A Hierarchical Motion Planning Framework for Autonomous Driving in Structured Highway Environments

DONGCHAN KIM, GIHOON KIM, HAYOUNG KIM, AND KUNSOO HUH (Member, IEEE)
Department of Automotive Engineering, Hanyang University, Seoul, 04763, South Korea
Corresponding author: Kunsoo Huh (khuh2@hanyang.ac.kr)
This work was supported by the Ministry of Trade, Industry, and Energy (MOTIE), South Korea, through the Technology Innovation Program (Industrial Strategic Technology Development Program, Development of Test Procedure Standards for V2I Connected Automated Driving Systems) under Grant 20014460.

ABSTRACT This paper presents an efficient hierarchical motion planning framework with a long planning horizon for autonomous driving in structured environments. A 3D motion planning with time information is a non-convex problem because there exists more than one local minimum point and various constraints such as roads and obstacles. Accordingly, to deal with high computational complexity and the problem of falling into a local minimum in an enormous solution space, a decoupled method is utilized, that consists of two steps: Long-term planning and short-term planning. First, the long-term planner provides reasonable far-sighted behavior through two processes. In the first process, a rough path that includes a driving strategy is generated in the 2D spatial space. Then, the jump point search algorithm is utilized with time information on the path to reduce the computational burden of A*, giving an acceptable quality of solution at the same time. In this step, a safe, comfortable, and dynamically feasible trajectory is generated. Next, the short-term planner optimizes a short-sighted trajectory using particle swarm optimization. In this method, a steering angle set is encoded as a particle, resulting in a safe, comfortable, and kinodynamically feasible trajectory. The proposed algorithm is implemented and evaluated in a range of vehicle-in-the-loop simulation scenarios, which include various virtual static and dynamic obstacles generated by Intelligent Driver Model. In the evaluation results, the proposed method reduced the computation time by up to 0.696 s with increasing the step cost by up to about 3%. The proposed algorithm is executed every 100 ms for a planning horizon of 10 seconds, and the average computation time is 31 ms with the worst-case computation time of 94 ms.

INDEX TERMS Motion planning, Dijkstra’s algorithm, Jump point search algorithm, Particle swarm optimization

I. INTRODUCTION
During the last few decades, research on autonomous driving has been actively studied worldwide, both in academia and the industry [1], [2] to aid drivers and, reduce tedious driving-related tasks. It also aims to prevent accidents caused by carelessness of drivers, which accounts for the majority of casualties [3], [4]. As a core module and being the decision stage in autonomous driving, the motion-planning module generates a legal, safe, comfortable, and kinodynamically feasible trajectory considering the surrounding environments and vehicle states. To date, many motion planning algorithms have been developed.
a nonlinear optimization problem is solved numerically. It is well-known that the SQP algorithm should have a good initial value for an acceptable computation time [7]. Chen et al. proposed the constrained iterative linear-quadratic regulator (CILQR) to efficiently solve the optimal control problem with nonlinear system dynamics and a general form of constraints [8]. The computational efficiency of CILQR was shown to be much higher than that of the SQP solver. However, without a good initial value, it can also influence the convergence speed of the algorithm.

For other approaches of direct methods, spatiotemporal state lattice-based trajectory planning methods were introduced to generate a feasible trajectory in a dynamic scenario [9], [10]. The choice of resolution of the search space is a trade-off between computation time and solution quality. If the resolution increases, the computation time increases. If the resolution decreases, the optimality of the solution space may be degraded, and a feasible solution cannot be found. Dorff et al. proposed a trajectory planning and control method using nonlinear model predictive controller (NMPC) [11]. It is validated in partially occluded parking environments with safe planning horizon of 1 s to 1.5 s. In addition, in [12] and [13], MPC-based trajectory planning methods were used. MPC solves a sequence of finite-time optimization problems in a recursive manner and generates consecutive control actions regarding the vehicle’s motion. A comfortable trajectory is generated while guaranteeing the safety of an autonomous vehicle. However, this method shows poor performance for non-convex and high-complexity problems.

Conversely, decoupled methods attempt to generate an optimal solution through multiple steps by reducing the dimensions of the state space. Werling et al. [14] generated longitudinal and lateral trajectories separately considering dynamic obstacles in the Frenet coordinate and then selected one optimal trajectory. Among the generated trajectories using quartic and quintic polynomials versus time, the jerk-minimizing trajectory is chosen. This method has the drawback of frequent swerving motion due to shortsighted planning. In [15] and [16], Li et al. performed a path-speed decomposition method to obtain an optimal trajectory step by step. First, a spatial path was generated using a curvature polynomial, and then a trapezoidal velocity profile was smoothed using a polynomial spline to obtain a trajectory. This method has the drawback of a short planning horizon and incompleteness of the solution owing to the finite set of motion primitives. In [17], a combination of dynamic programming and quadratic programming was proposed to generate path and speed profiles, respectively. A 3D station-lateral-speed problem is transformed into two 2D station-lateral and station-speed problems to reduce the complexity of the problem and computation time. Unfortunately, a limitation is that the generated trajectory is not guaranteed to be kinematically feasible. Xu et al. [18] used a method that consisted of two parts: Trajectory planning and trajectory optimization. First, a rough path and speed profile are generated, and then, they are iteratively optimized. However, non-holonomic constraints are not guaranteed to be satisfied. Zhang et al. proposed a method which adopts several steps [19]. In the first step, in a Frenet frame, a smooth driving guide line is obtained, then, optimization is performed for path generation. The second strategy is the proposition of piecewise-jerk path formulation. Finally, the optimization is performed to search a safe and kinematically feasible solution considering obstacles.

In addition, Li et al. proposed a semantic-level maneuver decision-making and trajectory planning method [20]. After the upper-level maneuver is decided, the lower-level trajectory planning is decoupled into longitudinal and lateral directions. Then, the selected trajectory is optimized by the operator splitting quadratic program (OSQP). This method is validated using Prescan simulator under two scenarios. Zhang et al. [21] introduced a hybrid trajectory planning method that generates a smooth, safe, and computationally efficient solution. However, only static obstacles were considered in their work. Lim et al. [22] used a hierarchical strategy in which a sampling-based approach was used for behavioral planning, and optimization was conducted for trajectory planning. However, the planner lacks the capability to generate trajectories with long horizons. Jin et al. [23] proposed a method to adaptively change the longitudinal horizons considering obstacles for better performance of on-road autonomous driving with avoidance of both static and moving obstacles. In decoupled methods, static obstacle is typically considered in the path planning stage and dynamic obstacle is considered in the speed planning stage. Thus, it is not guaranteed that the decoupled method is optimal. In addition, the number of tuning parameters is usually much more than the direct methods and their adjustment becomes difficult.

**B. CONTRIBUTION**

Nonetheless, a decoupled method was used for motion planning in our work to reduce the problem complexity and computational burden. Motion planning for self-driving can be divided into two parts [24], behavioral planning and trajectory planning. Behavioral planning can be categorized as high-level decision-making and is responsible for long-term planning (\( t_{\text{horizon}} > 5s \)) in complex situations [25]. Many studies have been conducted on state machines when performing long-term planning [26]–[30]. Various behaviors are represented as predefined states, and then an appropriate behavior is selected at every step. However, when the situation becomes complicated and the maneuvers to model are increased, it becomes computationally intractable owing to the exponentially growing number of state transition rules. Therefore, to address this problem, Hubmann et al. [25] proposed a generic long-term planning method using the A* graph search algorithm. However, only longitudinal planning was regarded in their work.

