\ a_{max}], \quad k, i \in \mathbb{N} \tag{25}$$

where d_{max} denotes the maximum deceleration, a_{max} denotes the maximum acceleration, and $i = 1, 2, \cdots, N_i$. Notice that N_k and N_i are defined by sampling resolution. Thus, $(t_k, a_{t_k,i})$ constructs a time-acceleration node based graph for the following dynamic programming search. As shown in Fig. 22 (c), in a example of ROVER 5.0's operation, the total speed planning time T_p is 4s, the time resolution is 0.2s, and the sampled accelerations are defined by $[-2,\ -1,\ 0,\ 0.3,\ 0.6] m/s^2$.

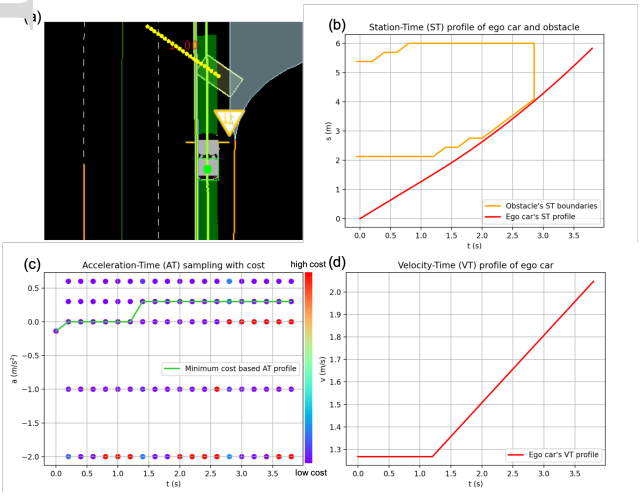

Fig. 22. An example of optimal speed generation with dynamic obstacles' ST boundaries demonstration.

Step 3 (Construct Cost Function): After constructing a graph including sampled $(t_k, a_{t_k,i})$ as nodes, each group of $(a_{t_0,i},\ a_{t_1,i},\ \cdots,\ a_{t_{N_k},i})$ derives speed profile $(s(t_k),\ s^{(1)}(t_k),\ s^{(2)}(t_k),\ s^{(3)}(t_k))$ for $k = 1, 2, \cdots, N_k$, which is evaluated by the summation of cost function as following,

$$\begin{aligned} J(s,t) \triangleq \sum_{i=0}^{N_k} & w_1(s^{(1)}(t_k) - V_{target})^2 + w_2(s^{(2)}(t_k))^2 \\ & + w_3(s^{(3)}(t_k))^2 + J_{obs}(s(t_k)) \end{aligned} \tag{26}$$

where,

$$J_{obs}(s(t_k)) = \begin{cases} \infty, & \text{for } s(t_k) \in s_{blk}(\cdot), \\ 0, & \text{otherwise}, \end{cases} \tag{27}$$

In Eq. (26), $s^{(1)}(t_k) - V_{target}$ denotes the target velocity following cost, $s^{(2)}(t_k)$ and $s^{(3)}(t_k)$ are the speed smoothness cost, and $J_{obs}(s(t_k))$ denotes the total obstacles cost, which has a extremely large cost value when the accumulated station $s(t_k)$ is within the block interval set $s_{blk}(\cdot)$. Thus, by using dynamic programming search, the minimal cost can be obtained easily.

Step 4 (Decision and Rough Speed Profile): Based on the above three steps, an optimal speed profile $(s(t_k), s^{(1)}(t_k), s^{(2)}(t_k), s^{(3)}(t_k))$ with the lowest cost can be obtained for each planning cycle. Furthermore, since ego car's speed profile and obstacles boundaries on $s(t)$ frame are known, the speed decisions, such as stop, yield, overtake, are straightforward. Fig. 22 (a) showed a scene that an ego car cruised on road, where a dynamic obstacle came across the lane in front of ego car. Fig. 22 (c) illustrated an acceleration-time (AT) sampling as ego car's speed planning candidates. The computation of each sampling point's cost is based on Eqs. (26) and (27). Thus, the final speed profile can be generated with the minimal cost through a DP search. Similar as the description in Sec. VII-A.1, in addition to consider the node cost at DP search, the steering and its rate limitations also needs to be considered. In practical DP search, the nodes connection in adjacent level is restricted by the steering limitations. Similar with color representations in Fig. 17, red end of the color spectrum denotes the higher cost, and violet end denotes the lower cost. The computation of cost is a trade-off among obstacle collisions, target speed following, speed smoothness, and steering limitations. Based on minimum cost criterion, an optimal acceleration lists relative with monotonic time instants (AT profile) are determined, as shown in green line. Then, Fig. 22 (b) and (d) demonstrated the final station-time (ST) and velocity-time (VT) profiles in a planning cycle. Especially in Fig. 22 (b), orange line showed the dynamic obstacle's ST boundaries and red line showed the ego car's ST profile. By comparison, we can see ego car's station is always behind of the dynamic obstacle at each time instant, so that, a yield decision can be made directly.

B. Trajectory Optimization

As we have already mentioned, the path-speed profiles and their corresponding decisions such as nudge, stop, and yield produced by previous trajectory based decision step are often sub-optimal and necessary for improvement. At the previous trajectory decision step, we find the path and speed only drivable and doesn't consider motion requirements. Therefore, we post-process the solution of the trajectory based decision by the following optimization procedure (Fan et al., 2018). The procedure consists of path and speed optimization by a novel fourth-order discretized quadratic programing (QP). This optimization essentially refine the trajectory to improve the smoothness.

1) Path Optimization: In this article, our optimization approach is conceptually similar to the motion planner by Baidu Apollo (Fan et al., 2018; *Baidu Apollo team (2017), Apollo: Open Source Autonomous Driving*, n.d.). The path optimization produced by the fourth-order discretized QP is a refinement of previous coarse path by trajectory based decision step. Compared with QP approach, the discretized

QP approach can generate a much smoother path which are minimizing an objective function with vehicle initial conditions and various linearized constraints. The schematic diagram is shown in Fig. 23.

