Design, Field Evaluation, and Traffic Analysis of a Competitive Autonomous Driving Model in a Congested Environment

Daegyu Lee℠, Hyunki Seong℠, Graduate Student Member, IEEE, Gyuree Kang℠, Seungil Han℠, D. Hyunchul Shim℠, Member, IEEE, and Yoonjin Yoon℠

Abstract—Recently, numerous studies have investigated cooperative traffic systems using the communication among vehicle-to-everything (V2X). Unfortunately, when multiple autonomous vehicles are deployed while exposed to communication failure, there might be a conflict of ideal conditions between various autonomous vehicles leading to adversarial situations on the roads. In South Korea, virtual and real-world urban autonomous multi-vehicle races were held in March and November of 2021, respectively. During the competition, multiple vehicles were involved simultaneously, which required maneuvers such as overtaking low-speed vehicles, negotiating intersections, and obeying traffic laws. In this study, we introduce a fully autonomous driving software stack to deploy a competitive driving model, which enabled us to win the urban autonomous multi-vehicle races. We evaluate module-based systems such as navigation, perception, and planning in real and virtual environments. Additionally, an analysis of traffic is performed after collecting multiple vehicle position data over communication to gain additional insight into a multi-agent autonomous driving scenario. Finally, we propose a method for analyzing traffic in order to compare the spatial distribution of multiple autonomous vehicles. We study the similarity distribution between each team’s driving log data to determine the impact of competitive autonomous driving on the traffic environment. Our fully autonomous software architecture, proven successful in winning urban autonomous multi-vehicle races in South Korea, is ready for deployment on urban robot taxis. Our traffic analysis addresses multi-agent scenarios and resolves competitive conflicts among robot taxi companies, crucial for smart city integration and optimizing autonomous vehicle performance in complex urban settings.

Index Terms—Autonomous vehicles, traffic information.

Manuscript received 6 November 2022; revised 24 January 2024; accepted 25 March 2024. Date of publication 10 April 2024; date of current version 1 August 2024. This work was supported in part by the Institute of Information and Communications Technology Planning and Evaluation (IITP) through a grant funded by the Korean Government [Ministry of Science and Information and Communication Technology (MSIT)] under Grant 2021-0-00029 and in part by Hyundai Autonomous Competition, organized by the Hyundai Motor Group, to provide assistance to the participating teams in the 2021 competition. The Associate Editor for this article was H. Wang. (Daegyu Lee and Hyunki Seong contributed equally to this work.) (Corresponding authors: D. Hyunchul Shim; Yoonjin Yoon.)

Daegyu Lee, Hyunki Seong, Gyuree Kang, and D. Hyunchul Shim are with the School of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 34141, Republic of Korea (e-mail: lee.dk@kaist.ac.kr; bynks@kaist.ac.kr; fingb20@kaist.ac.kr; hcshim@kaist.ac.kr).

Seungil Han is with the Autonomous Driving Strategic Technology Development Team, Hyundai Motor Company, Seoul 06797, Republic of Korea (e-mail: si.han@hyundai.com).

Yoonjin Yoon is with the Civil and Environmental Engineering, KAIST, Daejeon 34141, Republic of Korea (e-mail: yoonjin@kaist.ac.kr).

Digital Object Identifier 10.1109/TITS.2024.3382680

I. INTRODUCTION

R ECENT surge in vehicle-to-everything (V2X) based cooperative driving has demonstrated considerable improvements in effective traffic transition of autonomous vehicles, such as yielding and anticipatory slow-down [1]. When it comes to an urban environment, the requirements for effective autonomous driving becomes more complex. Due to the wide range of road conditions, including physical road geometry and communication reliability, more capabilities are necessary to maintain compliance with traffic law. Especially, when vehicle drive in a congested urban environment, the capability to navigate in a competitive manner through the congested traffic plays the critical role. However, when multiple autonomous vehicles are deployed, this competitive driving approach may lead to adversarial situations on the roads. In this case, public exposure and acceptance is another critical factor [2]. The negative impact of exposed communication failures or aggressive maneuvers can take months, if not years, to reverse.

In this study, we propose a full stack of autonomous driving architecture that can effectively deal with various uncertainties in the complex urban environments. The architecture includes modular-level development of perception, localization, planning, and control. It requires only the traffic signal information through V2X, and provide efficient localization in the GPS-denied environment. A real-world experiment was carried out in the Hyundai Motor Group’s biennial urban autonomous racing competition [3], [4], [5], [6]. The competition is a multi-agent autonomous vehicle race on the congested streets of Seoul, with the goal of minimizing the transversal time while fully complying with traffic laws. The vehicle (Fig. 1) equipped with our architecture won the competition only with a simple feed on traffic signal information through vehicle-to-infrastructures (V2I) and no GPS.

The trajectory analysis revealed that our vehicle trajectory shares much lesser degrees of similarities among the five competitors in curve, intersection and straight maneuvers. Such observations not only validates the effectiveness of the proposed architecture, but also indicate that it can provide increased level of resilience in autonomous driving in urban environment. Furthermore, we can analyze how competitive driving approaches can result in adversarial situations on roads in multiple autonomous vehicle deployment scenarios.
Fig. 1. The autonomous vehicle developed by the KAIST team competed in the Seoul competition held on November 29, 2021.

The remainder of this paper is organized as follows: Section II introduces previous related work. Section III describes the development of robust localization, perception, motion planning, and behavior planning. Section V describes the traffic-level analysis of competitive driving models. In Section VI, the experimental results of both a simulation and real-world application are presented. Finally, Section VII concludes this paper.

II. RELATED WORKS

A. Full-Stack Autonomy for Self-Driving

In recent years, several studies on autonomous driving have been conducted. Module-based approaches [6], [7], [8] have designed autonomous driving frameworks that consist of multiple modules, ranging from perception and decision-making to path planning and control. Learning-based approaches, on the other hand, have utilized neural networks to deploy autonomous systems through supervised learning [9], [10], adaptive learning [12], [13], or reinforcement learning [14], [15]. Our study focuses on developing a modular autonomy stack that incorporates both learning-based and non-learning-based algorithms, designed to adeptly handle a variety of driving situations in urban scenarios with resilience.

Generally, a full-stack of autonomous technology is composed of primary modular layers: sensors, computing devices and their software interfaces, perception, planning, and control [7], [16]. Owing to varying user requirements for autonomous stacks, integration of full-stack autonomy with public projects is essential for self-driving studies [17], [18]. Numerous studies for public use [8], [19] have evolved into open-source software projects for autonomous driving. Thanks to these efforts, essential functions for autonomous driving are now accessible to the public, allowing many individuals and organizations to utilize these contributed open sources [20]. In this study, we propose a full-stack autonomy system that covers localization, perception, and planning modules. Several works [6], [21] have demonstrated the effectiveness of full-stack systems in test environments for connected and autonomous vehicles, tackling a range of driving scenarios such as stopping for unexpected pedestrians, navigating construction zones, or managing intersections. However, in contrast to these scenario-based test environments, our full-stack autonomy system was deployed in a more demanding context: a complex urban setting with multiple autonomous vehicles operating under their own autonomous systems. Our system successfully handled multi-task scenarios with non-cooperative vehicles, including navigating intersections with traffic signals, changing lanes, and overtaking vehicles.

B. Autonomous Racing Competitions

Competitions provide excellent motivation to quickly respond to and accelerate demands in robotics technology. In an effort to create the first fully autonomous ground vehicles, the Defense Advanced Research Projects Agency (DARPA) Grand Challenge was held to encourage the development of various robotics technologies [22], [23], [24], [25]. The F1Tenth is a competition to develop autonomous driving algorithms in a high-speed environment based on a race car that is one-tenth the size of an actual car [26], [27]. This scale model has an advantage in that it provides a handy platform for various experimental attempts related to autonomous driving research. Recently, an autonomous driving algorithm development contest in a high-speed environment using full-scale racing cars was held at Indianapolis Motor Speedway, and the same competition was held in Las Vegas shortly thereafter [28], [29], [30], [31]. Additionally, as part of our research project, our team also participated in a high-speed race.

III. SYSTEM DESIGN

A. Software

Figure 2 shows an overview of our autonomous driving system, composed of perception, planning, and control modules. An advantage of a modular autonomous pipeline is that we can contribute to the development of individual
modulars with localization, planning, and perception algorithms. In addition, we manage the behavior of the autonomous vehicle based on the modular system, rather than an end-to-end autonomous pipeline. As a result, our vehicle estimates its position in the world with sensory data and executes planning and control algorithms sequentially. For perception, we developed a conservative multi-modal perception system utilizing an RGB camera, LiDAR point cloud, and semantic road map information. Moreover, the planning algorithm is designed hierarchically and consists of route planning, motion planning, and behavior planning. First, the route-planning algorithm determines the global path to arrive at the target position. Subsequently, the vehicle works to avoid obstacles encountered during the drive using a motion-primitives-based motion-planning algorithm. In low-speed traffic scenarios, competitive driving behavior planning allows the vehicle to overtake or change lanes to escape traffic congestion. At the end of the autonomous stack, we control the electric vehicle directly over the controller area network (CAN) gateway.

B. Hardware
An Ouster OS2-128 LiDAR, two Velodyne Puck 16-channel LiDAR, two Sekonix SF3325-100 RGB cameras, and MicroStrain 3DM-GX5-1MU were used for the autonomous driving, with an industrial-grade computing platform (CPU: Intel Xeon, 8th-Gen Core processor, two of GPU: RTX 2020 Ti, 11 GB), an Intel NUC (Intel i7-10710U Processor, 12 M cache, up to 4.70 GHz) and an Nvidia Drive PX 2 (AutoChauffeur) [CPU: 4x Denver, 8x Cortex A57, GPU: 2x Parker GPGPU (2 x 2 SM Pascal, 512 CUDA cores) + 2x dedicated MXM modules] as the computing devices. Using these three computing devices, our full-autonomous stack can obtain solutions for the localization, planning, and perception algorithms. Most of the algorithms were implemented in C++ and executed using the robot operating system (ROS) [32] in Ubuntu 18.04 Linux. In addition, we employed a deep-learning-based algorithm with the Python script, and integrated inference results on the ROS system.