In this paper, a hierarchical framework for motion planning in structured environments is proposed that provides a long-term solution that considers both the longitudinal and
lateral directions. The algorithm structure is divided into two steps, long-term planning and short-term planning. Long-term planning is performed first and consists of two processes. In the first process, an optimal spatial path is obtained by considering the road geometry and obstacles. This part determines which space to drive is desirable in a complex situation. In the next process, a spatiotemporal trajectory is generated by placing the position and velocity profiles versus time on the previously obtained long-term path, which has a reduced searching space. This process provides a legal, safe, comfortable, and dynamically feasible long-term maneuver. In the second step, short-term planning is performed using the front part of the long-term solution. This step provides a safe, comfortable, and kinodynamically feasible trajectory in a combined resampling and optimization manner. Finally, the performance of the proposed algorithm is verified through a vehicle-in-the-loop simulation (VILS) with a real car under various scenarios. The contributions of this study can be summarized as follows:

- The proposed hierarchical scheme can generate safe, comfortable, and kinodynamically feasible trajectories that can deal with static and dynamic obstacles.
- Jump point search (JPS) algorithm which has been used only for the problems with a uniform step cost, is utilized in our problem which has a non-uniform step cost with the help of a carefully designed assumption. Leveraging JPS algorithm reduces the computation time significantly and allows real-time long-term planning with an acceptable solution quality.
- The short-term trajectory planner optimizes the initial trajectory from the long-term planner. Furthermore, steering angle sets are encoded as particles in particle swarm optimization (PSO) through a kinematic vehicle model that satisfies the non-holonomic constraint. Additionally, in PSO, the driving comfort, safety, and dynamic constraints are considered as well.
- The feasibility of the proposed method was demonstrated in various automated driving scenarios using VILS.

C. PAPER ORGANIZATION

The remainder of this paper is organized as follows: The overall algorithm framework is introduced in Section II. The optimal long-term path planning using the shortest-path algorithm and long-term trajectory planning using a JPS algorithm are described in Section III. In Section IV, short-term trajectory optimization using PSO algorithm is explained. Section V explains the experimental setup and evaluation of the proposed algorithm in a vehicle-in-the-loop environment. Section VI summarizes the proposed algorithm and discusses the future work.

II. ALGORITHM FRAMEWORK

To generate a trajectory in an autonomous driving situation that is non-convex [31], the autonomous vehicle must consider complex driving environments which comprise static and dynamic surrounding vehicles, roads, traffic rules, etc. Based on the scene information, an autonomous vehicle must run within the normal speed range, change lanes, and drive safely without colliding with surrounding vehicles. Therefore, when a trajectory is generated, safety, comfort, and limits of the vehicle kinematics and dynamics should be considered. The overall algorithm framework for optimal motion planning is shown in Fig. 1. This study focuses on a hierarchical motion planning framework for autonomous driving on unidirectional roads. The information of the surrounding vehicle states from the perception module is assumed to be available to the planning module.

In the first step, the long-term (~10s) spatial path including the longitudinal and lateral motions, is obtained by considering the surrounding vehicles. A maneuver with a lane-level lateral motion target is generated by considering the current state and prediction during a specific time horizon of the surrounding environments.

In the second step, the position and velocity profiles are applied to the optimal spatial path from the first step to obtain a long-term spatiotemporal trajectory. Safe motion planning is carried out in which a collision-free trajectory is generated considering static and dynamic obstacles. The improved A* algorithm, JPS, was utilized as the base algorithm to reduce the computational effort [32]. The original JPS is optimized for solving the grid-based problem, which has a uniform cost for each action. In our problem, which is without a uniform cost for each action, a sub-optimality condition is defined and the JPS is applied.

Finally, short-term trajectory optimization was performed using the front-part of the optimal long-term trajectory. It is of practical importance to follow the adjacent short-term (~3s) trajectory when tracking the control of the generated trajectory. In our work, the PSO algorithm was utilized to make the short-term trajectory kinodynamically feasible, smooth, and safe.

III. LONG-TERM PLANNING

In this section, the concept of long-term planning of the proposed hierarchical framework is explained in detail. First, the optimal spatial path generation is described, then the
spatiotemporal trajectory using the optimal spatial path is described.

A. OPTIMAL SPATIAL PATH GENERATION

To create an optimal spatial path, a search space is created that comprise lane-level nodes. An optimal path selection process was performed on the search space. Each node is represented by the following 3D space as state n.

$$ N = \left\{ n \in [s, d, l]^T | s \in \mathbb{R}, d \in \mathbb{R}, l \in [1, N_l] \right\} $$  \hspace{1cm} (1)

where \( s \) and \( d \) are defined as the longitudinal and lateral positions along the road, respectively, \( l \) is defined as the rightmost lane having a value of 1 and the leftmost lane having a value of \( N_l \). The road geometry was defined according to the Frenet coordinates, as described in Fig. 2. The search space consists of a total of \( N_p \times N_l \) nodes in the Frenet coordinates. In addition, except for the initial node \( n_0 \), the \( k \)th step longitudinal position \( s_k \) of node \( n_k \) is expressed as follows:

$$ \Delta s = v_0 \Delta t_l $$  \hspace{1cm} (2)

$$ s_k = s_0 + k \Delta s $$  \hspace{1cm} (3)

where \( v_0 \) and \( s_0 \) are the initial velocity and longitudinal position of the ego vehicle, respectively, and \( \Delta t_l \) is the time interval. \( \Delta s \) is determined by the current speed of the ego vehicle. However, to prevent the search space from disappearing under low speeds, \( \Delta s \) is set to be greater than \( \Delta s_{\min} \), which is set to 10 m in our work.

Fig. 3 shows an example of the optimal spatial path generation. The blue vehicle is the ego autonomous vehicle and the white vehicle represents the obstacle vehicle during the prediction horizon. The nodes were placed in the direction of the road with an interval of \( \Delta s \). Each node is generated at every layer, and the result is defined as a directed, acyclic graph. The edge is a linear path from a node in one layer to a node in the next layer. The cost of edge \( e_{n_k \rightarrow n_{k+1}} \) is shown in the following equation, where \( n_k \) is a node in layer \( L_k \) and \( n_{k+1} \) is a node in layer \( L_{k+1} \).

$$ J(e_{n_k \rightarrow n_{k+1}}) = w_{dist} \text{dist}(e_{n_k \rightarrow n_{k+1}}) + w_c c(e_{n_k \rightarrow n_{k+1}}) + w_{col} \text{col}(e_{n_k \rightarrow n_{k+1}}) $$  \hspace{1cm} (4)

where \( w_{dist}, w_c, w_{col} \) are weights for each index.

The first cost term represents the length of each edge using the Euclidean distance between two nodes, and it prevents the ego vehicle from changing lanes frequently by assigning a penalty to a long edge, as shown in (5).

$$ \text{dist}(e_{n_k \rightarrow n_{k+1}}) = \sqrt{(n_k(1, 2) - n_{k+1}(1, 2))^2} $$  \hspace{1cm} (5)

The second cost term is for the consistency of consecutive plans. This term reduces the difference in the lateral offset between the current path candidate and the previous optimal spatial path, as described in (6):

$$ c(e_{n_k \rightarrow n_{k+1}}) = |d_{k+1}^{pre} - d_{k+1}| $$  \hspace{1cm} (6)

where \( d_{k+1}^{pre} \) and \( d_{k+1} \) are the lateral positions of the previously selected optimal path and a new path candidate, respectively.

The velocity profile of the previous planning cycle was used to calculate the predicted position of the ego vehicle. It is assumed that using the velocity profile of the previous planning cycle is reasonable because the planning frequency is sufficiently high. The last collision cost term suppresses the collision between the predicted ego vehicle and the surrounding vehicles predicted with the constant velocity (CV) model and is defined as

$$ \text{col}(e_{n_k \rightarrow n_{k+1}}) = \begin{cases} 
  c_{col}, & \text{if } |s_{e,i} - s_{O,j}^i| < s_{sf} \text{ and } l_i = l_{O,i}^i \\
  0, & \text{otherwise}
\end{cases} $$  \hspace{1cm} (7)

with \( s_{sf} = \frac{L_O}{2} + \frac{L_{ego}}{2} + s_{\text{margin}} \).