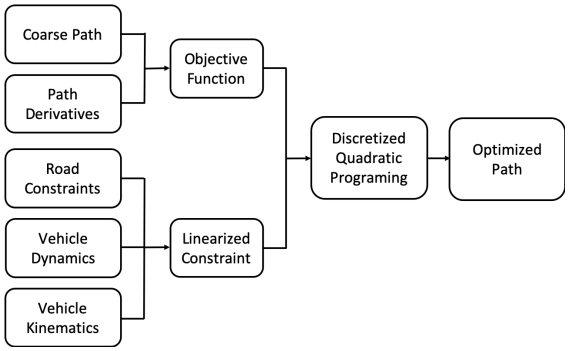

Fig. 23. Discretized quadratic programing path optimization.

With no loss of generality, in Frenet frame, we consider an objective function which can be designed as a linear combination of smoothness cost and guidance line tracking error cost. By design, it can minimize and compromise the lateral deviation from guidance line, lateral speed, lateral acceleration, and lateral jerk. Thus, mathematically, the objective function can be formulated as an optimization problem,

$$\begin{aligned}&\min_{\mathcal{L}_i\in\Re^5} J(\mathcal{L}_i),\ \text{where}\\ &J(\mathcal{L}_i)\triangleq\sum_{i=0}^{N} w_0(l_{opt,i}-l_{d,i})^2+w_1(l_{opt,i}^{(1)})^2+w_2(l_{opt,i}^{(2)})^2\\ &\qquad+w_3(l_{opt,i}^{(3)})^2+w_4(l_{opt,i}^{(4)})^2\end{aligned} \tag{28}$$

where,

$$\mathcal{L}_i=\begin{bmatrix}l_{opt,i} & l_{opt,i}^{(1)} & l_{opt,i}^{(2)} & l_{opt,i}^{(3)} & l_{opt,i}^{(4)}\end{bmatrix}^T \tag{29}$$

Here, $\mathcal{L}_i$ denotes optimized path lateral information including lateral displacement, lateral speed, lateral acceleration, lateral jerk, and the derivative of lateral jerk. In Eq. (28), $l_{opt,i}$ is the optimized path lateral displacement, and $l_{d,i}$ is the guidance line produced by trajectory based decision step. $(l_{opt,i}-l_{d,i})$ denotes the lateral deviation between the optimized path and pre-produced guidance line, and $(l_{opt,i}^{(1)},\ l_{opt,i}^{(2)},\ l_{opt,i}^{(3)},\ l_{opt,i}^{(4)})$ denotes the first derivative to the fourth derivative of lateral displacement which are introduced in order to get smooth function, and w_i for $i=0,\ 1,\ 2,\ 3,\ 4$ are the weightings.

We derive and expand the optimization problem in Eq. (28), and prove the objective function $J(\mathcal{L}_i)$ has a quadratic form with respect to parameter $\mathcal{L}_i$ in (29). Firstly, using (32), we have

$$\begin{aligned}J(\mathcal{L})&=\sum_{i=0}^{N}\left(\sum_{j=0}^{4} w_j(l_{opt,i}^{(j)})^2\right)-2w_0(l_{opt,i})(l_{d,i})+w_0(l_{d,i})^2\\ &=\sum_{i=0}^{N}(\mathcal{L}_i^T W_i\mathcal{L}_i-2\mathcal{L}_i^T c_1+c_2)\\ &=\mathbb{L}^T\mathbb{W}\mathbb{L}-2\mathbb{L}\mathcal{C}_1+\mathcal{C}_2\end{aligned} \tag{30}$$

where,

$$\begin{aligned}\mathbb{L}&=[\mathcal{L}_0\quad\mathcal{L}_1\quad\cdots\quad\mathcal{L}_N]^T\\ \mathbb{W}&=\text{diag}\,([W_0\quad W_1\quad\cdots\quad W_N])\\ W_i&=\text{diag}\,([w_0\quad w_1\quad w_2\quad w_3\quad w_4])\end{aligned} \tag{31}$$

and, $\mathcal{C}_1$ and $\mathcal{C}_2$ are constants. Notice here that diagonal matrix $\mathbb{W}$ is positive definite. Thus, the objective function $J(\mathcal{L})$ has a quadratic convex form with respect to parameter vector $\mathcal{L}$ from Eq. (30).

The constraints in path optimization problem consist of extrinsic constraints and intrinsic constraints. The extrinsic constraints are vehicle initial conditions, road boundary, and dynamic feasibility. In Frenet frame, these constraints are applied on $\mathcal{L}_i$ in Eq. (29) for $i=0,\ 1,\ \cdots N$. The optimized path needs to match vehicle's initial lateral displacement and derivatives such as $\mathcal{L}_0=$ init. In order to apply boundary constraints, road boundary is extracted from map information. The boundary constraints at each sampling point can be described as $l_{low,i}<l_{opt,i}<l_{high,i}$ for $i=1,\ 2,\ \cdots N$. Additionally, the dynamic feasibility constraints related with curvature are applied to $(l_{opt,i}^{(1)}\ l_{opt,i}^{(2)}\ l_{opt,i}^{(3)}\ l_{opt,i}^{(4)})$ at each sampling point for $i=1,\ 2,\ \cdots N$. On the other hand, the intrinsic constraints are based on the discretized path derivative definition with assumption of the derivative of lateral jerk as a constant, namely

$$\mathcal{M}_s\mathcal{L}=\mathcal{C} \tag{32}$$

where,

$$\mathcal{M}_s=\begin{bmatrix}1 & \Delta s & \frac{1}{2}\Delta s^2 & \frac{1}{6}\Delta s^3 & \frac{1}{24}\Delta s^4\\ 0 & 1 & \Delta s & \frac{1}{2}\Delta s^2 & \frac{1}{6}\Delta s^3\\ 0 & 0 & 1 & \Delta s & \frac{1}{2}\Delta s^2\\ 0 & 0 & 0 & 1 & \Delta s\\ 0 & 0 & 0 & 0 & 1\end{bmatrix} \tag{33}$$

$$\mathcal{C}=[0\quad 0\quad 0\quad 0\quad c]^T \tag{34}$$

Here, c denotes a constant, which is the derivative of lateral jerk. Δs is the sampling interval in path's longitudinal direction. After applying all constraints, the optimization problem's solution can be readily obtained by a quadratic programming solver.