IV. METHODS
A. Resilient Localization
The localization system is one of the most essential modules in the autonomous stack for real-world field testing. Unless a high-quality differential global positioning system (DGPS) sensor is used, global navigation satellite system (GNSS)-based localization quality is greatly deteriorated when passing near a building, especially in metropolitan areas due to a GPS multi-path problem or weak signal strength [33]. However, some researchers have used a Kalman filter with a GPS [34], [35], [36], [37], [38], [39], [40], and several studies have been conducted to enhance the localization system based on sensor fusion, while reducing the weight of the GPS [41], [42]. Therefore, LiDAR or vision-sensor-based localization methods would be more ideal to estimate vehicle position.

In this study, we propose a novel localization system, without a GPS, utilizing only a pre-built 3-D point-cloud map, and RGB camera-based online lane detection. As a result of our previous work deploying robots in GPS-denied areas [43], we have built a localization system that enables location without a GNSS. In a specific district-scale urban area, pre-built 3-D map-based localization [44], [45], [46], [47], [48] can be utilized after constructing a 3-D point-cloud map from several milestone works [49], [50], [51], [52], [53]. One of the advantages of utilizing a pre-built map is that we can discover any map errors before deploying autonomous vehicles in the real world.

1) Registration: We define $W \subset \mathbb{R}^2$ as the map coordinates of the autonomous vehicle. Let $\hat{x}_W = \{x_1, y_1, \theta_1\} \in W$ be the estimated vehicle position $(x, y)$ with heading $\theta$ defined in the pre-built 3-D point-cloud map at time $t$. In addition, we define $B \subset \mathbb{R}^2$ as the vehicle body coordinates. Here, the annotated values $B$ indicate information obtained from the origin of the vehicle body’s coordinates—i.e., the center point of the rear axle. We also define the voxel-filtered LiDAR points $z_i^B = \{z_1^B, ..., z_k^B\}$ at time $t$, where $z_i^B$ is one of the voxelized points from incoming LiDAR points, and the voxelized point-cloud map $M_W = \{m_1^W, ..., m_n^W\}$, where $m_i^W$ is one of the voxelized points of the original full-size point-cloud map. As a result, the vehicle is considered to be located at the position $\hat{x}_W$ in the 3-D map $M_W$.

We define our registration-based pose estimation problem as minimizing the error $d_t$ between the voxelized LiDAR points $z_i^W$ and $M_W$ as follows:

$$d_t = \arg \min_d \sum_{i=0}^n \| (m_i^W - z_i^W) \|_2,$$

where $z_i^W = \{z_1^W, ..., z_k^W\}$ is transformed using the solution of (1) which consists of transformation matrices $T_i$ and $z_i^B$ at time $t$. Therefore, the positioning problem can be considered as estimating $T_i$ at time $t$ because the $T_i \in SE(2)$ aligns $B$ to $W$.

Some studies [54], [55] have implemented various algorithms to attain a much faster frequency than that using the conventional iterative closest point (ICP) variant algorithm [56] to run a registration algorithm; by contrast, in this study, we focus on utilizing a high-density LiDAR point-cloud to register the 3-D points algorithm, except for the feature-based algorithm [57], [58]. However, in the case of high-channel stacked up models, such as 64 or 128 channels (in this study, we installed an OS2-128 model), the LiDAR sensor driver publishes approximately 65 k and 130 k points every 0.1 s, for the 64- and 128-channel models, respectively. Furthermore, the point size of our full-size point-cloud map, which is the general size of the district-scale area, is higher than 10,000 k points. Due to the characteristic of the ICP variant algorithm, depending on the number of inputs and target points, the computational burden severely increases, causing performance degradation. However, the ICP variant algorithm takes advantage of the voxelization approach rather than the normal distributions transform (NDT)-based approach [47], [59], leading to leverage on real-time computing performance [60]. As a result, to perform an efficient registration algorithm up to the typical urban maximum speed of 50 kph,
we employ the generalized iterative closest point (GICP) variant algorithm [60], [61], a registration algorithm for 3-D point-clouds, to model the point-represented environment as a Gaussian distribution, \( z_i^W \sim N(\hat{z}_i, C_i^m) \) and \( M_i^W \sim N(\hat{m}_i, C_i^m) \). Subsequently, the transformation error \( d_i \) can be defined as

\[
d_i = \hat{m}_i - T_{\hat{r}_i} \hat{z}_i.
\]

Thus, the \( d_i \) distribution can be expressed as

\[
d_i \sim N(\hat{m}_i - T_{\hat{r}_i} \hat{z}_i, C_i^m - T_{\hat{r}_i} C_i^m T_{\hat{r}_i}^{T}) = N(0, C_i^m - T_{\hat{r}_i} C_i^m T_{\hat{r}_i}) \tag{3}.
\]

Therefore, the vehicle position can be determined by calculating the \( T_{\hat{r}_i} \) that maximizes the log likelihood of (3), such that

\[
T_{\hat{r}_i} = \arg \max_{T_{\hat{r}_i}} (\log(p(d_i))) = \arg \min_{T_{\hat{r}_i}} \sum_{i} d_i^T (C_i^m - T_{\hat{r}_i} C_i^m T_{\hat{r}_i}) d_i. \tag{4}
\]

Equation (4) can be re-defined such that the cost function \( J(x) \) between the incoming LiDAR data \( z_i^B \) and \( M_i^W \) can be calculated as

\[
J(x) = \arg \max_{x} \sum_{i=0}^{n} \left( -\frac{(M_i^W - T_{\hat{r}_i} z_i^B) \Sigma_i^{-1} (M_i^W - T_{\hat{r}_i} z_i^B)^{T}}{2} \right).
\]

To achieve real-time operation, we utilize a Voxelized-GICP algorithm [60], which extends the conventional GICP algorithm using the voxel-based association approach.

However, despite utilizing an enhanced registration algorithm, there is a still limitation in deploying an autonomous vehicle on the district-scale environment because the target point-cloud \( M_i^W \) size affects the calculation time for the solution of (5). Therefore, we put a sliding-window of the point-cloud map as a limit to the size of \( M_i^W \) to accommodate the registration computing capacity. As the full-size \( M_i^W \) is not used in (5), we register only the points in a radius \( \psi(v_t) \) from the robot position \( \hat{x}_i \). The surrounding radius increases proportionally to the velocity \( v_t \) at time \( t \),

\[
\delta^\psi_t = \| m_i^W - \hat{x}_i \|_2. \tag{6}
\]

where \( \delta^\psi_t \) indicates the distance from \( \hat{x}_i \) to the voxelized 3-D map points \( m_i^W \). Therefore, the points in the sliding-window \( M_i^W = \{ m_i^W, \ldots, m_i^W \} \) replace the entire 3-D map points \( M_i^W \) in (5).

Thus,

\[
m_i^W = \begin{cases} m_i^W, & \text{if } \delta^\psi_t < \psi(v_t), \\ \text{none, otherwise,} \end{cases}
\]

where point \( m_i^W \) is selected from \( M_i^W \), which is determined from \( \delta^\psi_t < \psi(v_t) \). Hence, we limit the registration target according to the LiDAR sensor frequency, which is approximately 10 Hz, without delay.

2) Robustness: The performance of this registration-based scan-matching algorithm is not always guaranteed, particularly for large-scale 3-D maps and dynamic driving scenarios. Because an ICP-variant registration algorithm calculates the solution iteratively to find \( T_{\hat{r}_i} \), the previous transformation matrix can be defined as \( T_{\hat{r}_{i-1}} \). Thus, we can define a scan-matching-based transition distance between \( t \) and \( t - 1 \) as \( \| \hat{x}_t - \hat{x}_{t-1} \|_2 \). Because the ICP-variant registration algorithm starts to compute \( T_{\hat{r}_i} \) from the initial input transform matrix —i.e., in the registration algorithm, it is the initial guess transform matrix — a rational guessing matrix input is crucial for an effective scan-matching performance. In addition, there are parameters for the ICP-variant registration algorithm to constrain the maximum number of iterations \( n_{\max} \) and the searching corresponding-point-distance \( \epsilon_{\text{corres}} \). Because both \( n_{\max} \) and \( \epsilon_{\text{corres}} \) affect the computing time, they are significant in maintaining the localization performance during the relatively high-speed driving. As a result, the registration algorithm can be expected to solve the scan-matching points as

\[
n_{\max} \times \epsilon_{\text{corres}} < \| \hat{x}_t - \hat{x}_{t-1} \|_2. \tag{8}
\]

However, in this study, we use LiDAR odometry for the robustness of the navigation system, without expecting scan-matching performance, while continuously increasing the value of \( n_{\max} \). Therefore, we compute the LiDAR odometry \( x_{1:t}^L = [x_1^L, \ldots, x_{t}^L] \in \mathbb{R}^2 \) based on the LiDAR feature-based algorithm to obtain \( x_t^L = (x_t^L, y_t^L, \theta_t^L) \) [49], [50], [51], [52].

We focus on LiDAR feature-based predictive transformation \( \Delta x_{t+\Delta t}^L \) to estimate the high-probabilistic guessing matrix for the ICP-variant registration algorithm as

\[
\Delta x_{t+\Delta t}^L = T_{L}^{-1}(x_{t+\Delta t}^L - x_t^L), \tag{9}
\]

where \( T_{L}^L(x) \) represents the rigid transformations from \( x_{t+\Delta t}^L \), and \( T_{L}^{-1} \) represents an inverse transformation from \( W \) to \( B \). As a result, we designed our robust localization system to detect when the registration algorithm fails due to the limitations of \( n_{\max} \) and \( \epsilon_{\text{corres}} \).

