A real-time search-based motion planning framework with risk assessment for urban environments

1st Kailin Tong
Control Systems Group (Dept.-E)
Virtual Vehicle Research GmbH
Graz, 8010 Austria
kailin.tong@v2c2.at

***Abstract*—A reliable autonomous driving system should be capable of handling driving tasks in real-time while ensuring user safety. However, many search-based planners suffer from high computation time, which makes them unsuitable for embedded applications. Some simple methods, like Roll-out from OpenPlanner, achieve high computation efficiency, but lack robustness and might be unsafe in certain conditions. To solve this problem we proposed an efficient search-based planning framework incorporating risk evaluation. Firstly, based on necessary inputs from perception, map, and global planning stack, a 3D grid map is generated to capture environmental semantics for our extended hybrid A* search algorithm. We proposed a heuristic function consisting of risk evaluation, which not only reduces A* search time but also planes a safer trajectory. A two-step optimization process refines the results from the hybrid A* search into a smooth and continuous trajectory. Our proposed framework was tested using realistic simulation scenarios within the open-source Autoware autonomous driving stack, showing superior responses and more robustness than the baseline OpenPlanner. Additionally, the proposed hybrid A* search framework shows high computation efficiency. It has a worst-case run-time of 55 ms for a 12-second planning horizon and a planning horizon of 200 meters, allowing for real-time application.**

***Index Terms*—automated vehicles, motion planning, risk assessment, vehicle safety**

I. INTRODUCTION

Autonomous driving has the potential to significantly improve road safety by removing human error, reduce traffic congestion through improved traffic efficiency, and lower emissions [1, pp. 3–4]. The functional capabilities of vehicle automation have advanced significantly, with some prototypes functioning on roadways [2]. Many researchers also contributed open-source simulation tools and benchmarks to aid in algorithm iteration in addition to the advancement of perception, motion planning, and control technologies [3].

Three dimensions constitute the configuration space for autonomous vehicles on a 2D plane: two dimensions, (x, y), give the location of the vehicle, and one dimension, (theta), specifies the direction of the vehicle. The time dimension must be taken into account while analyzing the velocity, angular velocity, and their differentiation. There has been a lot of work done that uses various state space descriptions to try to tackle this high dimensional challenge, as surveyed in [4], [5]. In this study, we categorize various methods into two orthogonal groups: spatio-temporal linked planning and spatio-temporal decoupled planning, which deal with location and tempo independently and concurrently, respectively.

The first set of planners frequently uses a hierarchical planning framework: given a route, potential paths are planned first, followed by the selection of a preferred maneuver and the generation of a speed profile. OpenPlanner from Autoware [6], which uses on-road verification, is one illustration. In its local planning module, a trajectory generator develops a set of local roll-outs that differ from a reference path, and a behavior generator acts as the orchestrator, selecting an ideal local trajectory and generating a velocity profile while taking driving semantics into account. Similar ideas can be found in [7], [8]. Although decoupling space and time improves computing efficiency, it is difficult to design a path and velocity profile independently to achieve desired movements in some dynamic circumstances such as overtaking and merging. Wang et al. recently integrated reinforcement learning into the hierarchical planning framework and improved its generalization capabilities [9].

Simply adding time and velocity dimensions in the case of spatiotemporal coupled planning results in a massive expansion of the search space. The search space must thus be carefully discretized. Ziegler and Stiller suggest employing a spatiotemporal lattice to depict the search space in structured areas ([10]). One of a predetermined range of durations and velocities links their lattice points together. In [11], this method is improved further by using various constant accelerations and smooth curvature spirals to link the vertices of the lattice.