To evaluate and demonstrate the effects for path optimization with different weightings in cost function, the path

optimization algorithm was applied to the rough path in a real last-mile vehicle's planning scenario with obstacles. For comparison, the obtained optimal path is shown in Fig. 24 with red line for $[\omega_0\ \omega_1\ \omega_2\ \omega_3] = [100\ 5\ 100\ 5]$, and with green line for $[\omega_0\ \omega_1\ \omega_2\ \omega_3] = [5\ 5\ 100\ 5]$. The test results clearly demonstrate that the proposed path optimization method guaranteed the smoothness across the DP based rough path (blue line). Comparing red path with green pat, it is evident that different weightings of path deviation and path smoothness results in different obtained optimized path. When increasing the weighting of path deviation (ω_0), the optimized path tends to be closed with DP rough path and ignores the path smoothness. Whereas, when decreasing weighting of path deviation (ω_0), the optimized path considers path smoothness more. The optimized path in green line was used in real applications.

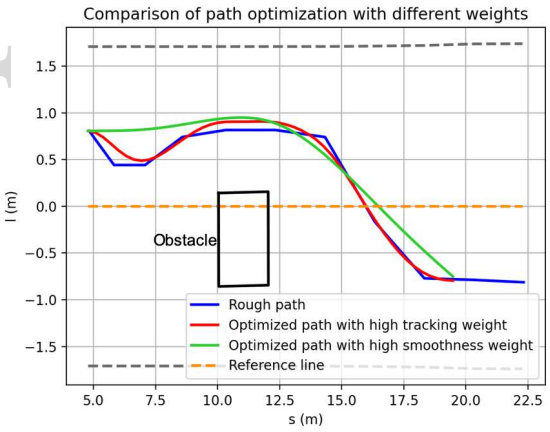

Fig. 24. An example of optimal path generation with different weighting functions.

2) Speed Optimization: Similar as path optimization in Sec. VII-B.1, the speed profile by trajectory based decision step cannot satisfy dynamic feasibility and various constraints, so the fourth-order discretized quadratic programing algorithm is also used here to optimize rough speed profile. The schematic diagram is shown in Fig. 25.

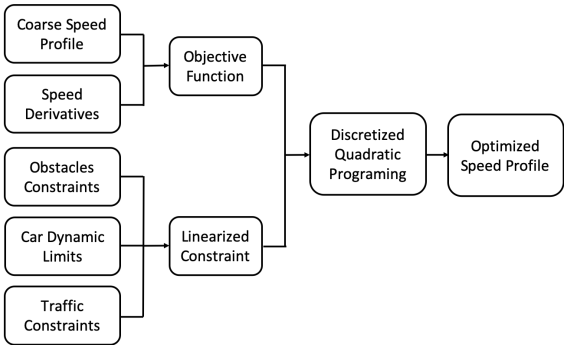

Fig. 25. Discretized quadratic programing speed optimization.

The derivation follows by Sec. VII-B.1 by establishing the relationship that path's longitudinal station information $\mathcal{S}$ at any sampling time t_i. Similarly, an objective function for speed optimization can be designed as follows,

$$\min_{\mathcal{S}_i \in \Re^5} J(\mathcal{S}_i), \text{ where}$$
$$J(\mathcal{S}_i) \triangleq \sum_{i=0}^{N} w_0(s_{opt,i} - s_{d,i})^2 + w_1(s^{(1)}_{opt,i})^2 + w_2(s^{(2)}_{opt,i})^2 + w_3(s^{(3)}_{opt,i})^2 + w_4(s^{(4)}_{opt,i})^2 \tag{35}$$

where,

$$\mathcal{S}_i = \begin{bmatrix} s_{opt,i} & s^{(1)}_{opt,i} & s^{(2)}_{opt,i} & s^{(3)}_{opt,i} & s^{(4)}_{opt,i} \end{bmatrix}^T \tag{36}$$

Here, the vector $\mathcal{S}_i$ includes the optimized longitudinal station, speed, acceleration, jerk, and the derivative of jerk. Similar as Eq. (28).

In Eq. (35), the $s_{d,i}$ is guidance speed profile produced by trajectory based decision step. $(s_{opt,i} - s_{d,i})$ denotes the distance between the optimized path and pre-produced guidance speed profile, and $(s_{opt,i},\ s^{(1)}_{opt,i},\ s^{(2)}_{opt,i},\ s^{(3)}_{opt,i},\ s^{(4)}_{opt,i})$ denote the optimized path longitudinal station and the first derivative to the fourth derivative of longitudinal station which are introduced in order to get smooth function, and w_i for $i = 0,\ 1,\ 2,\ 3,\ 4$ are the weights.

By expanding by Eq. (35), the speed optimization problem can be converted to the matrix form as below,

$$\begin{aligned} J(\mathcal{S}) &= \sum_{i=0}^{N} \left(\sum_{j=0}^{4} w_j (s^{(j)}_{opt,i})^2 \right) - 2w_0(s_{opt,i})(s_{d,i}) + w_0(s_{d,i})^2 \\ &= \sum_{i=0}^{N} (\mathcal{S}_i^T W_i \mathcal{S}_i - 2\mathcal{S}_i^T c_1 + c_2) \\ &= \mathbb{S}^T \mathbb{W} \mathbb{S} - 2\mathbb{S}\mathcal{C}_1 + \mathcal{C}_2 \end{aligned} \tag{37}$$

where,

$$\mathbb{S} = [\mathcal{S}_0 \quad \mathcal{S}_1 \quad \cdots \quad \mathcal{S}_N]^T \tag{38}$$

and the definitions of $\mathbb{W},\ W_i,\ \mathcal{C}_1,\ \mathcal{C}_2$ are the same as before.