In the failure-case, we replace \( T_{\hat{r}_{i-1}} \) with \( T_{t+\Delta t} \) —i.e., in the normal case \( T_{\hat{r}_{i-1}} \) is used for next-step registration where the predictive transformation \( T_{t+\Delta t} \) is computed from \( \hat{x}_{t+\Delta t} \), as follows:

\[
\hat{x}_{t+\Delta t} = \hat{x}_t + \Delta x_{t+\Delta t}^L. \tag{10}
\]

Finally, the robust localization algorithm is illustrated in Fig. 3.

3) Resilience: In this study, the use of high-quality DGPS is excluded for autonomous driving near high-rise buildings; instead, a localization algorithm is utilized based on a scan-matching algorithm using LiDAR sensors. Because stability of localization is one of the most crucial factors for operating an unmanned vehicle, the resilience of navigation algorithms is considered using RGB cameras, as well as LiDAR sensors. Our designed resilient-navigation system identifies a failure of LiDAR-based pose estimation and performs lane-detection-based control. First, we compare our estimated pose \( \hat{x}_i \) with \( M_{22}^{W} \) to monitor whether our vehicle keeps to the center of the lane, which can be utilized as a localization health checker.
Second, switching to fail-safe mode, it conducts pose re-initialization itself based on road-marker recognition using an RGB camera. When the solution of (1) is higher than our scan-matching thresholds, the localization health checker gives an alert alarm, and our vehicle changes the control mode from navigation to lane detection. If the solutions of the scan-matching algorithm and the lane-detection-based control are not feasible, this case is considered a localization failure.

The resilient localization algorithm is depicted in Fig. 3.

1) V2X-Enabled Traffic Signal: A human-mimicking method, such as vision-perception-based traffic signal detection, is more intuitive than communication-based traffic signal recognition. However, V2X can be an excellent solution for stability when deploying vehicles in an urban area. In this study, we utilized V2I, which has a communication system between the vehicle and infrastructure [6]. Specifically, we utilized TCP-based V2I in the infra-supported area by sending the red arrow, if the registration algorithm fails, prediction based on LiDAR odometry compensates the failure and enables the robot to implement the next step. Even though a vehicle moves dynamically, LiDAR-based odometry can calculate the pose accurately, thereby reducing the likelihood of scan-matching failures.

To estimate the distance to the object, we implemented the azimuth-aware fusion algorithm, which derives the azimuth of the image bounding box and searches a corresponding LiDAR cluster \( O_i^\text{fusion} \).

For the LiDAR-only detection, we used the ResNet-based keypoint feature pyramid network [64] to convert point-cloud data into bird’s-eye-view images, and utilized the image-net-based approach. Therefore, we can define a detected object \( O_i^\text{lidar} \) using the LiDAR-only method. Subsequently, the detected results \( \{ O_i^\text{fusion}, O_i^\text{lidar} \} \subset \{ O_i \}_{i \in [1:n]} \) obtained from the multi-modal and LiDAR-only methods are pipelined to the multi-object tracking (MOT) algorithm — i.e., we employed the FastMOT [65], which guarantees real-time performance. Finally, the MOT algorithm assigns each object an identifying number.

C. Route Planning

1) Construction of Multi-Layered Road-Graph: One of the most efficient methods of autonomous driving is to use a 2-D semantic road map, which \( M_{2D}^W \) can be utilized for behavior planning and identifying environmental recognition failures. Therefore, we can construct a 2-D semantic road map \( M_{2D}^W \) by accumulating a road-marker feature \( F_{\text{road,rgb}} \) that is detected with a deep-learning-based lane detector [66]. At the same time, the result of the bird’s eye view feature point \( F_{\text{road,rgb}} \) is accumulated on the global coordinates according to \( \hat{x}_i \). After constructing a primary semantic map, we refine \( F_{\text{road,rgb}} \) to build \( M_{2D}^W \). As an equivalent to \( F_{\text{road,rgb}} \), point-cloud-based information \( F_{\text{road,point}} \) is accumulated, where \( F_{\text{road,point}} \) is generated using point reflectivity and a high-pass filtering mask [67]. Therefore, we convert unordered raw point-cloud data to ordered point-cloud data \( \{ P_i \}_{i = 1, \ldots, n} \), where each point \( P_i \) is a vector of \( (x, y, z, r) \), to compute \( F_{\text{road,point}} \) as,

\[
F_{\text{road,point}} = P(r_{1:n} \star H_{\text{sharp}}),
\]

where \( * \) is a convolution operation, \( H_{\text{sharp}} \) is a high-pass filtering(sharpening) mask, and \( r \) is reflectivity.

As a result, we can define the semantic map \( M_{2D}^W \) as being composed of segmented links \( E_{i,m} = (E_{1,i,j,k}, \ldots, E_{n,i,j,k}) \), segmented lanes \( L_{1:m} = (L_{1}, L_{r}, \ldots, L_{m,l,r}) \) and nodes \( N_{1,i,j} = (N_{1,i}, \ldots, N_{l,i}) \) in \( E_i \) where links are accumulated vehicle poses \( \hat{x}_i \) and lanes are detected road-marker features \( \{ F_{\text{road,rgb}}, F_{\text{road,point}} \} \subset L_{m,l,r} \). Each individual link \( E_{i,j,k} \subseteq \{ E_{i,j,k} \} \) and segmented lane \( L_{m,l,r} \) has attributes such as hash-ID, position, and road type. Moreover, we refine the road map using a handcrafted method and include additional attributes for parallel right \( E_{i,j} \) and left \( E_{i,k} \) links, which can be utilized for the behavior planning and path planning. Therefore, we constructed our own point-cloud map and road graph, to ensure the capability of managing the changes in the real-world environment.

2) Graph-Searching Based Route Planning: Given the estimated position \( \hat{x} \), the vehicle is expected to drive along a route to reach the target destination. Let the optimal route be defined as \( p_{1:n}^* = \{ p_1, \ldots, p_n \} \), with size \( n \) of poses. Specifically, the cost of each segmented link is \( E_{i,j,k} \) as \( f^*(E_i) \). We define
\[ P_g(E_i) = \sum_{n=0}^{\infty} f^g(E_i) \] as the accumulated cost of the consecutive link to the destination node \( N_G = \{x_g, y_g\} \). Then, we can define the route planning problem as
\[
p^{*} = \arg \min_{N_S} P_g(E_{i, j, k} | N_S),
\]
where \( N_S \) is the start node. Assuming that road graph \( M_{2D}^W = \{N_i, E_{1,m}, C_{1:m}, \kappa_{1:m}\} \) is built to connect all the previous and subsequent \( N_{j, i} \) and \( E_{i, j, k} \), the planning problem \( p^{*} \) can be defined as searching for a continuous route from the start node \( N_S \) to the target node \( N_G \) based on \( G \). Because the graph-searching based algorithm finds routes on the grid [68], we have adopted the analogous concept for the road-graph. When \( N_S \) is defined from \( \hat{x} \), we maximize \( P_g(\cdot) \) by propagating the graph to arrive at the destination. When the graph-searching reaches the destination, an optimal route is obtained on the visited node using a depth-first searching algorithm [69]. If a route is not obtained on the single consecutive road-graph, \( p_{\text{fin}}^{*} \) is computed based on the propagating parallel lane because our graph can find parallel links, which are represented as \( E_{1,m} \) and \( E_{1,1,m} \) for the left and right parallel links from \( E_{1} \), respectively.

3) Extended Road-Graph: The extended 2-D semantic road graph \( M_{\text{ext}}^W = \{N_i, E_{1,m}, C_{1:m}, \kappa_{1:m}\} \) is defined by further enriching connectivity \( C_{1:m} \) and curvature \( \kappa_{1:m} \) and interpolating between multiple lanes. Here, we adopt the ideas in [70] to precompute the state lattice offline, which is defined in the Frenet space along a \( p_{\text{fin}}^{*} \), where the Frenet frame is defined as the coordinate system spanned by the tangential and normal vectors at any point of the reference line. The state lattice is defined as a discrete function \( [x(s), y(s), \theta(s), \kappa(s)] \) along the arc length \( s \), also called the station. Here, the refined \( M_{\text{ext}}^W \) is formed by lattice layers distributed along the station \( s \) connecting interpolated splined points as depicted in Fig. 4. We utilize \( M_{\text{ext}}^W \) for trajectory prediction of objects. Furthermore, we present the road-graph-searching-based motion planning algorithm using \( M_{\text{ext}}^W \) in Section IV-D.

D. Motion Planning

The task of the motion planning module is to find a collision-free and dynamically feasible path to arrive at a specific goal configuration. There are several major methods in the motion planning field: grid-based, sampling-based, and motion-primitive-based planning. The graph-based motion planners, such as the A* search algorithm [71], [72], [73], [74], [75], find the shortest path based on an admissible heuristic cost function [71], [72]. The planner searches collision-free nodes with a grid-map-based graph representation. Variations of the A* algorithm consider the non-holonomic constraint of autonomous vehicles for urban environment applications [73], [74]. To extend the search space to continuous coordinates, hybrid states containing the discretized cell information and continuous 3-D states \((x, y, \theta)\) were configured. The hybrid representation guarantees the algorithm will search kinematically feasible trajectories for urban driving [75]. The sampling-based methods explore the configuration space using probabilistic node sampling to find a feasible path. A tree of collision-free paths is incrementally extended using random or heuristic steering and collision test functions [76], [77].

With additional routines, such as rewiring, these methods trim the redundant nodes in the tree to obtain a minimum-cost path to a goal configuration [78], [79]. Because of the probabilistic completeness, most of the research focuses on exploiting unstructured environments [80], [81] or planning for high-dimensional systems [82], [83].