\( \frac{L_O}{2} + \frac{L_{ego}}{2} + s_{\text{margin}} \).  \hspace{1cm} (8)

Here, subscript \( j \) denotes a number from 1 to \( N_o \) where \( N_o \) is the number of obstacles in the region of interest. For each \( j \) from \( i = k_s \) to \( i = k_e \), it is checked if the condition in (7) is satisfied. If the condition is satisfied even once, cost \( c_{col} \) is imposed. Furthermore, \( k_s \) and \( k_e \) represent the start and end indices of the s-axis position, where the predicted ego vehicle is within the edge \( e_{n_k \rightarrow n_{k+1}} \), respectively. Additionally, \( s_{e,i} \) and \( s_{O,j}^i \) are the s-axis positions of the predicted ego vehicle and \( j \)th obstacle \( O_{i}^j \) at the \( i \)th step, respectively. Further, \( l_i \)
and $t^{O_i}$ are the lane values of nodes $n_i$ and $j^{th}$ obstacle, $O^i_{j}$ at the $i^{th}$ step, respectively. A constant value $c_{col}$ is set sufficiently large so as to make it dominate the overall cost function for penalizing collisions. The lengths of the obstacle and ego vehicle are $L_O$ and $L_{ego}$, respectively. The margin distance value is $s_{margin}$, and $s_{sf}$ is the safety distance to restrict collisions between vehicles.

This becomes a shortest-path problem, in which dynamic programming is utilized to solve it quickly. The shortest-path problem is an algorithm for finding the shortest path between nodes in a graph. In our problem, one node per layer is selected to construct the path using the cost function in (4). The optimal solution with minimum total cost is given by a sequence of nodes on the $N_p$ layers covering the space that satisfies:

$$\arg\min_{\{e_{nk}\rightarrow n_{k+1}\}} \sum_{k=0}^{N_p-1} J(e_{nk} \rightarrow n_{k+1}).$$  \hspace{1cm} (9)

**B. SPATIOTEMPORAL TRAJECTORY GENERATION PROBLEM FORMULATION**

A spatial path planner provides the desired maneuver to a spatiotemporal trajectory planner considering the long-term objective of autonomous driving in the spatial search space. An optimization problem is formulated to generate a trajectory based on the optimal spatial path given, and is represented as:

$$\min J = c_{tot} = \sum_{k=0}^{N_p-1} c(x_k, a_k, x_{k+1}, O_{ol})$$

s.t. $x_{k+1} = A \begin{bmatrix} x_k \\ 1 \end{bmatrix} + B a_k$

$$c(\cdot) = w_{vc} c_{vc}(x_{k+1}) + w_{AC} A(x_k, x_{k+1}, O_{ol})$$

$O_{ol} = \left\{ \left[ t^{O}_{ten}, s^{O}_{ol,s,i}, t^{O}_{ol,c,i}, s^{O}_{ol,c,i}, t^{O}_{ol,c,i} \right]_{i=1\cdots m} \right\}$

$O = \left\{ O_k | k = 0 \cdots N_p \right\}$

The detailed description is contained in Sections III-B1 to III-B3. State $x$ in the state space $\mathcal{X} \subseteq \mathbb{R}^3$ is represented as follows:

$$\mathcal{X} = \left\{ x = [s, v, t]^T | s \in \mathbb{R}^+, v \in \mathbb{R}^+, t \in \mathbb{R}^+ \right\}$$  \hspace{1cm} (11)

where the state $x$ includes the longitudinal position $s$, velocity $v$ and time $t$.

The problem is formulated as a discrete planning problem [33] with a similar approach as in [25] and is solved using the JPS algorithm, which will be explained in detail in Section III-C.

1) Transition Model

The state $x_k$ represents a state in the planning step $k$, and the state transition model in discretized form is expressed as follows:

$$x_{k+1} = \begin{bmatrix} s_{k+1} \\ v_{k+1} \\ t_{k+1} \end{bmatrix} = \begin{bmatrix} 1 & \Delta t_{lt} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t_{lt} \end{bmatrix} \begin{bmatrix} s_k \\ v_k \\ t_k \\ 1 \end{bmatrix} + \begin{bmatrix} 0.5(\Delta t_{lt})^2 \\ \Delta t_{lt} \\ 0 \end{bmatrix} a_k$$  \hspace{1cm} (12)

where $a_k$ is an acceleration candidate belonging to a discretized acceleration set $A$. The state is expanded using the state transition model for a limited time $T_{lt}$ with an interval of $\Delta t_{lt}$.

2) Representation of the Environment

The autonomous vehicle encounters a static and dynamic obstacle $O$ on the road, the information of a specific obstacle at the $k^{th}$ step is parameterized as a vector, $\left[ t^{O}_{ten}, s^{O}_{ol,s,i}, d^{O}_{ol,k} \right]^T$ where $t^{O}_{ten}$ is the length of the obstacle, $s^{O}_{ol}$ is the longitudinal position, and $d^{O}_{ol}$ is the lateral position.

Next, a procedure is performed to extract information about the obstacles that affect the ego vehicle. The long-term optimal spatial path obtained is checked to determine the extent of overlap with the prediction of surrounding vehicles. The overlapped obstacle $O_{ol}$ is represented as follows using $O$ which includes the accumulated information of $O_k$ during the prediction horizon:

$$O_{ol} = \left\{ \left[ t^{O}_{ten}, s^{O}_{ol,s,i}, t^{O}_{ol,c,i}, t^{O}_{ol,c,i} \right]_{i=1\cdots m} \right\}$$  \hspace{1cm} (13)

$$O = \left\{ O_k | k = 0 \cdots N_p \right\}$$  \hspace{1cm} (14)

where $s^{O}_{ol,s,i}$ and $t^{O}_{ol,c,i}$ represent the start and end positions in the longitudinal direction of the optimal spatial path at the $i^{th}$ overlap with the specific obstacle, respectively. Furthermore, $t^{O}_{ol,s,i}$ and $t^{O}_{ol,c,i}$ represent the time information at the corresponding step, respectively.

Fig. 4 shows an example in which one obstacle is overlapped twice on the optimal spatial path. In Fig. 4(a), the predicted ego vehicle (light blue color) through the optimal spatial path (blue solid line) represents the position information for the $N_p$ step calculated from the velocity profile of the previous planning cycle.

In Fig. 4(b), the overlapped information is expressed as $\left\{ \left[ t^{O}_{ten}, s^{O}_{ol,s,i}, t^{O}_{ol,c,i}, t^{O}_{ol,c,i} \right]_{i=1,2} \right\}$ using (13) where $m = 2$. It is considered overlapped if the prediction of the surrounding vehicle is in the same lane as the optimal spatial path, and the difference in the longitudinal position between the obstacle and the ego vehicle is within $s_{sf}$. Assuming highway driving, the variation in $s_{sf}$ owing to the heading angle is ignored. The overlapped path is represented by the red line in Fig. 4(b).
At \( k = 1 \) and \( k = 2 \), the ego vehicle and obstacle overlap. Therefore, it is assumed that the position of \( s \) between the two points continues to overlap. Next, at \( k = 3 \), the optimal spatial path moves to the left lane and does not overlap with the obstacle at the same time step. Therefore, the first overlap section is set as: \([s_{ol,s,1}, s_{ol,e,1}]\).

Next, at \( k = N_p - 1 \), the optimal spatial path and obstacle do not overlap, whereas at \( k = N_p \), they overlap. In this case, utilizing the \( s \) positions of the ego vehicle, \( s \) position of the obstacle and \( s_{sf} \), the second overlap section is set as \([s_{ol,s,2}, s_{ol,e,2}]\). Furthermore, \( t_{ol,s,1}, t_{ol,e,1}, t_{ol,s,2}, t_{ol,e,2} \) represent time information corresponding to each \( s \) overlap value.