By Eq. (37), we have proved that the objective function $J(\mathcal{S})$ has a quadratic convex form with respect to parameter vector $\mathcal{S}$.

In order to solve the optimal speed, the appropriate constraints need to be applied into $\mathcal{S}_{t_i}$ in Eq. (36) in $\{t_i | i = 1,\ 2,\ \cdots,\ N\}$. The same as path optimization problem, the initial condition constraint need to be matched on ego car as $\mathcal{S}_{t_i} = \text{init}$ for $i = 0$. In addition to initial condition constraint, monotonic constraint, longitudinal displacement, speed, acceleration, jerk constraints are depicted as below.

$$s_{opt,t_i} < s_{opt,t_{i+1}}$$
$$\mathcal{S}_{lower} < \mathcal{S}_i < \mathcal{S}_{upper}$$

Similarly, considering the derivative definition and the assumption of the derivative of longitudinal jerk as a constant, we have the intrinsic constraints as,

$$\mathcal{M}_t \mathcal{S} = \mathcal{C} \tag{39}$$

where,

$$\mathcal{M}_t = \begin{bmatrix} 1 & \Delta t & \frac{1}{2}\Delta t^2 & \frac{1}{6}\Delta t^3 & \frac{1}{24}\Delta t^4 \\ 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 & \frac{1}{6}\Delta t^3 \\ 0 & 0 & 1 & \Delta t & \frac{1}{2}\Delta t^2 \\ 0 & 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \tag{40}$$

$$\mathcal{C} = [0 \quad 0 \quad 0 \quad 0 \quad c]^T \tag{41}$$

Here, c denotes a constant, which is the derivative of longitudinal jerk. Δt is the sampling time in trajectory.

Similar as the evaluation of path optimization with different weightings in Sec. VII-B.1, the effect of the different weighting functions on the speed optimization performance is also evident from the test results. The test scenario is a decelerating process when ego car failed avoid an obstacle. As shown in Fig. 26, we compared speed planning performance with a high speed following weighting and a high speed smoothness weighting. In Fig. 26 (a), we can see the planned speed tried to be closed with reference speed for high speed following weighting. In Fig. 26 (b), however, the jerk profile was oscillating under the same high speed following weighting, which means the speed smoothness cannot be maintained throughout the entire speed planning course. Whereas, the jerk profile (green line) became smooth when increasing the number of speed smoothness weighting. Therefore, the optimized speed profile in green line was used in real applications for smoothness.

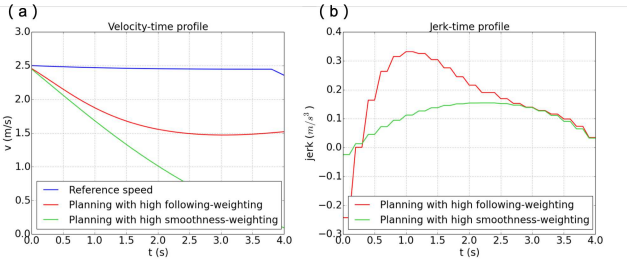

Fig. 26. An example of optimal velocity and jerk generation with different weighting functions: (a) is the optimized velocity profile with different weightings, and (b) is the optimized jerk profile with different weightings, respectively.

VIII. SIMULATION PLATFORM

For validation purpose, the most authoritative way to evaluate the performance of motion planning algorithm in autonomous driving is road tests. However, road tests have some challenges in reality, such as test site limitation, test condition repeatability, time consuming, and high cost. In order to address these limitations, we developed a simulation platform to realistically simulate and visualize typical driving behaviors of autonomous vehicles with both real traffic scene and artificial scene data. By applying simulation platform, the algorithm development has been expedited and planning performance has been improved.

A. Simulation Platform Architecture

The simulation platform is designed to mimic the real implementation environment. The framework diagram is shown in Fig. 27. From the flow chart, we can see the simulation data source is mainly from the real traffic scene of road tests. Additionally, the artificial scene data is significantly complementary if real tests data doesn't cover all traffic scene. When data get ready, the agent server module will re-construct the traffic scene based on input data to generate localization and perception information including vehicle states, dynamic obstacles track, objects detection, traffic lights detection, etc. These generated data, as well as map information are applied to motion planning component to produce corresponding behavior decision and motion planning. Finally, based on the planning results, evaluation module can assess the new algorithm's performance by pre-defined planning criterion.

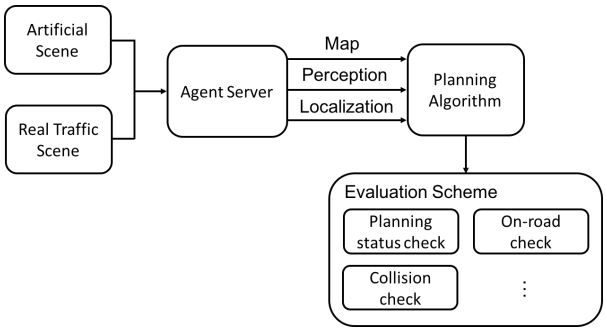

Fig. 27. Flow chart of simulation platform.

Evaluation module is the most important part in the simulation platform, because it provides a good quality control for novel autonomous driving algorithm development before implementing to the real road tests. The common evaluation criterion mainly includes planning&control, perception, predication, localization, etc, as shown in Table. II. Taking planning&control evaluations as example, planning status check evaluates feasibility of speed and acceleration from planned trajectory; Collision check evaluates if the ego car along with planned trajectory collides with perceived obstacles; On-road check evaluates if the planning results go beyond the road boundaries. By using evaluation module, we can evaluate planning performance readily, sequentially, expedite the planning algorithm development.