1) Road-Graph Searching Based Macro-Motion Planning: Motion planning based on the graph-searching algorithm [71], [72] is considered a milestone in robotics for the solution of obstacle avoidance problems. However, a grid-based graph-searching algorithm is not adaptable for autonomous driving because it does not consider traffic laws. We propose an extended 2-D semantic road graph \( M_{\text{ext}}^W \) that enriches connectivity and interpolates between multiple lanes. In this study, we divide the motion planning algorithm into macro and micro scale, and propose a road-graph based searching algorithm for macro-motion planning. The goal of the macro-motion planning algorithm is to find an optimal trajectory \( p_{\text{macro}}^{*} \) considering obstacles, lane changes and vehicle dynamics. We design the heuristic cost \( g_{I, 1;m}^{\text{macro}} \) as
\[
g_{I, 1;m}^{\text{macro}} = k_O || O_i - N_{I, 1;m} || + k_\kappa \kappa_{1;m} + k_\text{trans} \delta_{\text{macro}}^I + k_{\text{route}} || p_{\text{fin}}^{*} - N_{I, 1;m} || 
\]
considering the distance from the closest node \( N_{I, 1;m} \) to an obstacle \( O_i \), road-curvature \( \kappa_i \), transient path \( \delta_{\text{macro}}^I \), distance from the closest node \( N_{I, 1;m} \) to planned route \( p_{\text{fin}}^{*} \), and weight for each cost \( k_O, k_\kappa, k_\text{trans}, k_{\text{route}} \), respectively. In addition,
\[ \delta_{i}^{\text{macro}} \text{ is considered not to change trajectory dynamically as} \]
\[ \delta_{i}^{\text{macro}} = || p_{i,1:n}^{\text{macro}} - N_{i,1:m} ||, \]  
\[ (14) \]

where \( p_{i,1:n}^{\text{macro}} \) is the optimal trajectory of the previous step, to ensure \( p_{i,1:n}^{\text{macro}} \) not to be a large change. Therefore, a macro-motion planning algorithm can be calculated considering both travel distance and designed heuristic cost as
\[ p_{i,1:n}^{\text{macro}} = \arg \min \limits_{v_{i}} (f_{i,1:n}^{\text{macro}} + s_{i,1:m}^{\text{macro}}). \]  
\[ (15) \]

where travel distance cost \( f_{i,1:n}^{\text{macro}} \) is obtained through the accumulating distance of nodes.

2) Motion Primitives-Based Micro-Motion Planning:

Although the macro-motion trajectory \( p_{i,1:n}^{\text{macro}} \) is already computed, considering obstacles and the road-curvature model, the vehicle cannot possibly avoid obstacles within a safe margin. An accident can happen when the vehicle cannot accurately follow the trajectory due to vehicle dynamic characteristics, such as tire slip or vehicle control performance. Therefore, we utilize a motion primitive-based micromotion planning algorithm that has been widely adopted in recent work [84], [85], [86].

We define a set of positions with size \( k \) \( p_{i,k,1}^{\text{micro}} = \{p_{i,j}^{\text{micro}}, \ldots, p_{i,k}^{\text{micro}}\} \), where each position is defined as \( p_{i,j}^{\text{micro}} = (x_{i,j}^{\text{micro}}, y_{i,j}^{\text{micro}}) \). Therefore, a set of motion primitives can be defined as \( p_{i,j}^{\text{micro}} = (x_{i,j}^{\text{micro}}, y_{i,j}^{\text{micro}}) \), where \( j \) is the number of motion primitives. We generate a motion primitive \( p_{i,k}^{\text{micro}} \) derived from polynomial and macro planning \( p_{i,1:n}^{\text{macro}} \). First, we define \( \begin{array}{c} \mathbf{x}_{t} = (x_{t}, \dot{x}_{t}, \ddot{x}_{t}, y_{t}, \dot{y}_{t}) \end{array} \) and \( \mathbf{x}_{f} = (x_{f}, \dot{x}_{f}, \ddot{x}_{f}, y_{f}, \dot{y}_{f}) \) as the start and final states, and the time interval between the start and final states as \( \Delta t_{i} = t_{f} - t_{i} \). Then, \( p_{i,j}^{\text{micro}} \) is generated by calculating each \( p_{i,j}^{\text{micro}} = (x_{i,j}^{\text{micro}}, y_{i,j}^{\text{micro}}) \), as follows:
\[ x_{i,j}^{\text{micro}}(t) = a_{0} + a_{1}t + a_{2}t^{2} + a_{3}t^{3} + a_{4}t^{4} + a_{5}t^{5}, \]
\[ y_{i,j}^{\text{micro}}(t) = b_{0} + b_{1}t + b_{2}t^{2} + b_{3}t^{3} + b_{4}t^{4} + b_{5}t^{5}, \]  
\[ (16) \]

where the polynomial coefficients for \( x_{i,j}^{\text{micro}} \) and \( y_{i,j}^{\text{micro}} \) are denoted by \( \begin{array}{c} \begin{array}{c} a_{0}, a_{1}, a_{2}, a_{3}, a_{4}, a_{5} \end{array} \end{array}^{T} \) and \( \begin{array}{c} \begin{array}{c} b_{0}, b_{1}, b_{2}, b_{3}, b_{4}, b_{5} \end{array} \end{array}^{T} \), respectively. Here, we can update the initial state of \( \mathbf{x}_{t} \) using the wheel encoder and IMU. Furthermore, we selected a series of target states to be determined using \( w_{\text{goal}} \), and the target speed and acceleration at \( \mathbf{x}_{f} \). Our motion primitive generator can be expressed as follows:

\[
\begin{bmatrix}
0 & 1 & 2s & 3t & 4t^2 & 5t^3 & 6t^4 & 7t^5 \\
0 & 0 & 2 & 6s & 12t^2 & 20t^3 & 30t^4 & 42t^5 \\
0 & 1 & 2f & 3s & 4t & 5t^2 & 6t^3 & 7t^4 \\
0 & 0 & 2 & 6f & 12t^2 & 20t^3 & 30t^4 & 42t^5 \\
0 & 1 & 2f & 3s & 4t & 5t^2 & 6t^3 & 7t^4 \\
0 & 0 & 2 & 6f & 12t^2 & 20t^3 & 30t^4 & 42t^5
\end{bmatrix}\]
\[ \begin{bmatrix}
\begin{array}{c}
x(t)
\end{array}
\end{bmatrix}^{-1}\begin{bmatrix}
y(t)
\end{bmatrix}.
\]  
\[ (17) \]

Subsequently, we convert the generated \( p_{i,1:n}^{\text{micro}} \) consisting of \( p_{i,j}^{\text{micro}} = \{x_{i,j}^{\text{micro}}, y_{i,j}^{\text{micro}}\} \) into curvilinear coordinates.

Therefore, a single motion primitive, i.e., the set of positions \( p_{i,k,1}^{\text{micro}} = \{p_{i,1}^{\text{micro}}, \ldots, p_{i,k}^{\text{micro}}\} \), is modeled using an arc-length-based cubic spline, such that
\[ s_{i,j}^{\text{micro}} = \frac{n-1}{i=0} \sqrt{(x_{i,j}^{\text{micro}} - x_{i,j}^{\text{micro}})^2 + (y_{i,j}^{\text{micro}} - y_{i,j}^{\text{micro}})^2}, \]
\[ = \sum_{i=0}^{n-1} s_{i,j}^{\text{micro}}. \]  
\[ (19) \]

We define the motion primitives converted to curvilinear coordinates as \( P_{i,j}^{\text{micro}} = [p_{i,j}^{\text{micro}}] \). We also label a converted single primitive \( p_{i,j}^{\text{micro}} = \{p_{i,j}^{\text{micro}}, \ldots, p_{i,k}^{\text{micro}}\} \). Our final motion primitive can be computed as
\[ x_{i,j}^{\text{micro}} = a_{x}(s_{i,j}^{\text{micro}} - s_{i,j}^{\text{micro}})^3 + b_{x}(s_{i,j}^{\text{micro}} - s_{i,j}^{\text{micro}})^2, \]
\[ y_{i,j}^{\text{micro}} = a_{y}(s_{i,j}^{\text{micro}} - s_{i,j}^{\text{micro}})^3 + b_{y}(s_{i,j}^{\text{micro}} - s_{i,j}^{\text{micro}})^2, \]  
\[ (20) \]
\[ (21) \]

where the arc length \( s^{\text{micro}} \) is the cumulative distance sum of referential path \( p_{i,1:n}^{\text{macro}} \). In addition, coefficients \( a_{x,y}, b_{x,y}, c_{x,y}, \) and \( d_{x,y} \) of the cubic spline can be calculated using the boundary conditions of the first and second derivatives [87]. We propose cost function \( g_{\text{mic}}(p_{i,k}^{\text{micro}}, \mathbf{p}_{i,1:n}^{\text{macro}}) \) to obtain a micro optimal trajectory \( P_{i,1:n}^{\text{micro}} \) considering obstacles, curvature and vehicle dynamics as
\[ g_{\text{mic}}(p_{i,k}^{\text{micro}}, \mathbf{p}_{i,1:n}^{\text{macro}}) = w_{\text{ic}}\mathcal{L}_{1:k} + w_{\text{trans}}\mathcal{L}_{1:k} + w_{\text{trans}}\mathcal{L}_{1:k}. \]  
\[ (22) \]

where \( \mathcal{L}_{1:k} = \mathcal{L}_{1:k}^{\text{micro}}, \mathcal{L}_{1:k}^{\text{macro}} \), the second term represents the sum of the curvature of the primitive; the third term is the final transient state compared with the previous optimal primitive, and the weight for each cost \( w_{\text{ic}}, w_{\text{trans}} \), respectively. In addition, \( \delta_{i}^{\text{micro}} \) is considered not to change trajectory dynamically as
\[ \delta_{i}^{\text{micro}} = || p_{i,1:n}^{\text{micro}} - p_{i,k}^{\text{micro}} ||. \]  
\[ (23) \]

where \( P_{i,1:n}^{\text{micro}} \) is the micro optimal trajectory of the previous step, to ensure \( P_{i,1:n}^{\text{micro}} \) not to be a large transient. Therefore, micro-motion planning algorithm can be calculated considering both travel distance and designed heuristic cost as
\[ p_{i,1:n}^{\text{micro}} = \arg \min \limits_{v_{i}} (f_{i,1:n}^{\text{micro}} + s_{i,1:m}^{\text{micro}}), \]  
\[ (24) \]

where travel distance cost \( f_{i,1:n}^{\text{micro}} \) is obtained by the accumulating distance of nodes.
E. Behavior Planning

We proposed a route planning algorithm to calculate the minimum-transversal-distance route on the road-graph in IV-C. While driving to the destination in a congested urban environment, there are a few scenarios for overtaking other vehicles. Specifically, a complex behavior planning strategy is demanded for safe overtaking. The planning algorithm must be able to execute various lateral maneuvers considering the location, intentions of surrounding vehicles, and signals from the traffic infrastructure. Furthermore, because a naive conservative-spacing algorithm may prohibit the reduction of headway and discourage overtaking maneuvers, the ego vehicle needs to infer efficient longitudinal velocity according to the distances from not only the front vehicle but also the vehicle on the overtaking side.