The large dimension of the lattice makes it expensive to repeatedly build the lattice in dynamic situations, which is one of the disadvantages of lattice planners. In a recent study, the Intelligent Driver Model (IDM) was used as a speed feedback strategy to try to solve this problem, according to [12]. As a result, it just needs a spatial lattice. However, its capacity to respond is constrained by the deterministic speed feedback

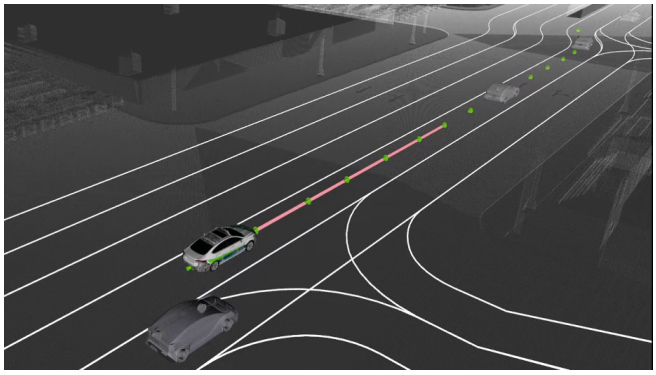


Fig. 1: Illustration of the multiple-lane dense traffic scenario including a curvy road, where our proposed motion planner plans trajectories. The ego vehicle is personalized according to our experiment vehicle. The green arrows are A* search outputs, while the pink strip is the refined trajectory.

policy.

Decoupling the lateral and longitudinal motion in a Frenet frame is another way to simplify the construction of a search graph. To design both rough long-term longitudinal movements and short-term trajectories, the authors in [13] employ A* search and quadratic programming (QP). The combined motion, however, is not confirmed and might result in undesirable behavior in dynamic situations. In the same manner, the authors in [14] utilize a linear lane-change model for hybrid A* search and further incorporate traffic lights into the search space. However, the derived trajectory has a jerky velocity profile and is serrated. It is insufficient for actual vehicle implementation because of this.

The idea of long-term and short-term planning [15] has lately drawn greater attention, as has the need to achieve a balance between the trade-off between the time required to create and search a graph and the optimality of produced trajectories [16], [17].. In the research they conducted, A* search initially offers a preliminary long-term plan, which is then improved by optimization taking safety and dynamical limitations into account. Although collision-free is taken into account in the aforementioned ways, none of them take the collision risk into account when defining the problem. The intended motions infringe on the Responsibility Sensitive Safety (RSS) hypothesis because they are likely to overtake while putting pressure on nearby human drivers and cut in dangerously. Traffic waves and sluggish traffic flow will result from this.

Risk evaluation is a usual function in collision avoidance systems. A common indicator of collision risk is Time-to-X, which includes time-to-collision (TTC), time-to-brake (TTB), and time-to-steer (TTS). Utilizing statistical models, such as Bayesian networks [18] and risk level sets [19], is another approach for estimating risk. The fact that statistical models need a lot of data is one of their shortcomings.

Contributions: In this work, we propose a real-time motion planning framework that takes consideration of collision risk, addressing the motion planning problem in complex urban environments.

Firstly, we provide a computationally efficient framework, which exploits the benefits of discretized A* search and continuous optimization. The worst-case calculation time is bounded by 55 ms. We use a 3D grid map to represent traffic semantics, which has negligible construction time and facilitates quick collision check and risk evaluation. We keep the very needed states in the search space of hybrid A* and propose an admissible heuristic to accelerate it. Also, trajectory and velocity profile refinement has negligible time cost but generates smooth and safe final trajectories.

Secondly, we propose a novel traffic heuristics for A* search which integrates risk assessment. Our planner avoids reckless cut-ins or risky overtaking with on-coming traffic and shows more robustness in different situations.

Finally, we validate our work in simulated scenarios, where measurement noises, control errors, and stochastic agent behavior exist. Thanks to the realistic validation, our proposed framework can be further deployed on cars.

II. PROBLEM STATEMENT

Motion planning is a critical step in achieving high levels of vehicle autonomy. Our motion planning framework's duty is to construct a trajectory from its current position to a target area. The proposed planning pipeline's inputs include an ego state, object list, lane waypoints, and map information, as depicted in Fig. 2. Our motion planning approach produces a trajectory made up of a sequence of waypoints with the states (t, x, y, θ, v) in the Cartesian frame (as explained in Fig. 3).