B. Algorithm Development By Simulation

In order to expedite new planning algorithm development in autonomous driving, the proposed simulation platform take a great importance. The flow chart of simulation assisted algorithm development is as shown in Fig. 28. When the updated planning algorithm induced by new features or existing issues needs to be validated, we can validate the algorithm in the proposed simulation platform before implementing to a large number of vehicles for real road tests. It will

TABLE II
EVALUATION CRITERION SUMMARY

Planning	Perception	Prediction	Localization	Simulation
Planning status evaluation	Perception status evaluation	Prediction completeness evaluation	Localization results consistency evaluation	Simulation completeness evaluation
Destination approaching evaluation	Perception-vision precision evaluation	Prediction precision evaluation	⋮	Time duration and mileage evaluation
Vehicle on-road evaluation	Perception-Bev precision evaluation	⋮		Simulation execution core evaluation
Collision evaluation	Perception-track precision evaluation			⋮
Obstacle avoidance evaluation	Traffic lights status evaluation			
Open space planning status evaluation	Traffic lights duration evaluation			
⋮	⋮			

dramatically improve the efficacy of algorithm development. In Fig. 28, after updating new planning algorithm, we compile the program and upload the corresponding binary into simulation platform. Then, we can select thousands of standard scenario sets and submit simulation tasks. Specially, we constructed a library including tens thousands of real traffic scene and artificial traffic scene in prior. The library is trying to cover real application scenarios as many as possible. Finally, evaluation scheme will validate the proposed algorithm's feasibility and accuracy. If the success rate of evaluation is reduced, then we continue to improve the algorithm. Otherwise, the proposed algorithm is ready for real road tests.

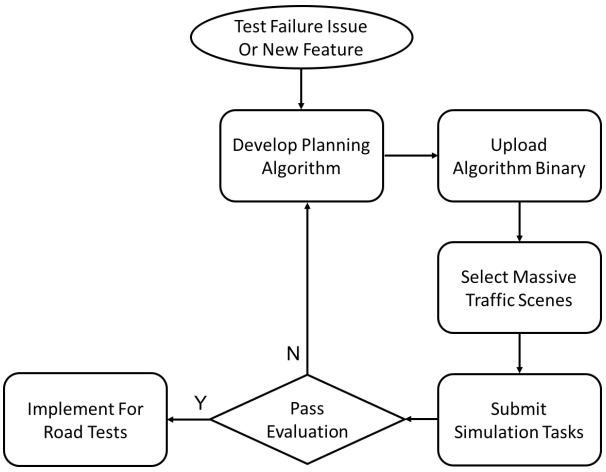

Fig. 28. Flow chart of algorithm development by simulation.

Fig. 29 demonstrated the statistic analysis of simulation case volume for our autonomous driving system at recent one month. The blue bar chart denotes the daily simulation by artificial scene, red bar chart denotes the daily simulation by real traffic scene, and green line is the total number of the simulation case. Based on large amount of simulation cases, we can see our autonomous driving algorithm is validated

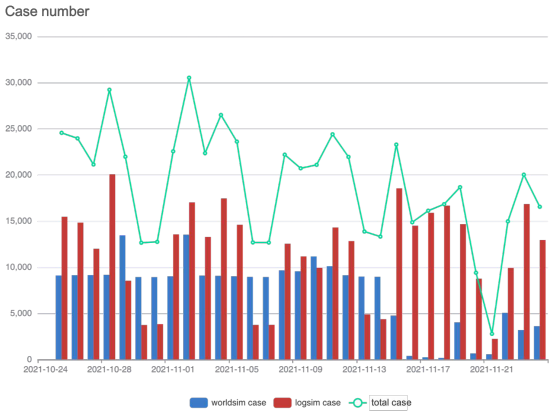

Fig. 29. Simulation case number statistics from recent one month data.

by tens of thousands of simulation cases everyday. Such big simulation data analysis can efficiently evaluate and validate our algorithm and systems, so that, dramatically improve autonomous driving performance. Similarly, the daily simulation results at recent one month for motion planning component are summarized in Fig. 30. The daily motion planning simulation set comprises with over 1,000 real traffic scene cases, which still increases along with more real delivery operations. So, it is the reason that the pass rate fluctuates along with time. However, when pass rate drops heavily at some day, that means new planning algorithm commit induced large number of failures, which real delivery operations cannot afford. For example, we can see pass rate dropped from 90.42% to 83.30 % at Nov. 02, 2021 in Fig. 30, which implies too many evaluations defined in first column of Table. II failed. Therefore, the comprehensive simulation validation can maintain the algorithm's performance and improve the efficiency.

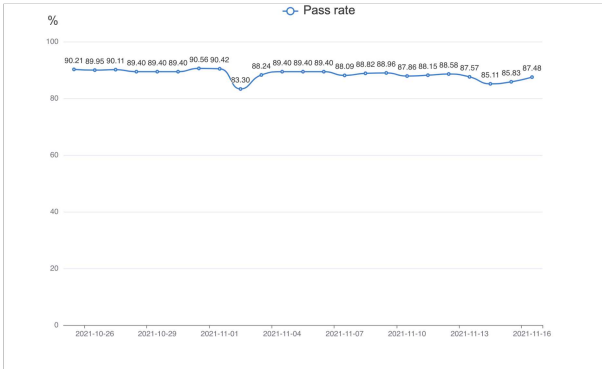

Fig. 30. Motion planning daily simulation's pass rate from recent one month data.

IX. CASE STUDY

This section describes the real express delivery services using JD's autonomous delivery vehicles in China. 192 ROVER vehicles were deployed in 6 different big cities in China, including in Suzhou, Beijing, Wuhan, Xi'an, Xianyang, and Guangzhou. The demonstration of our motion planning algorithm's performance are illustrated in Table III. The statistics received from operational staffs showed that the total mileage, running times, the number of departed delivery orders and the number of arrived orders are 115,475.35 km, 24,774 times, 350,574 orders and 295,205 orders, respectively, for operations from June, 2020 to November, 2021. From the Table. III, the operation results demonstrated the total completion rate of delivery orders is as high as 84.21% (295,205/350,574) by using our autonomous delivery vehicles.