To tackle those above-mentioned challenges, we designed a task-specific path selection (TSPS) algorithm that can derive a feasible high-level lateral maneuver considering the surrounding vehicles, traffic environment, and state and intent of the ego vehicle. Furthermore, we designed a geometry-aware velocity planning (GVP) algorithm that can consider the geometric relationship between the ego and surrounding vehicles for safe overtaking.

1) Task-Specific Path Selection: We present a TSPS module to consider multiple options and determine optimal high-level lateral maneuvering for various scenarios. To simplify this complex behavior planning problem, we divide lateral maneuver into three path options: global optimal route, overtaking/avoidance trajectory, and ego lane. More precisely, ego lane is computed from the pre-built road-graph. Let our neighbor road-graph set \( p_{n}^{a,b} \) be defined as

\[
P_{n}^{a,b} = \{ N_{i,j}, E_{j} \mid d_{j}^{a,b}, \hat{x}_{i} \}_{j \in [1,n]},
\]

where \( d_{j}^{a,b} \) is a Euclidean distance from \( \hat{x}_{i} \) to the road-graph set. Subsequently, we can extract the closest lane to the vehicle in the road-graph set as

\[
P_{i,j}^{ego} = \arg \min_{\forall d_{j}^{a,b}} (p_{n}^{a,b}).
\]

As a result, we implement the TSPS algorithm to derive a feasible path from a set of path models from the original route, motion-planning path, and closest lane from the road graph, respectively, \( p_{n}^{ego} \), \( p_{n}^{micro} \), and \( p_{n}^{ego} \) as

\[
p_{nsp} = \begin{cases} p_{n}^{ego} \\ p_{n}^{micro} \\ p_{n}^{ego}
\end{cases}
\]

The TSPS algorithm follows the hierarchical process shown in Fig. 5. First, if the original global route is feasible, the TSPS derives this route as the final path without additional operation. If not, the TSPS then checks whether the ego vehicle is stuck. Except in the case of a red traffic signal, the algorithm accumulates a stuck counter at every moment of zero velocity status and decides whether the vehicle is stuck according to the value of the counter. Because a long stuck status degrades the overall progress of the ego vehicle, after a few stuck steps, the TSPS derives a micro-motion path to induce the ego vehicle to escape from the stuck situation. If the ego vehicle is in normal status, the TSPS examines the viability of the local path (micro path). If the path is not feasible, it derives the original route for the ego vehicle to keep its lane without changing lanes or overtaking. If the vehicle is close to a traffic signal, the TSPS considers the following driving scenarios: turn-left, turn-right, or go-straight. Because the ego vehicle needs to follow the direction of the route exactly, the algorithm outputs the original route as the final path when the scenario is turn-left or turn-right for the vehicle not to deviate from the original route. In the go-straight scenario, the algorithm returns the micro-local path, as the dependency on the original route direction is not prominent. In the aforementioned process, even if the local path is feasible, following the local path and performing immediate overtaking is risky. The safety of the side area must be considered before executing lateral maneuvers through the micro-local path. Therefore, if there is a surrounding vehicle in the side area, the TSPS outputs the closest neighboring path instead of the local path for the ego vehicle to keep its lane, while maintaining a safe distance. Consequently, the ego vehicle performs safe lateral maneuvers only when the safe-overtaking condition is satisfied.

2) Geometry-Aware Velocity Planning: For safe overtaking in urban environments, the ego vehicle must plan progressive longitudinal maneuvers, while maintaining safe headway from the vehicles on both the front and overtaking sides. However, a naive velocity-planning strategy, such as adaptive cruise control, generates velocity commands considering the forward headway only, which may not allow reducing the distance from the front vehicle and may discourage overtaking maneuvers. Moreover, because the ego vehicle’s overtaking switches the target of the front vehicle, it is necessary to consider the geometric relationship between the ego and surrounding vehicles during the overtaking transition. Therefore, we designed the GVP algorithm to consider the dynamic-geometric relationship between the ego and surrounding vehicles for safe overtaking scenarios. We implemented the algorithm with design factors that can 1) operate in both general-lane-following and
overlapping scenarios with a single algorithm and 2) alter the aggressiveness of overlapping through simplified parameters.

The GVP calculates two velocity commands, \( v_T \) and \( v_E \), which are computed using an adaptive cruise control algorithm considering the vehicle on the target lane \( p^\text{target} \) and ego lane \( p^\text{ego} \) respectively. Those velocity commands are used to derive new velocity plans \( v^\text{micro} \) and \( v^\text{ego} \) considering the road geometry during overlapping as

\[
\begin{align*}
v^\text{micro} &= \left(1 - \frac{d^T e}{d^T e + d^E e}\right) v_T + \left(1 - \frac{d^E t}{d^T t + d^E t}\right) v_E, \\
v^\text{ego} &= \left(1 - \frac{d^T e}{d^T e + d^E e}\right) v_T + \left(1 - \frac{d^E t}{d^T t + d^E t}\right) v_E
\end{align*}
\]

where \( d^T \) and \( d^E \) are the distances from the terminal point of \( p^\text{micro} \) to \( p^\text{target} \) and \( p^\text{ego} \) respectively, and \( d^T \) and \( d^E \) are the distances from the center of gravity of the ego vehicle to \( p^\text{target} \) and \( p^\text{ego} \) respectively.

The final velocity command of GVP is then computed as

\[
v^\text{ref} = \tau v^\text{micro} + (1 - \tau) v^\text{ego}
\]

where \( \tau \) is the aggressiveness factor. Consequently, vehicle operators could set a hyper-parameter \( \tau \) before autonomous driving begins.

V. TRAFFIC ANALYSIS

In Section IV-E, we proposed a driving model using TSPS and GVP algorithms, which can determine the aggressiveness of the driving model for overlapping and lane changes. Additionally, we designed a behavior planning algorithm that will obey traffic laws at the critical penalty area. Owing to these traffic constraints, most vehicles will drive analogously and generate traffic congestion. Therefore, we have expanded our study to include traffic analysis for multiple autonomous vehicles in traffic-congested situations.

In recent years, there have been numerous studies on cooperative traffic systems utilizing V2X communication, which includes vehicle-to-vehicle (V2V) and V2I. Without cooperative driving using communication, autonomous vehicles can have conflicting optimal conditions. Especially, in an autonomous driving competition, each team can assume that every other team will drive competitively, rather than cooperatively, for the race. Our concern was similar to this conflict that transversal-time-minimizing planning can be analogous between autonomous vehicles considering minimum distance, curvature, or transversal time.

In this section, we present a methodology for traffic analysis to determine the conflict of autonomous vehicles’ optimality. Specifically, to measure the similarity of spatial distribution, we measure a Kullback-Leibler divergence (KLD) estimation for spatial data \( P(x|z) \) and \( Q(x|z) \), with a finite \( k \). In [88], the author proved the estimation of the divergence for vectorial data using the k-th nearest-neighbor density estimate as

\[
D_{\text{KLD}}(P || Q) = \sum_{x \in \mathcal{X}} P_k(x|z) \log \left( \frac{P_k(x|z)}{Q_k(x|z)} \right),
\]

where \( z \) is the reference racing line or road-graph and \( P(x|z) \) can be considered the conditional distribution of the spatial data. Assuming the given spatial data \( P(x|z) \) and \( Q(x|z) \) have \( m \) and \( n \) samples of 2-D data, respectively, we can define the KLD for the spatial distribution as

\[
D_{\text{KLD}}(P || Q) = \sum_{x \in \mathcal{X}} P_k(x|z) \log \left( \frac{P_k(x|z)}{Q_k(x|z)} \right) + \log \left( \frac{m}{n-1} \right),
\]

where

\[
P_k(x|z) = \frac{k}{(n-1)} \frac{\Gamma(d/2+1)}{\pi^{d/2} \rho_k(z)}.
\]

In Eq. (32), \( \rho_k(x) \) and \( \rho_k(x) \) are the Euclidean distances to the \( k \)-th nearest neighbor of \( x \in \mathcal{X} \). In addition, \( \pi^{d/2} / \Gamma(d/2+1) \) is the volume of the unit-ball in \( \mathbb{R}^d \).

Furthermore, we analyze the small-scale data with the Euclidean distance error between the two autonomous vehicles data \( \mathcal{X}, \mathcal{Y} \) to understand the mean error. We compute the mean error value as

\[
\tilde{d}_{X,Y} = \frac{1}{n} \sum_{x \in \mathcal{X}, y \in \mathcal{Y}} (x - r_k(y)),
\]

where \( n \) is the size of the data \( \mathcal{X} \), and \( r_k(x) \) is the Euclidean distances to the \( k \)-th nearest-neighbour of \( x \in \mathcal{X} \).

As a result, we can utilize Eqs. (31) and (33) to understand the spatial data of various autonomous vehicles in the scope of traffic analysis.