3) Cost Function
If an action \( a_k \) is taken to state \( x_k \), it moves to the next state \( x_{k+1} \) through the transition model. The step cost \( c(x_k, a_k, x_{k+1}, O_{ol}) \) is defined for each path, and the overall cost \( c_{tot} \) which is summed along the path is represented as follows [33]:

\[
c_{tot} = \sum_{k=0}^{N_p-1} c(x_k, a_k, x_{k+1}, O_{ol}) \tag{15}
\]

The path with a minimal cost from the initial state to the goal state is obtained using the graph search algorithm. The step cost is defined as

\[
c(x_k, a_k, x_{k+1}, O_{ol}) = w_c c_v(x_{k+1}) + w_A c_A(a_k) + c_O(x_k, x_{k+1}, O_{ol}) \tag{16}
\]

where \( w_c \) and \( w_A \) are weights for each index.

The first cost term represents the deviation from the reference velocity \( v_{ref} \) and is defined as follows:

\[
c_v(x_{k+1}) = \left( \frac{v_{k+1} - v_{ref}(s_{k+1})}{v_{ref}(s_{k+1})} \right)^2. \tag{17}
\]

The limit velocity \( v_{limit} = \sqrt{\frac{d_{max}}{\kappa}} \) is determined by the maximum lateral acceleration \( d_{max} \) and road curvature \( \kappa \). Furthermore, there exists a road velocity limit \( v_{road} \). \( v_{ref} \) is decided to be the minimum value of these two velocities:

\[
v_{ref} = \min(v_{limit}, v_{road}). \tag{18}
\]

The second cost term penalizes the size of the action that belongs to acceleration set \( A \). It increases the driver’s comfort by providing cost in the form of squared acceleration.

The last cost term represents the cost of obstacles. An obstacle has an area on the distance-time plane in state space \( X \). The cost for the static and dynamic obstacles is expressed as follows using the overlapped obstacle \( O_{ol} \):

\[
c_O(x_k, x_{k+1}, O_{ol}) = c_{O_{ol}}(x_k, x_{k+1}), \tag{19}
\]

with \( c_{O_{ol}}(x_k, x_{k+1}) = \begin{cases} \inf_j & \text{if } \{x_k x_{k+1}\} \cap O_{ol} \neq \emptyset \\ 0 & \text{otherwise} \end{cases} \tag{20} \]

where \( x_k x_{k+1} \) is defined as the set of points on the vector between \( x_k \) and \( x_{k+1} \).
Algorithm 1 Function JPS

Require: \( x_s \) : start state, \( g \) : goal state, \( h(x) \) : heuristic, \( O \) : obstacle
1: \( \mathcal{C} \leftarrow \emptyset \)
2: \( N = \{ x_s, 0, 0, 0, \emptyset \} \)
3: \( \mathcal{O} \leftarrow N \)
4: \( \text{KeepSearching} \leftarrow 1 \)
5: while \( \text{KeepSearching} \) is 1 do
6: \( \text{if} \ \text{is empty then} \)
7: \( x_{\text{new}} = \text{transition}(N, x, -a_{\text{max}}) \)
8: \( N_{\text{new}} = \{ x_{\text{new}}, c_{\text{new}}, h(x_{\text{new}}), -a_{\text{max}}, N \} \)
9: \( \text{Add} \ N_{\text{new}} \to \mathcal{O} \)
10: \( \text{else} \)
11: \( N = \text{Extract} \) node with minimum \( c + h \) from \( \mathcal{O} \)
12: \( \text{Add} \ N \) to \( \mathcal{C} \) and \( \text{Delete} \) from \( \mathcal{O} \)
13: \( \text{if} \ N \) is \( g \) then
14: \( \text{trajectory} = \text{Backtracking}(N) \)
15: \( \text{KeepSearching} \leftarrow 0 \)
16: \( \text{else} \)
17: \( \text{foreach} \ a \in \text{Available action}(N) \) do
18: \( \{ x_{\text{new}}, c_{\text{new}} \} \leftarrow \text{jump}(N, x, a, g, O, N, c) \)
19: \( N_{\text{new}} = \{ x_{\text{new}}, c_{\text{new}}, h(x_{\text{new}}), a, N \} \)
20: \( \text{if} \ N_{\text{new}} \in \mathcal{C} \) then
21: \( \text{continue} \)
22: \( \text{else if} \ N_{\text{new}} \) has the same state with
23: \( n \in \mathcal{O} \) with smaller cost then
24: \( \text{Replace} \ n \) with \( N_{\text{new}} \)
25: \( \text{else} \)
26: \( \text{Add} \ N_{\text{new}} \to \mathcal{O} \)
27: \( \text{end if} \)
28: \( \text{end if} \)
29: \( \text{end if} \)
30: \( \text{end if} \)
31: \( \text{return} \ \text{trajectory} \)

Algorithm 2 Function jump

Require: \( x_k \) : initial state, \( a_k \) : action, \( g \) : goal state, \( O \) : Obstacle, \( C_{\text{cum}} \) : cumulative cost
1: \( x_{k+1} \leftarrow \text{transition}(x_k, a_k) \)
2: \( \text{if} \ x_{k+1} \) is inside an obstacle then \( \triangleright \) collision
3: \( \text{return} \ x_{k+1} \leftarrow (x_k, a_k, x_{k+1}, O) + C_{\text{cum}} \)
4: \( \text{else if} \ x_{k+1} = g \) then \( \triangleright \) goal state
5: \( \text{return} \ x_{k+1} \leftarrow (x_k, a_k, x_{k+1}, O) + C_{\text{cum}} \)
6: \( \text{else if} \ t_k \) is near end of \( t_{\text{ol,c,i}} \) for any \( o \in O \) then
7: \( \triangleright \) forced neighbor
8: \( \text{return} \ x_{k+1} \leftarrow (x_k, a_k, x_{k+1}, O) + C_{\text{cum}} \)
9: \( \text{else if} \ a \neq 0 \) then
10: \( \text{for} \ a = 0 \) do
11: \( \text{if} \ \text{jump}(x_{k+1}, a_k, g) \) has finite cost then
12: \( \text{return} \ x_{k+1} \leftarrow (x_k, a_k, x_{k+1}, O) + C_{\text{cum}} \)
13: \( \text{end if} \)
14: \( \text{end for} \)
15: \( \text{else} \)
16: \( \text{return} \ \text{jump}(x_{k+1}, a_k, g), c(x_k, a_k, x_{k+1}, O) + C_{\text{cum}} \)
17: \( \text{end if} \)

C. SPATIOTEMPORAL TRAJECTORY GENERATION PROBLEM SOLVING

Many search algorithms exist to solve planning problems. Dijkstra, A*, and JPS are widely used in discrete planning problems. The JPS algorithm [32] reduces the computation time of A* by adding only the jump points to the open list. The optimality of JPS is guaranteed only when the step cost is uniform, which is not the case in our study. Nonetheless, to take advantage of JPS in terms of computational efficiency and to carry out long-term planning in real time, the condition for the sub-optimality of the solution is defined as follows:

- In the time domain where obstacles overlap within the planning horizon of the ego vehicle, if \( a = 0 \) is taken once as the minimum action, only that action can be taken for the rest of the time.

A suboptimal solution is equivalent to the optimal solution if there are no other vehicles on the path, and it further, shows different movements if there are obstacles within the planning horizon of the ego vehicle. Let us assume that both the ego vehicle and the preceding vehicle on the path have a lower velocity than the reference velocity. In this case, as shown in Fig. 5(a), the optimal solution when \( w_A \) is low as in our case is as follows: first to accelerate to reach the target velocity of 20 m/s in the example, and then to decelerate near the obstacle. In contrast, as shown in Fig. 5(b), the suboptimal solution is to maintain \( a = 0 \) after acceleration. If it collides with the obstacle, as shown in Fig. 5(b), the node lastly accelerated is not added to the open list. Thereafter, the trajectory that maintains a constant speed is chosen as the suboptimal solution.

To an original optimal problem, JPS cannot be utilized because the minimum action that has a minimum step cost is not consistent and changes according to the state and action. However, to solve a suboptimal problem, as in our case, JPS can be utilized because maintaining a constant speed at last is a solution, and the constant speed \( (a = 0) \) can be interpreted as the minimum action.