Most delivery operations are focused on the Suzhou city. 117 out of 192 ROVER vehicles were deployed in Suzhou for daily delivery, and the running mileage in Suzhou reached 81,516.20 km, accounting for 70.60% (81,516.20/115,475.65) of the total mileage. There are three main operation districts in Suzhou, which are Suzhou-Likou Business District, Changshu-Southeast Business District and Changshu-Zhenmen Business District. The operating mileages in above three districts are 28,567.50 km, 35,278.80 km and 15,462.20 km, respectively. The completion rate (the number of delivery orders/the number of completed orders) in three districts are 81.86% (58,413/71,358), 83.25% (118,444/142,275) and 80.78% (44,763/55,413), respectively. Moreover, the operation routes and all kinds of operation scenarios are shown in Fig. 31.

Specially, during the outbreak of Covid-19, for the purpose of contact-less goods transportation, JD.com deployed two ROVER autonomous delivery vehicles in Wuhan to deliver medical supplies and living material from JD's local distribution center to Wuhan's ninth hospital and three large communities. The total mileage, running times, the number of departed delivery orders and the number of arrived orders are 198.13 km, 136 times, 2,161 orders and 1,653 orders, respectively. The operating data demonstrated the potential ability of our method in complex urban environments.

(a) task routes in Changshu map (b) cruise mode on the roads

(c) waiting in front of traffic lights (d) left turn scenario in intersections

(e) cross-intersection motion (f) trying to park in destinations

Fig. 31. The operation routes and all kinds of operation scenarios: (a) is a route planning example in a monitoring system; (b) shows a autonomous delivery vehicle drives on cruise on-road scenario; (c) shows a vehicle waiting for traffic lights; (d) is the left turn scenario in an intersection, which is very different with left turn for passenger vehicles; (e) is a cross-intersection scenario with pedestrians and bicycles, which is also different with passenger vehicles; (f) is parking in a destination for delivery.

In addition to the above real autonomous delivery completeness data, the internal program running criteria in each vehicle is also important. For example, the CPU usage percentage and the component's end-to-end process time are two critical running criteria for autonomous driving vehicles. Fig. 32 (a) showed a CPU (8-core CPU in one xavier) average usage percentage for total autonomous system is only around 57%, and our core autonomous algorithms (highlighted 'rover' in Fig. 32) including planning, control, perception, prediction, and localization only take 22% CPU usage averagely. To eliminate the heavy outliers impact, we also provided a CPU 90% percentiles usage for comparison. From Fig. 32 (b), we can see the whole system's CPU 90% percentiles usage is still below 75%, and the CPU 90% percentiles usage for the core autonomous algorithms 'rover' is also as low as 35%. Moreover, the end-to-end process

TABLE III

THE OPERATING DATA FOR DIFFERENT CITIES.

Cities	Delivery station	Mileage (Km)	Running times	Delivery orders	Completed orders
Beijing		17,847.40	5,428	55,488	44,482
	Beijing Mulin Business Department	3,266.65	576	6,193	4,714
	JD Pai (Beijing Wuzi University)	7,714.78	3,382	39,352	32,125
	7FRESH Business Department in Beijing JD Building	6,865.97	1,470	9,943	7643
Guangzhou		4,355.90	835	9,866	7,934
	JD Pai (Research Institute of Guangzhou University)	4,355.90	835	9,866	7,934
Suzhou		81,516.20	15,298	266,771	228,129
	Suzhou Likou Business Department	27,835.40	4,183	71,358	58,413
	Changsu Southeast Business Department	42,337.63	8,729	142,275	118,444
	Changsu Zhenmen Business Department	10,873.34	2,187	55,413	44,763
	JD Pai (Changshu Institute of Technology in East Lake)	469.83	199	8,375	6,509
Xianyang		5,239.54	1,245	10,624	8,208
	JD Pai (Shanxi International Business College)	5,239.54	1,245	15,624	11,208
Xi'an		6,318.18	1,832	5,964	4,799
	Xi'an Feitian Business department	6,318.18	1,832	5,964	4,799
Wuhan		198.13	136	2161	1653
	JD Pai (College of Arts and Sciences, Wuhan University)	198.13	136	2161	1653
Total		115,475.35	24,774	350,574	295,205

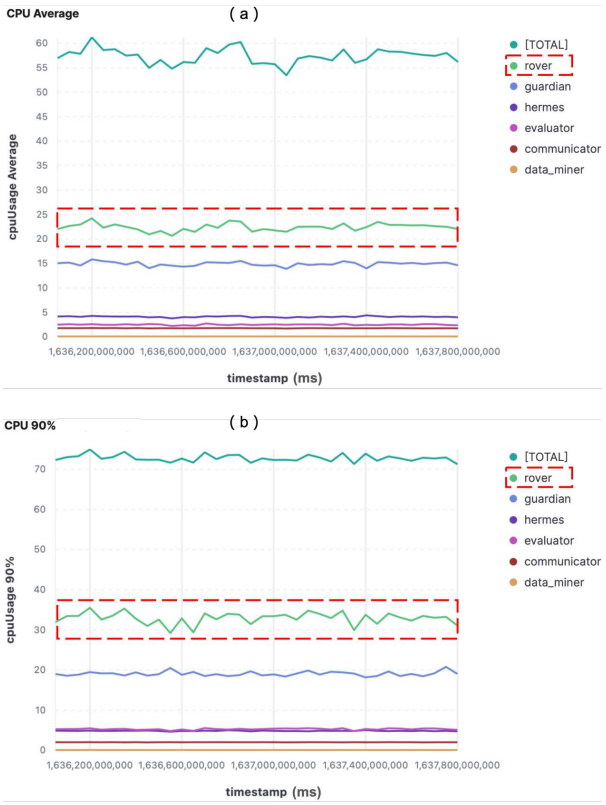

Fig. 32. CPU usage percentage comparison for different modules: (a) is the CPU average usage percentage in autonomous vehicles; (b) is the CPU 90% percentiles usage percentage in autonomous vehicles.