VI. RESULTS

A. Test Environment

1) Sangam Dataset: In this study, several experiments were conducted using the Sangam real-world dataset that contains 128-channel LiDAR points, front and rear RGB camera images, a single IMU, and vehicle-state data. We also annotate data from vehicles that can be utilized for object detection and tracking. In the real world, we evaluated the proposed autonomous stack of system modules in a mixed traffic environment, where both autonomous and human-driving vehicles were deployed.
2) Simulation Dataset: We also present a simulation environment that enables us to evaluate our proposed system in various scenarios. When evaluating algorithms for motion and behavior planning, it is difficult to repeatedly perform the same experiment through field testing. To end this, we utilized the IPG CarMaker simulator distributed by Hyundai Motor Company. In the simulation, the Sangam area, where field tests were conducted, is simulated, allowing us to implement virtual test scenarios for the autonomous vehicles in the application areas, with high-resolution 3-D visualization photorealistic quality.

B. Resilient Localization

We evaluated our proposed resilient localization system, which does not depend on GPS, in the urban city area. Our evaluation of localization can be divided into two components: registration algorithm selection and performance of the resilient system. In the pioneering studies [54], [56], [57], [58], [60], [61], the authors proposed a series of registration algorithms that can measure corresponding points between 3-D input and target data. Thanks to these studies, we can implement a non-GPS-based localization system utilizing a 3-D pre-built point-cloud map. We considered two main real-time registration algorithms that are implemented using multithreading accelerated computing, named Voxelized-GICP [60] and NDT-OMP [47], [59].

As shown in Table I, parameters of registration affect the performance of the scan matching-based localization algorithm. To identify the impact of each parameter, we conducted a parametric study by altering the input cloud channel, sensing radius, sliding-window map radius, voxel size, and matching error threshold. We focused on determining the desired output that can cover the urban map scale, computing time, and performance at high speed. While selecting the registration algorithm, we implemented our proposed sliding-window-based registration algorithm because using the entire map is too slow to run even a few steps of registration. However, only the registration-based scan-matching algorithm has a limitation in a large-scale environment. Therefore, we implemented robust navigation using LiDAR odometry for registration algorithm evaluation, and our proposed method could determine the dynamic movement of the vehicle. After evaluating a series of registrations with various parameters, we elected to use Voxelized GICP with a coarse voxel size of 1.0 m, which was verified up to a driving speed of 60 kph in the urban area. The results of the resilient localization algorithm are depicted in Fig. 7. Our proposed methods can estimate the pose in the urban area utilizing LiDAR odometry and scan-matching resiliently. (b) LiDAR odometry shows a drift on the straightaway. In addition, the single scan-matching method without our proposed method diverges after a left-turn at the intersection.
The proposed method has the lowest average error while still maintaining a high frequency. A combination of scan matching and LiDAR odometry was utilized for this purpose. Additionally, the scan matching algorithm was improved using the proposed method. In cases where only scan matching is used, scan matching failed when the vehicle was driven dynamically.

| Method                  | Proposed | LiDAR odometry | Scan match (w. proposed) | Scan match (w/o. proposed) |
|-------------------------|----------|----------------|--------------------------|----------------------------|
| Average error (m)       | 0.05678  | 1.0162         | 0.0314                   | Fail                       |
| Number of data          | 82,137   | 9,157          | 6,399                    | 1,943                      |

Fig. 8. The proposed overtaking trajectory was visualized. A collision-free path can be generated by combining macro and micro overtaking paths. (a) Using the simulation environment, macro overtaking paths are visualized. (b) The macro overtaking path can be adjusted based on motion primitives if the micro overtaking path is enabled. We conducted this evaluation in the real world.

matching frequency was low. Moreover, LiDAR odometry is also able to cover the entire map, but it has a higher average error in the large-scale map. Therefore, we verified that the proposed algorithm was able to estimate the vehicle pose with high-frequency covering the speed up to 60 kph without using a GPS. In addition, we validated our proposed method in an area of over 400,000 m² and a course length of approximately 5 km.

C. Motion Planning

We have proposed road-graph searching-based planning and motion primitives-based planning for macro and micro motion planning algorithms, respectively. In addition, as we implemented the TSPS algorithm to select an optimal path according to driving situations, we designed our motion planning system to find a collision-free path utilizing a road-graph and motion primitives. As a result, we evaluated our motion planning algorithm in the simulation environment as well as in real-world experiments as illustrated in Fig. 8. Furthermore, in the competition event, we validated our algorithm that enabled our vehicle to overtake the congested traffic situation as shown in Fig. 9.

D. Behavior Planning

Before real-world deployment, we studied the quantitative evaluation of the TSPS and GVP algorithms in the simulated...
Fig. 11. The result of the positioning log data are depicted. We could obtain real-world data from all the autonomously driven vehicles. Therefore, we could extract the similarities between the five teams in terms of spatial factors. In addition, we separated the situations in terms of curve (Area: A, B, D-G), intersection (Area: C), and straight road (Area: H, I).

TABLE III

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 13.343 | 14.286 | 13.452 | 14.159 |
|       | - (Mean) | 12.461 | 2.21 | 13.037 | 1.58 |
| CBNU  | - - | 0.943 | 0.109 | 0.812 |
| INU   | - - | 1.764 | 6.209 | 1.569 |
| INHA  | - - | - | 0.833 | 0.131 |
|       | - - | 10.016 | 0.792 |
|       | - - | - | 0.702 |
|       | - - | - | 1.158 |

Traffic analysis was conducted by comparing the trajectories of participating teams. Vehicle position data were collected from the location information every team sent to the infrastructure for traffic signal recognition. One of the six teams had technical issue in location feed, and is excluded in the analysis. The result of the positioning log data are shown in Fig. 11. In addition, we conducted section-wise analyses including intersection, curve, and straight section. We found that curve and intersection sections are the most critical component to win the competition, as the efficient maneuver in the congested traffic environment with the various aggressiveness factors \( \tau \). Because the factor can change the resultant behavior of the ego vehicle, we investigated the lateral and longitudinal maneuvers concerning five different aggressiveness factors. Figure 10 (left) shows a driven trajectory of the ego and front vehicles during an overtaking scenario. While the TSPS generated collision-free motion plans, the GVP computes proper velocity plans considering the dynamic geometric relationship during the lateral overtaking maneuver. We further evaluated our algorithms with longitudinal metrics, such as velocity and progress. Because a lower \( \tau \) makes the ego vehicle conservative, the most considerable velocity and terminal progress drop was at \( \tau = 0 \). By contrast, when \( \tau = 1 \), the ego vehicle performed overtaking with almost no velocity reduction and showed the highest terminal progress. Considering the simulated environment allows perfect observation compared to the real world, we choose \( \tau = 0.75 \) which shows a minor velocity drop and progress decrement in real-world deployment.

**E. Traffic Analysis**

Traffic analysis was conducted by comparing the trajectories of participating teams. Vehicle position data were collected from the location information every team sent to the infrastructure for traffic signal recognition. One of the six teams had technical issue in location feed, and is excluded in the analysis. The result of the positioning log data are shown in Fig. 11. In addition, we conducted section-wise analyses including intersection, curve, and straight section. We found that curve and intersection sections are the most critical component to win the competition, as the efficient maneuver in the congested in the corners and intersections greatly affect the overall performance.

The results of the spatial similarity between the teams are presented in Table III using Eqs. (32) and (33). From the Table III, one can observe that traffic data from our team (KAIST) differed from other team data, owing to whether

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 13.211 | 14.396 | 19.807 | 11.741 |
|       | - (Mean) | 1.977 | 1.896 | 2.280 | 2.126 |
| CBNU  | - - | 0.452 | 3.946 | 0.924 |
| INU   | - - | 0.497 | 0.872 | 0.350 |
| INHA  | - - | - | 3.230 | 1.386 |
|       | - - | - | 0.849 | 0.737 |
|       | - - | - | 5.306 |
|       | - - | - | 0.844 |

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 13.769 | 13.611 | 13.660 | 17.509 |
|       | - (Mean) | 1.699 | 1.838 | 1.629 | 2.535 |
| CBNU  | - - | 0.111 | 0.398 | 2.407 |
| INU   | - - | 0.260 | 0.860 | 0.352 |
| INHA  | - - | - | 0.375 | 2.863 |
|       | - - | - | 0.734 | 0.539 |
|       | - - | - | 3.604 |
|       | - - | - | 1.180 |

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 15.506 | 26.581 | 22.547 | 27.501 |
|       | - (Mean) | 1.433 | 1.676 | 1.169 | 1.563 |
| CBNU  | - - | 7.073 | 3.750 | 8.588 |
| INU   | - - | 0.712 | 0.559 | 0.582 |
| INHA  | - - | - | 2.827 | 0.161 |
|       | - - | - | 0.470 | 0.615 |
|       | - - | - | 3.399 |
|       | - - | - | 0.799 |

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 11.102 | 12.517 | 19.334 | 15.618 |
|       | - (Mean) | 1.870 | 1.648 | 2.854 | 1.862 |
| CBNU  | - - | 0.601 | 8.581 | 3.158 |
| INU   | - - | 0.474 | 1.721 | 0.354 |
| INHA  | - - | - | 7.983 | 2.336 |
|       | - - | - | 1.606 | 0.728 |
|       | - - | - | 5.539 |
|       | - - | - | 1.696 |

| Team  | KAIST | CBNU | INU  | INHA | SKLU |
|-------|-------|------|------|------|------|
| KAIST | - (KLD) | 15.278 | 14.406 | 22.287 | 22.115 |
|       | - (Mean) | 2.194 | 2.279 | 1.682 | 2.534 |
| CBNU  | - - | 0.405 | 6.006 | 4.993 |
| INU   | - - | 0.239 | 0.483 | 0.608 |
| INHA  | - - | - | 6.472 | 5.535 |
|       | - - | - | 0.616 | 0.807 |
|       | - - | - | 0.846 |
|       | - - | - | 1.046 |
GPS was utilized in the localization method—i.e., because the global position was transformed from our 3-D pointcloud map without using GPS, a continuous error appeared for the entire route. However, when comparing teams other than ours with each other, both the KLD and RMSE values indicate high similarities in routes for the entire race. This result implies that every team is likely to utilize a route related to obeying traffic laws, minimizing distance, and minimizing lane changes. The section-specific comparisons are presented in Tables IV - VII.