FIGURE 6: Forced neighbor. (a) Grid world (b) our problem.
Searching begins with the addition of the current state and zero cost to the open list. The state \((x_k)\) with the lowest cost is excluded from the open list, and added to the closed list. If \(x_k\) is the goal state, then the search ends. If not, the jump point is searched within the action set which is modified to prevent the velocity of the next state from being lower than zero. If the searched state \((x_{k+1})\) is in the open list, but the sum of the step cost and heuristic is lower than that of the state in the list, the information of the corresponding state is updated. If the state \((x_{k+1})\) is not in the open list, it is added to the open list. This process is repeated for the state in the open list, and if the selected state is the goal, the search is finished and the optimal profile is constructed by backtracking via the closed list. If the open list is empty as no state candidate exists, emergency braking with \(a_{max} = 8\) is conducted for safety, as added in Lines 7 to 9. The pseudocode is described in Algorithm 1, where \(N\) represents a node consisting of \{state, action, path cost, heuristic, parent node\}. Furthermore, \(\Phi\) and \(\mathcal{C}\) indicate the open list and closed list, respectively.

In Line 18 of Algorithm 1, the JPS algorithm is performed as described in Algorithm 2, which represents a recursive function that makes a jump between the nodes. The jump step continues until the stop conditions are met [25], which consist of a collision, goal state, and forced neighbor as described in Lines 2 to 7 of the algorithm. In Line 1, the transition model described in (12) is utilized to obtain the next state, \(x_{k+1}\), using the current state and action. Line 2 indicates the collision situation, Line 4 is when the goal state is reached and Line 6 is when the forced neighbor is encountered. The forced neighbor is represented as \(x_{k+1}\) in Fig. 6, which is the state blocked by obstacles and must be passed through \(x_{k}\) [32]. Lines 8 to 11 proceed when the first step’s action is not the minimum action. In this case, the jump function recurses with the minimum action and checks if it collides with obstacles. If no collision occurs, the cost has a finite value, and the next state \(x_{k+1}\) is returned. Otherwise, the jump is performed again with the same step action, \(a_k\). The search heuristic used in this study is represented as follows:

\[
h = w_h \sum_{k=t}^{T_{lt}-1} \left( \frac{v_{k+1} - v_{ref}(s_{k+1})}{v_{ref}(s_{k+1})} \right)^2, \quad \text{where} \quad (21)\]

\[
v_{k+1} = v_k + \Delta t_l \arg\min_{a \in A} (v_{ref}(s_{k+1}) - (v_k + a \Delta t_l)). \quad (22)\]

In (22), the action that minimizes the difference between the reference velocity and the state’s velocity at \(k + 1\) is calculated. Then, the selected action is applied to obtain \(v_{k+1}\), which is utilized in (21). This process is repeated from \(k = t\) until \(k = T_{lt} - 1\). The heuristic is admissible in (21) and has exactly the same form as the cost \(c_v\) in the step cost in (16), and it is guaranteed to have a lower value than the minimal cost from \(k = t\) to \(k = T_{lt} - 1\) which is expressed as \(\sum_{k=t}^{T_{lt}-1} c^* (x_k, a_k, x_{k+1}, O_{ol})\). In addition, as long as the heuristic is the lower bound on the actual cost and is consistent, the A* algorithm is complete. Consistency is fulfilled if \(h(x_k) \leq c(x_k, a_k, x_{k+1}, O_{ol}) + h(x_{k+1})\). Thus, an admissible and consistent heuristic was used in this study.

Next, to verify the performance of the solution using the JPS, three metrics which comprise the number of nodes in the open list, computation time, and step cost are evaluated under the test scenarios generated in Section V. In Table 1, the metrics are compared with the JPS for the test scenarios. The average value was used to evaluate the first two metrics, and the ratio to the JPS was utilized for the last metric. For the number of nodes in the open list, it is significantly reduced for all scenarios. Accordingly, the computation time was also greatly reduced for each test scenario. Finally, in the worst-case scenario related to the step cost, the optimal solution from A* reaches 0.97 times that of the solution from the JPS, which is acceptable.

### IV. SHORT-TERM PLANNING

In this section, the short-term planning of the proposed hierarchical framework is explained. First, to increase the resolution, resampling of the spatiotemporal trajectory obtained from long-term planning is performed. Then, the part that optimizes the trajectory by encoding a steering angle set as a particle is explained using the resampled trajectory as the reference position value in the PSO.

#### A. RESAMPLING OVER TRAJECTORY

The spatiotemporal trajectory obtained from the long-term planning is a behavioral strategy with a long time horizon. However, tracking a short-term trajectory is important for the control module, therefore, the front part of the entire long-term trajectory was used for the resampling step. The results
tiant trajectory has sufficient resolution for considering the continuous changes of the vehicle state. Linear interpolation between discrete nodes is used for the resampling step. The planning horizon of $T_{st}$ is discretized by $\Delta t_{st}$ resulting in the $N_{pso}$ number of sample nodes.

**B. TRAJECTORY OPTIMIZATION**

PSO is an algorithm that was inspired by the flocking behavior of swarms [34]. It is well-known that PSO performs well when solving nonlinear optimization problems and has the advantages of a fast convergence speed with only a small number of parameters that need to be adjusted [35]. In addition, it is a very efficient global search algorithm and derivative free. Furthermore, although there is no guarantee that PSO provides the global optimal solution, it is less likely to converge to a local minimum and finds good quality solution with reasonable computational effort. For this reason, many studies have used the PSO for path planning [35]–[38].

Each particle $p^i_t$ at time $t$ in the swarm indicates a possible solution in the multidimensional steering search space. A particle consists of a total of $N_{pso}$ steering angle values and becomes an $N_{pso}$-dimensional vector. The fitness of the particle is calculated through a designed fitness function, which is described in detail in Section IV-C. Each particle moves with a speed vector $v^i_t$, and the trajectory is adjusted accordingly. In addition, the personal best solution $p^i_{pbest}$ and swarm’s best solution $p_{gbest}$ are updated respectively. Each particle is updated as follows:

$$p_{t+1} = v_{t+1} + p_t$$ (23)

Then, the velocity of each particle is updated considering the current velocity, the personal best solution $x_{pbest}$, and the global (the swarm’s) best solution $x_{gbest}$, as follows:

$$u_{t+1} = w_i u_t + w_p r_1 (p_{pbest} - p_t) + w_g r_2 (p_{gbest} - p_t)$$ (24)

where $w_i$ is the inertia weight damping ratio, $w_p$ and $w_g$ are the personal learning coefficient and global learning co-

The optimization phase is terminated when the given optimization criteria are met, or if the maximum iteration limit is reached.

Fig. 7 shows an example of a steering angle set encoding as a particle of the PSO algorithm, and the $i^{th}$ particle $p^i_t$, is expressed as shown in (25).

$$p^i_t = [\delta_0, \delta_1, \ldots, \delta_{N_{pso}-1}]^T$$ (25)

A particle consists of a steering angle set of the ego vehicle during horizon $N_{pso}$. A particle is passed through the kinematic [39], [40] that satisfies the non-holonomic constraint to obtain position information $s_k, d_k$, and heading angle $\theta_k$ as follows:

$$s_k = \Delta t_{st} \cdot v_{k-1} \cdot \cos (\theta_{k-1}) + s_{k-1}$$ (26)

$$d_k = \Delta t_{st} \cdot v_{k-1} \cdot \sin (\theta_{k-1}) + d_{k-1}$$ (27)

$$\theta_k = \Delta t_{st} \cdot \frac{v_{k-1}}{L} \cdot \tan (\delta_{k-1}) + \theta_{k-1}$$ (28)

where L is the length of the ego vehicle, and the subscript $k$ denotes a number from 1 to $N_{pso}$. The obtained position set ensures that the curvature of the trajectory is within the kinematic constraints.