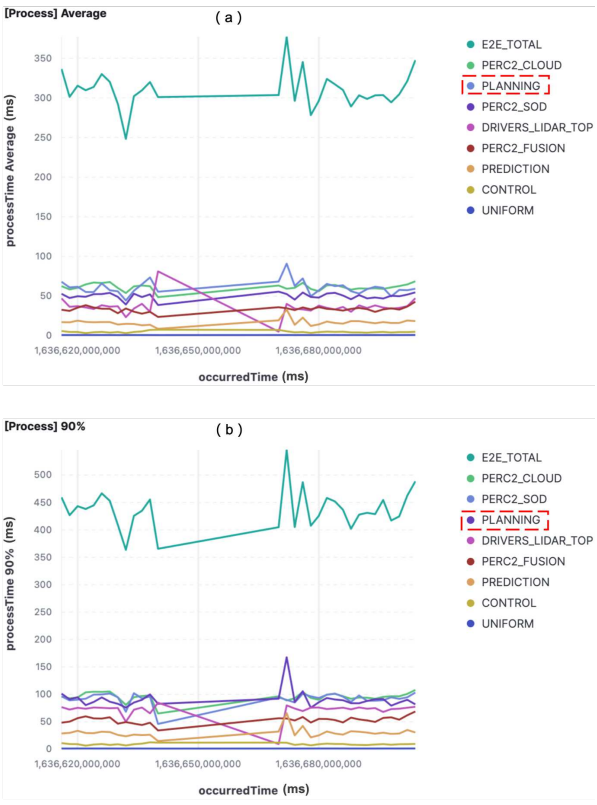

Fig. 33. End-to-end process time comparison for each cycle: (a) is the end-to-end average process time in autonomous vehicles; (b) is the end-to-end 90% percentiles process time in autonomous vehicles.

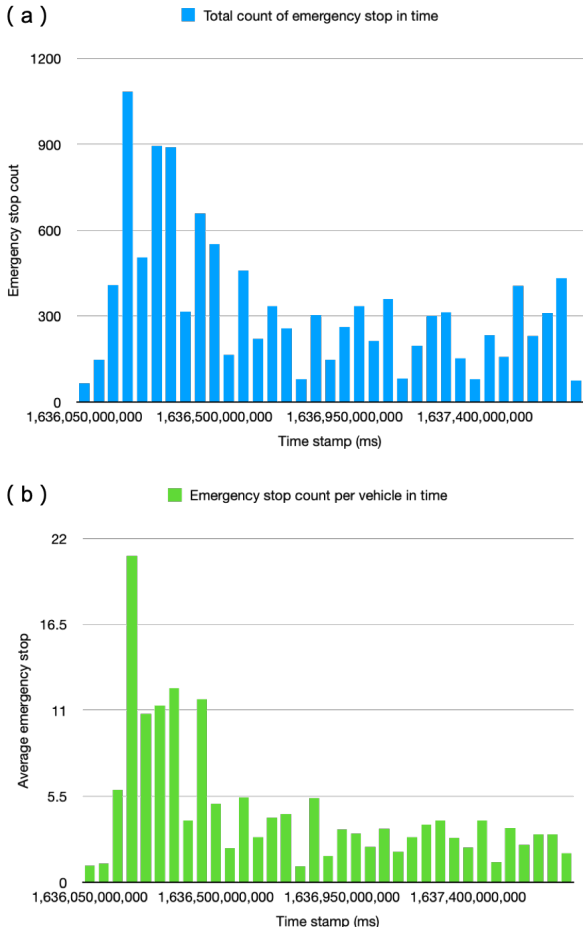

Fig. 34. Emergency stop statistic data based on autonomous delivery operations during recent one month: (a) is the total count of emergency stop happening in the all operating delivery vehicles; (b) is the average emergency count per vehicle during each day period.

time in each cycle is the other important health criteria for autonomous driving systems, where, 'end-to-end' means a whole cycle from lidar sensor component to control component. As shown in Fig. 33 (a) and (b), the planning component's average process time and 90% percentiles process time are around 60ms and 100ms, respectively. The end-to-end average process time and 90% percentiles process time are around 300ms and 450ms. Therefore, Fig. 33 illustrated that no matter proposed motion planning architecture or the whole autonomous system, a relatively small process time can be maintained though delivery vehicles operation.

In the meanwhile, to quantitatively evaluate the overall autonomous driving performance and demonstrate the algorithm's efficacy, our system counted the total emergency stop number in all of the operating delivery vehicles during recent one month period, as shown in Fig. 34 (a). Moreover, by dividing by the number of delivery vehicles operating at that day, the average emergency stop count can be easily obtained by Fig. 34 (b). Note: this emergency stop isn't generated by motion planning algorithm, but generated by infrastructure system, so, we can consider the emergency stop number as motion planning performance in real operations. By Fig. 34, We can see a very low emergency stop number based on our autonomous driving algorithms can be maintained during daily operations at very complex urban environments (mostly below 10 times during 8 hours operation per day). Therefore, this emergency stop evaluation demonstrated our motion planning algorithm's efficacy.

X. CONCLUSIONS

In this article, a motion planning framework was proposed and implemented to achieve intelligent obstacle and traffic based decision and trajectory smoothing in autonomous driving applications. A combined parallel scenarios and hierarchical tasks framework was provided to improve the efficacy of motion planner. The proposed motion planning framework was classified with route planning, scenario planning, and trajectory planning. For route planning, both single and multiple destinations based routing algorithm was described. Then, the final smooth routing results was obtained by a nonlinear optimization method. For scenario planning, we developed various scenario based planning strategy, according with routing and environment changes. Additionally, some new scenarios are specifically designed for last-mile delivery applications. Finally for trajectory planning, it contains with trajectory-based decision and trajectory optimization. Trajectory-based decision generated a rough path-speed profile based on the obstacles and traffic regulations. Then, the rough path-speed profile was optimized to generate a smoother trajectory which meets different constraints.

In this paper, as mentioned, we focused on the motion planning for autonomous driving in practical applications. In order to validate motion planning algorithms readily, a simulation platform was introduced for reliability and scalability of algorithms development. The strategy of the proposed motion planning has been effectively implemented and validated in hundreds of ROVER 5.0 delivery vehicles in China. In the road tests and large deployments, the algorithm has been evaluated and validated in more than 100,000 hours and 100,000 kilometers under various complex urban scenarios. The tests and deployments results demonstrated that using the proposed motion planning strategy, the planning effectiveness and performance under can be substantially improved, even in the presence of crowded urban road conditions. Finally, the two critical program running criteria in autonomous driving applications was discussed.