In addition, when the number of spatial data is insufficient to compare data distributions, the similarity between teams was quantified using RMSE rather than KLD.

![Fig. 12. Simulation screen is depicted. The simulation qualification was a timed trial composed of eight missions: overtaking a low-speed vehicle, handling a cut-in vehicle, passing a narrow environment, accident vehicle avoidance, passing un-signalized intersection, and following the traffic signals.](image-url)

### VII. Conclusion

In this paper, we proposed a full autonomous driving software stack to deploy a competitive driving model covering module-wise autonomous driving modules. In addition, autonomous modules consisting of navigation, perception, and planning systems were developed to enable our autonomous...
vehicle to be deployed in a complex urban area. We evaluated our system in the real world and won a competition for full-scale autonomous vehicles in South Korea. The traffic analyses provided additional insight on a multi-agent autonomous vehicle driving environment. The similarity in the projected optimal route of individual teams created conflicts among vehicles, which resulted in the performance loss. The strong performance of our architecture demonstrated the importance of adaptable approach in non-cooperative multi-agent autonomous driving.

APPENDIX

This appendix provides additional results of the competition for simulation qualification and real-world competition. 23 teams competed for simulation qualification, and only six teams qualified for final. The simulation qualification was a timed trial composed of eight missions: overtaking a low-speed vehicle, handling a cut-in vehicle, passing a narrow passage, timed trial composed of eight missions: overtaking a low-speed vehicle, handling a cut-in vehicle, passing a narrow passage, accident vehicle avoidance, passing un-signalized intersection, and following the traffic signals, as illustrated in Fig. 12.

After qualification, the six qualifying teams were supported by Hyundai Motors providing an electronic vehicle, the KIA Niro. On November 29th, 2021, the main event was held in Sangam, Seoul, which is a complex area where media and broadcasting companies are located in Seoul, and there are many steel structures and high-rise buildings, causing a GPS-degraded environment also known as urban canyons. In the final competition, our team won with the minimum transverse time. While driving autonomously, our vehicle was not penalized due to traffic laws or threatening driving. In addition, our vehicle overtook other teams four times, while detecting a low-speed vehicle and congested traffic scenarios.

The results of the final event are depicted in Fig. 13 and listed in Table VIII. To enhance the safety of the autonomous vehicle, there were critical penalties for traffic violations in the final event. Moreover, the full competition video is available at https://youtu.be/EJD34qMe768.

REFERENCES

[1] W. Morales-Alvarez, O. Sipele, R. Léberon, H. H. Tadjine, and C. Olavierri-Monreal, “Automated driving: A literature review of the take over request in conditional automation,” Electronics, vol. 9, no. 12, p. 2087, Dec. 2020.
[2] D. J. Fagnant and K. Kockelman, “Preparing a nation for autonomous vehicles: Opportunities, barriers and policy recommendations,” Transp. Res. A, Policy Pract., vol. 77, pp. 167–181, Jul. 2015.
[3] I. Shim et al., “An autonomous driving system for unknown environments using a unified map,” IEEE Trans. Intell. Transp. Syst., vol. 16, no. 4, pp. 1999–2013, Aug. 2015.
[4] K. Jo et al., “Overall reviews of autonomous vehicle AI—system architecture and algorithms,” IFAC Proc. Volumes, vol. 46, no. 10, pp. 114–119, Jun. 2013.
[5] U. Lee et al., “EureCar turbo: A self-driving car that can handle adverse weather conditions,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2016, pp. 2301–2306.
[6] C. Jung, D. Lee, S. Lee, and D. H. Shim, “V2X-communication-aided autonomous driving: System design and experimental validation,” Sensors, vol. 20, no. 10, p. 2903, May 2020.
[7] S. Penelton et al., “Perception, planning, control, and coordination for autonomous vehicles,” Machines, vol. 5, no. 1, p. 6, Feb. 2017.
[8] S. Kato et al., “Autoware on board: Enabling autonomous vehicles with embedded systems,” in Proc. ACM/IEEE 9th Int. Conf. Cyber-Phys. Syst. (ICCPs), Apr. 2018, pp. 287–296.
[9] X. Zhang, J. Sun, X. Qi, and J. Sun, “Simultaneous modeling of car-following and lane-changing behaviors using deep learning,” Transp. Res. C, Emerg. Technol., vol. 104, pp. 287–304, Jul. 2019.
[10] C. Y. Jung, H.-K. Seong, and D. H.-C. Shim, “Development of the end-to-end learning based autonomous driving framework and experiments on a full-scale autonomous vehicle,” J. Inst. Control, Robot. Syst., vol. 26, no. 5, pp. 342–347, May 2020.
[11] Y. Zhang, Q. Xu, J. Wang, K. Wu, Z. Zheng, and K. Lu, “A learning-based discretionary lane-change decision-making model with driving style awareness,” IEEE Trans. Intell. Transp. Syst., vol. 24, no. 1, pp. 68–78, Jan. 2023.
[12] X. Ji, X. He, C. Lv, Y. Liu, and J. Wu, “Adaptive- neural-network-based robust lateral motion control for autonomous vehicle at driving limits,” Control Eng. Pract., vol. 76, pp. 41–53, Jul. 2018.
[13] X. Zhang, H. Ma, Z. Wang, M. Fan, and B. Zhao, “Model free adaptive control algorithm based on ReOSELM for autonomous driving vehicles,” in Proc. 40th Chin. Control Conf. (CCC), Jul. 2021, pp. 3803–3809.
[14] Y. Zhang, C.-Y. Chan, and C.-Y. Hung, “A reinforcement learning based approach for automated lane change maneuvers,” in Proc. IEEE Intell. Veh. Symp., Jun. 2018, pp. 1379–1384.
[15] H. Seong, C. Jung, S. Lee, and D. H. Shim, “Learning to drive at unsignalized intersections using attention-based deep reinforcement learning,” in Proc. IEEE Intell. Transp. Syst. Conf. (ITSC), Sep. 2021, pp. 559–566.
[16] Y. Dai and S.-G. Lee, “Perception, planning and control for self-driving system based on on-board sensors,” Adv. Mech. Eng., vol. 12, no. 9, Sep. 2020, Art. no. 168781402095694.
[17] S. Tokunaga, Y. Horita, Y. Oda, and T. Azumi, “IDF-Autoware: Integrated development framework for ros-based self-driving systems using MATLAB/simulink,” in Proc. Workshop Auto. Syst. Des., 2019, pp. 1–9.
[18] Z. Zang, R. Tumu, J. Betz, H. Zheng, and R. Mangharam, “Winning the 3rd Japan automotive AI challenge—Autonomous racing with the autoware. Auto open source software stack,” 2022, arXiv:2206.00770.
[19] H. Fan et al., “Baidu Apollo EM motion planner,” 2018, arXiv:1807.08048.
[20] V. M. Raju, V. Gupta, and S. Lomate, “Performance of open autonomous vehicle platforms: Autoware and Apollo,” in Proc. IEEE 5th Int. Conf. Converg. Technol. (ICT), Mar. 2019, pp. 1–5.
[21] A. Cosgun et al., “Towards full automated drive in urban environments: A demonstration in GoMentum station, California,” in Proc. IEEE Intell. Vehicles Symp. (IV), Jun. 2017, pp. 1811–1818.
[22] S. Thrun et al., “Stanley: The robot that won the DARPA grand challenge,” J. Field Robo., vol. 23, no. 9, pp. 661–692, 2006.

[23] A. Agha et al., “NeBula: Quest for robotic autonomy in challenging environments; TEAM CoSTAR at the DARPA subterranean challenge,” 2021, arXiv:2103.11470.

[24] J. Song and J. Alves-Foss, “The DARPA cyber grand challenge: A competitor’s perspective,” IEEE Secur. Privacy, vol. 13, no. 6, pp. 72–76, Nov. 2015.

[25] J. Lim, “Robot system of DRC-HUBO+ and control strategy of team KAIST in DARPA robotics challenge finals,” J. Field Robo., vol. 34, no. 4, pp. 802–829, Jun. 2017.

[26] M. O’Kelly, H. Zheng, D. Karthik, and R. Mangharam, “F1tenth: A portable device for autonomous racing,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2021, pp. 14484–14490.

[27] T. Shan and B. Englot, “LEGO-LOAM: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain,” in Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS), Oct. 2018, pp. 4758–4765.

[28] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, “LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping,” 2020, arXiv:2009.07258.

[29] W. Xu and F. Zhang, “FAST-LIO: A fast, robust LiDAR-inertial odometry package by tightly-coupled iterated Kalman filter,” IEEE Robot. Autom. Lett., vol. 6, no. 2, pp. 3317–3324, Apr. 2021.

[30] T.-M. Nguyen, M. Cao, S. Yuan, Y. Lyu, T. H. Nguyen, and L. Xie, “LIRO: Tightly coupled LiDAR-inertia-ranging odometry,” in Proc. IEEE Int. Conf. Robot. Autom. (ICRA), May 2021, pp. 14484–14490.

[31] L. Sommer and A. Koch, “OpenMP device offloading for embedded heterogeneous platforms—true-in-progress,” in Proc. Int. Conf. Embedded Softw. (EMSOFT), Sep. 2020, pp. 4–6.

[32] P. Dellenbach, J. Deschaud, B. Jacquet, and F. Goullete, “CT-ICP: Real-time elastic LiDAR odometry with loop closure,” in Proc. Int. Conf. Robot. Autom. (ICRA), May 2022, pp. 5580–5586.

[33] D. Chetverikov, D. Svirko, D. Stepanov, and P. Krsek, “The trimmed iterative closest point algorithm,” Proc. Object Recognit. Supported User Interface. Service Robots, vol. 3, pp. 545–548, Aug. 2002.