**C. FITNESS FUNCTION**

In the PSO, particles move in the direction of minimizing the fitness function to find the optimal value. In this study, the fitness function is composed of six terms and is expressed by the following equation.

$$f_{pso} = w_{ref}f_{ref} + w_{as}f_{as} + w_{ad}f_{ad} + w_{js}f_{js} + w_{jd}f_{jd} + f_{col}$$ (29)

where $w_{ref}, w_{as}, w_{ad}, w_{js}, w_{jd}$ and $w_{col}$ are the weights for each index except for the last collision cost.

The first fitness term represents a reference cost and is defined as

![Figure 8](image)
where $s_{i,ref}$ and $d_{i,ref}$ are the reference nodes for the longitudinal and lateral directions from the resampled behavioral trajectory. This fitness term makes a particle vector from the PSO that does not deviate significantly from the generated trajectory through the node and edge of the road geometry.

The next four fitness terms indicate the acceleration and jerk values were calculated using

$$f_{as} = \sum_{i=0}^{N_{pso} - 2} a_{s,i}^2$$

$$f_{ad} = \sum_{i=0}^{N_{pso} - 2} a_{d,i}^2$$

$$f_{js} = \sum_{i=0}^{N_{pso} - 3} \frac{(-3s_{i+3} + 3s_{i+2} - 3s_{i+1} + s_i)^2}{\Delta t_{st}^2}$$

$$f_{jd} = \sum_{i=0}^{N_{pso} - 3} \frac{(-3d_{i+3} + 3d_{i+2} - 3d_{i+1} + d_i)^2}{\Delta t_{st}^3}$$

The acceleration and jerk values were calculated using forward finite differences. The physical limits of the vehicle dynamics for safe driving are considered using a circle of forces [41]. The acceleration values used when calculating the acceleration costs $f_{as}$ and $f_{ad}$ were constrained to be within the maximum acceleration:

$$a_{s,i}^2 + a_{d,i}^2 \leq a_{max}^2.$$  

If the condition in (35) is violated, the $i^{th}$ cost of $f_a$ is set to be infinite to prevent the particle from being selected. The restriction check is not performed in Cartesian coordinates but rather in the Frenet space.

The last fitness term is related to collisions. The vehicle shape is represented using two circles for collision checks, similar as in [6]: wherein the distance between the centers of each circle of obstacles and the ego vehicle is required to be greater than the sum of each circle’s radius. When the requirements are violated, $f_{col}$ becomes infinite. In Fig. 8(a), the resampled trajectory is shown, and in Fig. 8(b), the initial trajectory is optimized using the PSO algorithm, which generates a kinodynamically feasible, smooth, and safe trajectory.

### V. VEHICLE-IN-THE-LOOP SIMULATION

The VILS [42], [43] is implemented on the TS KATRI PG track, as shown in Fig. 9 with virtual scenarios and a test vehicle equipped with GPS, wherein it is assumed that information regarding virtual surrounding obstacles’ positions and velocities is perfectly known. The proposed hierarchical motion planning framework is implemented in C++ language on a PC running Ubuntu 16.04 equipped with a quad-core Intel Core i7-6700K CPU. The execution period of the algorithm was 100ms. For validation, five highway scenarios in a unidirectional road were generated. Our first four scenarios included static and dynamic obstacles modelled by intelligent driver model (IDM). The last scenario included an obstacle performing a sudden cut-in maneuver. These scenarios were selected to show that our algorithm can perform many functionalities such as lane changing, car-following, obstacle avoidance, and stopping in various speed ranges. The motion planner parameters for implementation are listed in Table 2.

### A. TRAFFIC GENERATION

The obstacle vehicles are controlled by an IDM. To imitate human drivers who might be aggressive or defensive, both types of vehicles are included in the environment for the first two scenarios. Only one type of vehicle was considered for test scenarios 3 and 4. The dynamics of the vehicle are

![Diagram](image_url)

**FIGURE 9:** Test area: TS KATRI PG (Proving Ground).

| Parameter | Value | Unit | Parameter | Value | Unit |
|-----------|-------|------|-----------|-------|------|
| $N_t$     | 2     | -    | $\Delta t_{st}$ | 0.1   | s    |
| $T_{lt}$  | 10    | s    | $N_{pso}$   | 30    |      |
| $\Delta t_{lt}$ | 1    | s    | $w_i$     | 0.7   |      |
| $N_p$     | 10    | -    | $w_p$     | 2     | -    |
| $w_{dist}$ | 100   | -    | $w_y$     | 2     | -    |
| $w_c$     | 10    | -    | $w_{ref}$ | 1000  | -    |
| $w_{col}$ | 1     | -    | $w_{ad}$  | 10    | -    |
| $w_i$     | 2     | -    | $w_{ja}$  | 1     | -    |
| $w_A$     | 0.1   | -    | $w_{jd}$  | 10    | -    |
| $T_{st}$  | 3     | s    | $A$       | [-2:0.5:1] | m/s² |
TABLE 3: IDM parameters used for different driving styles in test scenarios.

| Parameters                | $v_0$ | $s_0$ | $a$ | $b$ | $T$ |
|---------------------------|-------|-------|-----|-----|-----|
|                           | m/s   | m     | m/s²| m/s²| s   |
| Scen.1 aggressive driver  | 20    | 12    | 5   | 5   | 0.4 |
| Scen.1 defensive driver   | 18    | 25    | 2   | 2   | 1.3 |
| Scen.2 aggressive driver  | 10    | 12    | 5   | 5   | 0.4 |
| Scen.2 defensive driver   | 10    | 25    | 2   | 2   | 1.3 |
| Scen.3                     | 0     | 12    | 5   | 5   | 0.4 |
| Scen.4                     | 20    | 12    | 5   | 5   | 0.4 |

described by the following equations:

$$\dot{x} = v$$  \hspace{1cm} (36)$$

$$\dot{v} = a \left( 1 - \left( \frac{v}{v_0} \right)^4 - \left( \frac{s^*(v, \Delta v)}{s} \right)^2 \right)$$  \hspace{1cm} (37)$$

where $s^*(v, \Delta v) = s_0 + vT + v\Delta v/2\sqrt{ab}$, $v_0$, $s_0$, $T$, $a$ and $b$ are the model parameters. The parameters of each vehicle are listed in Table 3.

B. SCENARIOS
1) Scenario 1 (Fast Dynamic Obstacle Scenario)
In this scenario, dynamic obstacles are set to have relatively high-speed profiles using the IDM parameters as shown in Table 3. A total of 10 dynamic obstacles were arbitrarily positioned on two-lane roads within a certain distance in front of the ego vehicle. The ego vehicle is expected to perform appropriate lane change and car-following maneuvers, generating a safe and smooth spatiotemporal trajectory. In Fig. 10(a), the upper figure shows the results of the motion-planning algorithm. The long-term path (blue) is the output of the first part of the algorithm, and the short-term trajectory (light blue) is the result of the last part of the algorithm. The bottom left figure shows the result of the JPS algorithm when solving the problem of the second part of the algorithm in the distance-time plane by displaying the planned trajectory and jump point. The bottom-right figure shows the velocity profile generated by the algorithm. The reference velocity was set to 20 m/s. The long-term profile (blue) is from the second part of the algorithm, and the short-term profile (light blue) is from the last part of the algorithm. In Fig. 10(a), the ego vehicle undergoes a lane change while accelerating to the reference speed. Next, in Fig. 10(b), the ego vehicle keeps the lane while maintaining the reference speed. Finally, in Fig. 10(c), the ego vehicle slows down while performing a car-following maneuver. There are two obstacles in the right lane, therefore a lane maintenance path is generated for the long-term path. Finally, the results in Fig. 10(d) show both the reference velocity and velocity profile of the ego vehicle during the entire scenario. In Fig. 11, the plots show the trajectories of the ego vehicle and obstacles within a certain time gap. The blue trajectory depicted with a triangle inside belongs to the ego vehicle, and the other trajectories belong to the obstacles.
2) Scenario 2 (Slow Dynamic Obstacle Scenario)
In this scenario, dynamic obstacles are set to have relatively low-speed profiles, as shown in Table 3. This scenario aims to evaluate the lane change and car-following performance in low-speed traffic. In Fig. 12(a), a long-term path is generated to change the lane considering the surrounding obstacles. As shown in Fig. 12(b), car-following is performed because the risk of a lane change is high owing to the obstacle behind in the right lane. Then in Fig. 12(c), the ego vehicle performs a safe lane change maneuver as there is sufficient space. In the JPS plot in Fig. 12(c), while changing the lane and avoiding collision with obstacles, the planned trajectory shows that the ego vehicle tries to accelerate to the reference velocity. The overall maneuvering is shown in Fig. 13.