ACKNOWLEDGMENT

We would like to thank planning & control team members of autonomous driving division in JD.com for their help with implementing and testing the motion planner. We would also like to thank perception, localization, system architecture, simulation, and operation teams for their efforts related to this work. Especially, we would like to thank Xiaoyong Ma for providing daily operation data and Tao Yin for demonstration photos during operations. Finally, We also gratefully acknowledge the support of JD.com American Technologies Corporation.

REFERENCES

Applegate, D. L., Bixby, R. E., Chvatal, V., & Cook, W. J. (2006). *The traveling salesman problem: a computational study*. Princeton university press.

Bacha, A., Bauman, C., Faruque, R., Fleming, M., Terwelp, C., Reinholtz, C., ... others (2008). Odin: Team victortango's entry in the darpa urban challenge. *Journal of field Robotics*, *25*(8), 467–492.

Baidu apollo team (2017), apollo: Open source autonomous driving. (n.d.). `https://github.com/ApolloAuto/apollo`. GitHub.

Berglund, T., Brodnik, A., Jonsson, H., Staffanson, M., & Soderkvist, I. (2009). Planning smooth and obstacle-avoiding b-spline paths for autonomous mining vehicles. *IEEE Transactions on Automation Science and Engineering*, *7*(1), 167–172.

Brezak, M., & Petrović, I. (2013). Real-time approximation of clothoids with bounded error for path planning applications. *IEEE Transactions on Robotics*, *30*(2), 507–515.

Buehler, M., Iagnemma, K., & Singh, S. (2007). *The 2005 darpa grand challenge: the great robot race* (Vol. 36). Springer.

Buehler, M., Iagnemma, K., & Singh, S. (2009). *The darpa urban challenge: autonomous vehicles in city traffic* (Vol. 56). springer.

Dolgov, D., Thrun, S., Montemerlo, M., & Diebel, J. (2008). Practical search techniques in path planning for autonomous driving. *Ann Arbor*, *1001*(48105), 18–80.

Fan, H., Zhu, F., Liu, C., Zhang, L., Zhuang, L., Li, D., ... Kong, Q. (2018). Baidu apollo em motion planner. *arXiv preprint arXiv:1807.08048*.

Ferguson, D., Howard, T. M., & Likhachev, M. (2008). Motion planning in urban environments. *Journal of Field Robotics*, *25*(11-12), 939–960.

Ferguson, D., & Stentz, A. (2007). Field d*: An interpolation-based path planner and replanner. In *Robotics research* (pp. 239–253). Springer.

Glaser, S., Vanholme, B., Mammar, S., Gruyer, D., & Nouveliere, L. (2010). Maneuver-based trajectory planning for highly autonomous vehicles on real road with traffic and driver interaction. *IEEE Transactions on intelligent transportation systems*, *11*(3), 589–606.

González, D., Pérez, J., Milanés, V., & Nashashibi, F. (2015). A review of motion planning techniques for automated vehicles. *IEEE Transactions on Intelligent Transportation Systems*, *17*(4), 1135–1145.

Gu, T., Atwood, J., Dong, C., Dolan, J. M., & Lee, J.-W. (2015). Tunable and stable real-time trajectory planning for urban autonomous driving. In *2015 ieee/rsj international conference on intelligent robots and systems (iros)* (pp. 250–256).

Howard, T. M., & Kelly, A. (2007). Optimal rough terrain trajectory generation for wheeled mobile robots. *The International Journal of Robotics Research*, *26*(2), 141–166.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The international journal of robotics research*, *30*(7), 846–894.

Kümmerle, R., Ruhnke, M., Steder, B., Stachniss, C., & Burgard, W. (2015). Autonomous robot navigation in highly populated pedestrian zones. *Journal of Field Robotics*, *32*(4), 565–589.

Li, B., Liu, S., Tang, J., Gaudiot, J.-L., Zhang, L., & Kong, Q. (2020). Autonomous last-mile delivery vehicles in complex traffic environments. *Computer*, *53*(11), 26–35.

McNaughton, M., Baker, C. R., Galatali, T., Salesky, B., Urmson, C., & Ziglar, J. (2008). Software infrastructure for an autonomous ground vehicle. *Journal of Aerospace Computing, Information, and Communication*, *5*(12), 491–505.

McNaughton, M., Urmson, C., Dolan, J. M., & Lee, J.-W. (2011). Motion planning for autonomous driving with a conformal spatiotemporal lattice. In *2011 ieee international conference on robotics and automation* (pp. 4889–4895).

Paden, B., Čáp, M., Yong, S. Z., Yershov, D., & Frazzoli, E. (2016). A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Transactions on intelligent vehicles*, *1*(1), 33–55.

Rajamani, R. (2011). *Vehicle dynamics and control*. Springer Science & Business Media.

Thrun, S., Montemerlo, M., Dahlkamp, H., Stavens, D., Aron, A., Diebel, J., ... others (2006). Stanley: The robot that won the darpa grand challenge. *Journal of field Robotics*, *23*(9), 661–692.

Urmson, C., Anhalt, J., Bagnell, D., Baker, C., Bittner, R., Clark, M., ... others (2008). Autonomous driving in urban environments: Boss and the urban challenge. *Journal of Field Robotics*, *25*(8), 425–466.

Werling, M., Ziegler, J., Kammel, S., & Thrun, S. (2010). Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 ieee international conference on robotics and automation* (pp. 987–993).

Zeng, W., & Church, R. L. (2009). Finding shortest paths on real road networks: the case for $a^{\star}$. *International journal of geographical information science*, *23*(4), 531–543.

Ziegler, J., Werling, M., & Schroder, J. (2008). Navigating car-like robots in unstructured environments using an obstacle sensitive cost function. In *2008 ieee intelligent vehicles symposium* (pp. 787–791).