[34] R. B. Rusu, N. Blidow, and M. Beetz, “Fast point feature histograms (FPFH) for 3D registration,” in Proc. IEEE Int. Conf. Robot. Autom., May 2009, pp. 3212–3217.

[35] S. Salti, F. Tombari, and L. Di Stefano, “SHOT: Unique signatures of histograms for surface and texture description,” Comput. Vis. Image Understand., vol. 125, pp. 251–264, Aug. 2014.

[36] K. Koide, J. Miura, and E. Menegatti, “A portable three-dimensional LiDAR-based system for long-term and wide-area people behavior measurement,” Int. J. Adv. Robotic Syst., vol. 16, no. 2, Mar. 2019, Art. no. 172988141984153.

[37] K. Koide, M. Yokozuka, S. Oishi, and A. Banno, “Voxelized GICP for fast and accurate 3D point cloud registration,” in Proc. IEEE/RSJ Int. Conf. Robot. Autom. (ICRA), May 2021, pp. 11054–11059.

[38] J. Servos and S. L. Waslander, “Multi-channel generalized-ICP: A robust framework for multi-channel scan registration,” Robot. Auto., vol. 87, pp. 247–257, Jan. 2017.

[39] J. B. Kenney, “Dedicated short-range communications (DSRC) standards in the United States,” Proc. IEEE, vol. 99, no. 7, pp. 1162–1182, Jul. 2011.

[40] H. Park, A. Miloslavov, J. Lee, M. Veeraraghavan, B. Park, and B. H. Kim, “Integrated traffic–communication simulation evaluation environment for IntelliDrive applications using SAE J2735 message sets,” Transp. Res. Rec. J. Transp. Res. Board, vol. 2243, no. 1, pp. 117–126, Jan. 2011.

[41] P. Li, H. Zhao, P. Liu, and F. Cao, “RTM3D: Real-time monocular 3D detection from object keypoints for autonomous driving,” in Proc. ECAI’20. Comput. Vis. (ECCV), 2020, pp. 644–660.

[42] Y. Yang, “FastMOT: High-performance multiple object tracking based on deep SORT and KLT,” J. Korea Multimed. Soc., vol. 20, pp. 893–910, 2020.

[43] Z. Wang, W. Ren, and Q. Qiu, “LaneNet: Real-time lane detection networks for autonomous driving,” 2018, arXiv:1807.01726.

[44] E. Pardo-Iguzquiza, M. Chica-Olmo, and P. M. Atkinson, “Downscaling cokriging for image sharpening,” Remote Sens. Environ., vol. 102, nos. 1–2, pp. 86–98, May 2006.

[45] F. Duchêne et al., “Path planning with modified a star algorithm for a mobile robot,” Proc. Eng., vol. 96, pp. 59–69, Dec. 2014.
learning-based approaches.

[69] R. Tarjan, “Depth-first search and linear graph algorithms,” *SIAM J. Comput.*, vol. 1, no. 2, pp. 146–160, 1972.

[70] T. Stahl, A. Wischnewski, J. Betz, and M. Lienkamp, “Multilayer graph-based trajectory planning for race vehicles in dynamic scenarios,” in *Proc. IEEE Intell. Transp. Syst. Conf. (ITSC)*, Oct. 2019, pp. 3149–3154.

[71] P. E. Hart, N. J. Nilsson, and B. Raphael, “A formal basis for the heuristic determination of minimum cost paths,” *IEEE Trans. Syst. Cybern.*, vol. 4, no. 2, pp. 100–107, Jul. 1968.

[72] Z. Boroujeni, D. Goehring, F. Ulbrich, D. Neumann, and R. Rojas, “Flexible unit A-star trajectory planning for autonomous vehicles on structured road maps,” in *Proc. IEEE Int. Conf. Veh. Electron. Syst. (ICVES)*, Jun. 2017, pp. 7–12.

[73] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, “Practical search techniques in path planning for autonomous driving,” *Ann. Robot.*, vol. 1, no. 48105, pp. 18–80, 2008.

[74] S. Sedighi, D. Nguyen, and K. Kuhnert, “Guided hybrid A-star path planning algorithm for valet parking applications,” in *Proc. 5th Int. Conf. Control, Autom. Robot. (ICCAR)*, Apr. 2019, pp. 570–575.

[75] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, “A survey of motion planning and control techniques for self-driving urban vehicles,” *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–53, Mar. 2016.

[76] S. LaValle, “Rapidly-exploring random trees : A new tool for path planning,” *Dept. Comput. Sci.*, Iowa State Univ., Res. Rep. 9811, 1998.

[77] Y. Kuwata, G. Fiore, J. Teo, E. Frazzoli, and J. How, “Motion planning for urban driving using RRT,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2008, pp. 1681–1686.

[78] S. Karaman and E. Frazzoli, “Sampling-based algorithms for optimal motion planning,” *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, Jun. 2011, doi: 10.1177/0278364911406761.

[79] J. D. Gammell, S. S. Srivinasa, and T. D. Barfoot, “Informed RRT: Optimal sampling-based path planning focused via direct sampling of an admissible heuristic,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, Sep. 2014, pp. 2997–3004.

[80] H. Umari and S. Mukhopadhyay, “Autonomous robotic exploration based on multiple rapidly-exploring randomized trees,” in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Vancouver, BC, Canada, Sep. 2017, pp. 1396–1402.

[81] A. Bircher, M. Kamel, K. Alexis, H. Oleynikova, and R. Siegwart, “Receding horizon path planning for 3D exploration and surface inspection,” *Aut. Robots*, vol. 42, no. 2, pp. 291–306, Feb. 2018.

[82] D. J. Webb and J. van den Berg, “Kinodynamic RRT: Asymptotically optimal motion planning for robots with linear dynamics,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2013, pp. 5054–5061.

[83] Z. Kingston, M. Moll, and L. E. Kavraki, “Sampling-based methods for motion planning with constraints,” *Ann. Rev. Control, Robot.*, *Auto. Syst.*, vol. 1, no. 1, pp. 159–185, May 2018.

[84] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a Frenet frame,” in *Proc. IEEE Int. Conf. Robot. Autom.*, May 2010, pp. 987–993.

[85] S. Zhu and B. Aksun-Guvenc, “Trajectory planning of autonomous vehicles based on parameterized control optimization in dynamic on-road environments,” *J. Intel. Robot. Syst.*, vol. 100, nos. 3–4, pp. 1055–1067, Dec. 2020.

[86] L. Zheng, P. Zeng, W. Yang, Y. Li, and Z. Zhan, “Bézier curve-based trajectory planning for autonomous vehicles with collision avoidance,” *IET Intel. Transp. Syst.*, vol. 14, no. 13, pp. 1882–1891, Dec. 2020.

[87] S. D. Conte and C. De Boor, *Elementary Numerical Analysis: An Algorithmic Approach*. Philadelphia, PA, USA: SIAM, 2017.

[88] F. Perez-Cruz, “Kullback-leibler divergence estimation of continuous distributions,” in *Proc. IEEE Int. Symp. Inf. Theory*, Toronto, ON, Canada, Jul. 2008, pp. 1666–1670.

Daegyu Lee received the B.S. degree in automotive engineering from Kookmin University, Seoul, South Korea, in 2018, and the M.S. and Ph.D. degrees from the Division of Future Vehicle and Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2020 and 2024, respectively. He is currently a Researcher with the Electronics and Telecommunications Research Institute (ETRI), Daejeon. His research interests include autonomous systems and localization, focusing on robust and learning-based approaches.

Hyungi Seong (Graduate Student Member, IEEE) received the B.S. degree in mechanical engineering from Inha University, Incheon, South Korea, in 2019, and the M.S. degree from the Division of Robotics, Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2021. He is currently pursuing the Ph.D. degree in electrical engineering with KAIST. His research interests include decision-making, imitation learning, and reinforcement learning algorithms for autonomous driving and robotics.

Gyuree Kang received the B.S. degree in mechanical engineering from Sungkyunkwan University, Suwon, Gyeonggi, South Korea, in 2020, and the M.S. degree in robotics program from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2022, where she is currently pursuing the Ph.D. degree in electrical engineering.

Her research interests include autonomous systems, robotics, and task and motion planning.

Seunghil Han received the B.S. degree from the Division of Robotics, Kwangwoon University, Seoul, South Korea, in 2020, and the M.S. degree in robotics program from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2022. His research interests include learning-based motion planning, imitation learning, and reinforcement learning algorithms for autonomous driving and robotics.

D. Hyunchul Shim (Member, IEEE) received the B.S. and M.S. degrees in mechanical design and production engineering from Seoul National University, Seoul, South Korea, in 1991 and 1993, respectively, and the Ph.D. degree in mechanical engineering from the University of California at Berkeley, Berkeley, CA, USA, in 2000. He was with Hyundai Motor Company and Maxtor Corporation from 1993 to 1994 and from 2001 to 2005, respectively. In 2007, he joined the Department of Aerospace Engineering, KAIST, Daejeon, South Korea. He is currently a tenured Professor with the Department of Electrical Engineering and an Adjunct Professor with the Graduate School of AI, KAIST. He is also the Director of Korea Civil RPAS Research Center. His research interests include control systems, autonomous vehicles, and robotics.

Yoonjin Yoon received the B.S. degree in mathematics from Seoul National University, the dual M.S. degree in computer science and management science and engineering from Stanford University, and the Ph.D. degree in civil and environmental engineering from the University of California at Berkeley, Berkeley. She is currently an Associate Professor with the Civil and Environmental Engineering, Korea Advanced Institute of Science and Technology (KAIST), with joint appointment with the Graduate School Artificial Intelligence and the Graduate School of Data Science. Her main research interests include computational transportation science. Some of her earlier works has dealt with stochastic geometric air traffic flow optimization, topological urban airspace analysis, and urban energy optimization models. Her most recent efforts involve traffic forecast in urban mobility network using graph neural networks and urban predictions using graph representation learning.

Authorized licensed use limited to the terms of the applicable license agreement with IEEE. Restrictions apply.