FIGURE 12: Evaluation in scenario 2: The ego vehicle (blue) (a) chooses a lane change maneuver; (b) performs a car-following maneuver; (c) performs another lane change when there is enough space. (d) Speed profile of the ego vehicle during the scenario.

FIGURE 13: Trajectories of the ego vehicle and obstacles in scenario 2.

FIGURE 14: Evaluation in scenario 3: The ego vehicle (blue) (a) performs a lane change maneuver; (b) decreases speed smoothly; (c) stops behind the static obstacles. (d) Speed profile of the ego vehicle during the scenario.

FIGURE 15: The trajectory of the ego vehicle in scenario 3.
3) Scenario 3 (Static Obstacle Scenario)

In this scenario, static obstacles are placed in front of the ego vehicle. This scenario aims to evaluate the obstacle avoidance and stopping performance of the proposed algorithm. In Fig. 14(a), the ego vehicle avoids collision with a static obstacle ahead by choosing a long-term path for lane changing. As can be seen in the JPS plot, an optimal long-term trajectory is obtained which decelerates through the optimal long-term path while keeping it collision-free. In Fig. 14(b) and (c), it can be seen that the ego vehicle continuously decelerates and finally stops to avoid collision with the obstacles. The overall maneuvering is shown in Fig. 15.

4) Scenario 4 (Hazard Dynamic Obstacle Scenario)

In this scenario, the hazard dynamic obstacles overlap with the lane. As shown in Fig. 16(a), the ego vehicle first changes lanes. Then, then, the long-term trajectory is generated as if
there are no obstacles, as shown in Fig. 16(b). However, the short-term trajectory is generated considering the obstacles with collision cost described in Section IV-C, resulting in a trajectory that deviates from the reference trajectory. Next, as shown in Fig. 16(c), a short-term trajectory is generated toward the centerline when the collision risk disappears. The overall maneuvering is shown in Fig. 17.

5) Scenario 5 (Cut-in Scenario)
In this last scenario, a dynamic obstacle performs an abrupt cut-in maneuver in terms of duration and relative speed [44]. The obstacle performs a lane change for 3 s, the relative speed with the ego vehicle is -5 m/s, and the lane change starts 25 m ahead. This scenario is intended to see how the planner reacts to a lane change of the obstacle. In Fig. 18(a), the ego vehicle maintains the lane while avoiding collision, as shown in the JPS plot. Then, in Fig. 18(b), the ego vehicle avoids collision with the cut-in obstacle ahead by choosing a long-term path for lane changing. The ego vehicle successfully performs a lane change to the left, as shown in Fig. 18(c). The overall maneuvering is illustrated in Fig. 19.

The steering wheel angle of the ego vehicle for all test scenarios is shown in Fig. 20. The results show that appropriate steering wheel angle is applied to the ego vehicle in order to perform lane change maneuvers. In the test scenario 5, it is seen that an abrupt lane change is performed by applying a larger steering wheel angle than other scenarios to avoid collision with the cut-in obstacle.

Remark 1. The selected five scenarios include representative situations that may be encountered while driving on a highway. There are static and dynamic obstacles, including hazard vehicles that deviate from the center of the lane. The autonomous vehicle performs lane keeping, following, lane changing, stopping and swerve for safe highway driving.

Remark 2. In the test scenarios, the long-term path planner provides a desired maneuver with a lane-level lateral targets considering the lanes and obstacles. The paths generated show appropriate planning strategy, especially in lane selection. The long-term trajectory planner then applies the position and velocity profiles to the optimal path to avoid collisions with obstacles determined to overlap the path. Experimental results show satisfactory performance covering all scenarios.

Remark 3. The last short-term trajectory planner optimizes the trajectory considering comfort and safety. The optimization using the particles of steering angle sets with an adequate vehicle model makes smooth and feasible trajectory. In addition, the obstacles not considered in long-term planner are considered in optimization procedure to make collision free trajectory.

Fig. 21 shows a histogram of the computation time of the proposed algorithm for all scenarios. The distribution of the computation time is expressed through a histogram with an additional normal distribution. The average computation time was 31 ms, and the worst-case computation time was 94 ms.

VI. CONCLUSION AND FUTURE WORK
In this paper, the proposed hierarchical motion planning framework was shown to be a safe and efficient solution for autonomous driving in structured environments with various static and dynamic obstacles. The proposal was evaluated by means of a VILS with a real car under five representative scenarios.

The long-term planner, which consists of two processes, first provides a rough path for the driving strategy over a long time horizon. Then, the JPS is utilized to reduce the computational burden of A*, giving an acceptable quality solution. The generated trajectory was safe, comfortable, and dynamically feasible. This long horizon planner gives a long-sighted solution which can handle various maneuvers in decision-making where a plenty of obstacles exist.

Next, the short-term planner optimizes the front part of the rough trajectory by first resampling and then applying the PSO. This procedure provides kinodynamically feasible trajectory, enhancing the quality of trajectory in terms of comfort, smoothness and safety. The performance is evaluated under five scenarios, which include virtual static and dynamic obstacles.
dynamic obstacles. The proposed method provides a feasible and reasonable path and trajectory for performing lane changing, car-following, obstacle avoidance, and stopping in an appropriate manner. Furthermore, the computation time of the algorithm was within 100 ms.

As future work, the proposed planning framework will be tested on a broader range of traffic scenarios including urban situations, to further verify and evaluate the performance. The traffic light scenario can be conducted where a long horizon capable planning is important in maneuver decision-making (to pass or to stop at traffic light). In addition, to meet the requirements of advanced prediction methods [45]–[48], interaction-aware trajectory prediction with uncertainty could be considered in motion planning instead of simple physical model-based predictions. Finally, it could be interesting to focus on applying the deep learning technique to make human-like behavior decision-making.

REFERENCES

[1] L. Clausmann, M. Revilloud, D. Gruyer, and S. Glauner, “A review of motion planning for highway autonomous driving,” IEEE Transactions on Intelligent Transportation Systems, 2019.
[2] D. González, J. Pérez, V. Milanés, and F. Nashashibi, “A review of motion planning techniques for automated vehicles,” IEEE Transactions on Intelligent Transportation Systems, vol. 17, no. 4, pp. 1135–1145, 2015.
[3] V. Beanland, M. Fitzharris, K. L. Young, and M. G. Lenné, “Driver inattention and driver distraction in serious casualty crashes: Data from the Australian national crash in-depth study,” Accident Analysis & Prevention, vol. 54, pp. 99–107, 2013.
[4] N. H. T. S. Administration et al., “2015 motor vehicle crashes: overview,” Traffic safety facts research note, vol. 2016, pp. 1–9, 2016.
[5] Y. Meng, Y. Wu, G. Lu, and L. Liu, “A decoupled trajectory planning framework based on the integration of lattice searching and convex optimization,” IEEE Access, vol. 7, pp. 130,530–130,551, 2019.
[6] J. Ziegler, P. Bender, T. Dang, and C. Stiller, “Trajectory planning for bertha—a local, continuous method,” IEEE Intelligent Vehicles symposium proceedings. IEEE, 2014, pp. 450–457.
[7] Z. W. Geem, Recent advances in harmony search algorithm. Springer Science & Business Media, 2010, vol. 270.
[8] J. Chen, W. Zhan, and M. Tomizuka, “Autonomous driving motion planning with constrained iterative lqg,” IEEE Transactions on Intelligent Vehicles, vol. 4, no. 4, pp. 244–254, 2019.
[9] I. Ziegler and C. Stiller, “Spatiotemporal state lattices for fast trajectory planning in dynamic on-road driving scenarios,” in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems. IEEE, 2009, pp. 1879–1884.
[10] M. McNaughton, C. Urmon, J. M. Dolan, and J.-W. Lee, “Motion planning for autonomous driving with a conformal spatiotemporal lattice,” in 2011 IEEE International Conference on Robotics and Automation. IEEE, 2011, pp. 4899–4905.
[11] S. vom Dorff, M. Kneissl, and M. Fränzle, “Safe, deterministic trajectory planning for unstructured and partially occluded environments,” in 2021 IEEE International Intelligent Transportation Systems Conference (ITSC). IEEE, 2021, pp. 969–975.
[12] C. Liu, S. Lee, S. Varnhagen, and H. E. Tseng, “Path planning for autonomous vehicles using model predictive control,” in 2017 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2017, pp. 174–179.
[13] D. Madás, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philipsson, A. Eidehall, and K.-M. Dahlén, “On path planning methods for automotive collision avoidance,” in 2013 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2013, pp. 931–937.
[14] M. Werling, J. Ziegler, S. Kammler, and S. Thrun, “Optimal trajectory generation for dynamic street scenarios in a frenet frame,” in 2010 IEEE International Conference on Robotics and Automation. IEEE, 2010, pp. 987–993.
[15] X. Li, Z. Sun, D. Cao, Z. He, and Q. Zhu, “Real-time trajectory planning for autonomous urban driving: Framework, algorithms, and verifications,” IEEE/ASME Transactions on Mechatronics, vol. 21, no. 2, pp. 740–753, 2015.
[16] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, “Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles,” Mechanical Systems and Signal Processing, vol. 87, pp. 118–137, 2017.
[17] H. Fan, F. Zhu, C. Liu, L. Zhang, L. Zhuang, D. Li, W. Zhu, J. Hu, H. Li, and Q. Kong, “Baidu apollo em motion planner,” arXiv preprint arXiv:1807.08048, 2018.
[18] W. Xu, J. Wei, J. M. Dolan, H. Zhao, and H. Zha, “A real-time motion planning with trajectory optimization for autonomous vehicles,” in 2012 IEEE International Conference on Robotics and Automation. IEEE, 2012, pp. 2061–2067.
[19] Y. Zhang, H. Sun, J. Zhou, J. Pan, J. Hu, and J. Miao, “Optimal vehicle path planning using quadratic optimization for baidu apollo open platform,” in 2020 IEEE Intelligent Vehicles Symposium (IV). IEEE, 2020, pp. 978–984.
[20] L. G. Yu, B. Zhou, P. Chen, Y. Liao, and D. Li, “Semantic-level maneuver sampling and trajectory planning for on-road autonomous driving in dynamic scenarios,” IEEE Transactions on Vehicular Technology, vol. 70, no. 2, pp. 1122–1134, 2021.
[21] Y. Zhang, H. Chen, S. L. Waslander, J. Gong, G. Xiong, T. Yang, and K. Liu, “Hybrid trajectory planning for autonomous driving in highly constrained environments,” IEEE Access, vol. 6, pp. 32,800–32,819, 2018.
[22] W. Lim, S. Lee, M. Sunwoo, and K. Jo, “Hierarchical trajectory planning of autonomous car based on the integration of a sampling and an optimization method,” IEEE Transactions on Intelligent Transportation Systems, vol. 19, no. 2, pp. 613–626, 2018.
[23] X. Jin, Z. Yan, G. Yin, S. Li, and C. Wei, “An adaptive motion planning technique for on-road autonomous driving,” IEEE Access, vol. 9, pp. 2655–2664, 2020.
[24] J. Ziegler, P. Bender, M. Schreiber, H. Lategahn, T. Strauss, C. Stiller, T. Wang, U. Franke, N. Appenrodt, C. G. Keller et al., “Making bertha drive—an autonomous journey on a historic route,” IEEE Intelligent transportation systems magazine, vol. 6, no. 2, pp. 8–20, 2014.
[25] C. Hubmann, M. Aeberhard, and C. Stiller, “A generic driving strategy for urban environments,” in 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC). IEEE, 2016, pp. 1010–1016.
[26] C. Urmon, J. Ahnall, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatai, C. Geyer et al., “Autonomous driving in urban environments: Boss and the urban challenge,” Journal of Field Robotics, vol. 25, no. 8, pp. 425–466, 2008.
[27] J. Leonard, J. How, S. Teller, M. Berger, S. Campbell, G. Fiore, L. Fletcher, E. Frazzoli, A. Huang, S. Karaman et al., “A perception-driven autonomous urban vehicle,” Journal of Field Robotics, vol. 25, no. 10, pp. 727–774, 2008.
[28] J. Montemerlo, J. Becker, S. Bhat, D. Dahlkamp, D. Dolgov, S. Ettinger, D. Hahnel, T. Hilden, G. Hoffmann, B. Huhinke et al., “Junior: The stanford entry in the urban challenge,” Journal of field Robotics, vol. 25, no. 9, pp. 569–597, 2008.
[29] M. Olsson, “Behavior trees for decision-making in autonomous driving,” 2016.
[30] H. Kim, D. Kim, and K. Huh, “Intention aware motion planning with model predictive control in highway merge scenario,” SAE Technical Paper, Tech. Rep., 2019.
[31] C. Hubmann, “Belief state planning for autonomous driving: Planning with interaction, uncertain prediction and uncertain perception,” 2020.
[32] D. D. Harabor and A. Grastien, “Online graph pruning for pathfinding on grid maps,” in Twenty-Fifth AAAI Conference on Artificial Intelligence, 2011.
[33] S. M. LaValle, Planning algorithms. Cambridge university press, 2006.
[34] Y. Shi and R. C. Eberhart, “Emiptial study of particle swarm optimization,” in Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406), vol. 3. IEEE, 1999, pp. 1945–1950.
[35] N. S. Pal and S. Sharma, “Robot path planning using swarm intelligence: A survey,” International Journal of Computer Applications, vol. 83, no. 12, pp. 5–12, 2013.
[36] H. I. Kang, B. Lee, and K. Kim, “Path planning algorithm using the particle swarm optimization and the improved dijkstra algorithm,” in 2008 IEEE Pacific-Asia Workshop on Computational Intelligence and Industrial Application, vol. 2. IEEE, 2008, pp. 1002–1004.
[37] Y. Guo, W. Wang, and S. Wu, “Research on robot path planning based on fuzzy neural network and particle swarm optimization,” in 2017 29th
DONGCHAN KIM received the B.S. degree in automotive engineering from Hanyang University, Seoul, Republic of Korea, in 2015. He is currently working toward the Ph.D. degree with the Department of Automotive Engineering. His research interests include motion planning, trajectory prediction, vehicle state estimation, reinforcement learning, and deep learning.

GIHOON KIM received the B.S. degree in automotive engineering from Hanyang University, Seoul, Republic of Korea, in 2017. He is currently working toward the Ph.D. degree with the Department of Automotive Engineering. His research interests include interaction-aware motion planning, predictions, deep learning and model predictive control for automated vehicle.

HAYOUNG KIM received the B.S. degree in mechanical engineering from Hanyang University, Seoul, Republic of Korea, in 2014. He is currently working toward a Ph.D. degree in the Department of Automotive Engineering, Hanyang University. His research interests include interaction-aware motion planning, predictions, deep learning and model predictive control for automated vehicle.

KUNSOO HUH (M’10) received the Ph.D. degree from the University of Michigan, Ann Arbor, MI, USA, in 1992. He is currently a Professor with the Department of Automotive Engineering, Hanyang University, Seoul, Korea. His research interests include machine monitoring and control, with emphasis on their applications to vehicular systems. His current research focus includes sensor-based active safety systems, V2X-based safety systems, autonomous vehicle control, and AI applications in autonomous vehicle.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see https://creativecommons.org/licenses/by/4.0/