Urban driving is a typical yet difficult task for it due to the varying roadway geometry, numerous traffic factors, and intense interactions. This is particularly challenging while overtaking since it requires an overtaking vehicle (ego car) to move both longitudinally and laterally while keeping a safe gap from vehicles in front [4]. Generally following requirements should be considered for motion planning in complex urban transportation [13], [20]:

1) Real-time capability on normal usage hardware.
2) Avoiding a crash if not causing another one.
3) Hard traffic rules, including speed limit, traffic signals and right of way.
4) Soft traffic rules, such as keeping a safe distance to the car in front, avoiding reckless lane change and driving in the rightmost lane.
5) Vehicle kinematics feasibility, such as path curvature and acceleration limits.
6) Driving qualities, such as comfort and journey time.

In our work, 1) to 4) are explicitly guaranteed by the motion planning framework. In light of 5), we focus on longitudinal motion while applying a linear lateral model, as suggested in [13], [14]. The curvature is smoothed in the trajectory refinement stage. The last requirement is formulated as an optimization target both in A* search and trajectory and velocity profile refinement. We spotlight normal driving tasks

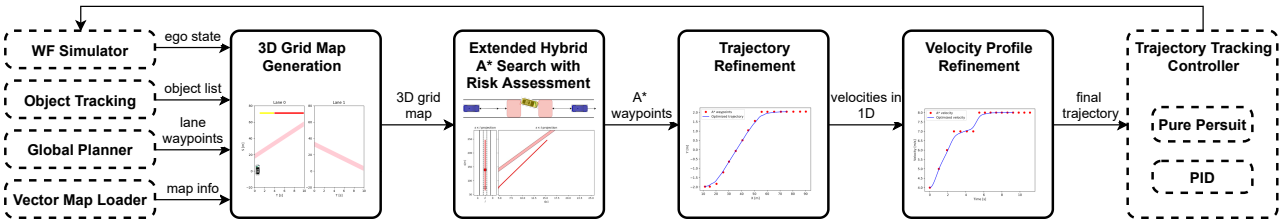


Fig. 2: Illustration of the proposed motion planning framework. Our contributions are in black frames, while dotted frames represent Autoware software.

and exclude the extreme cases in urbane driving, as a collision avoidance system shall take effect [21].

III. MOTION PLANNING FRAMEWORK

In this section, we give a big picture of the proposed motion planning framework and elaborate on its components.

A. Overview of the motion planning framework

Fig. 2 demonstrates the proposed motion planning framework. The necessary inputs are provided by Autoware, based on which a 3D grid map is generated. The real-time 3D grid map is further used in the A* search incorporating risk assessment. Due to the coarse discretization and motion decoupling in A* search, additional trajectory, as well as velocity profile refinement, is imperative. We refine the A* outputs by a two-step approach including points smoothing and minimal jerk polynomial interpolation. The final trajectory is given to the trajectory tracking controller of Autoware to drive the ego vehicle.

In this work, the proposed framework is applied in a scheme of Receding Horizon Control (RHC), which re-plans at a fixed cycle time to make full use of the latest ego and traffic state. The re-planning frequency is 10 Hz. In the following sections, each component of the proposed framework is introduced.

B. 3D grid map

The core idea of a 3D grid map is adding a time dimension to a 2D space so that motion planning takes the future dynamics of ego as well as moving obstacles into account. We adopt Frenet frame representation for 2D space as it is convenient for structured environments and modeling traffic semantics [22]. Typically the driving reference line is extracted from a route including lane information. In a Frenet frame, the space is decoupled into two orthogonal axes s and l. The vehicle states in a Cartesian frame are decoupled into lateral and longitudinal directions, as shown in Fig. 3. The states of tracked objects are projected into Frenet frame as well.

To obtain a grid map with (s,t,l) as dimensions, a behavior prediction model is necessary. A naive approach is to assume that vehicles drive with the same control inputs and invariant maneuvers. Actually, this is a strong basis for many Kalman-Filter-like predictors. In this work, we predict other vehicles' motions by assuming that they drive with a constant longitudinal velocity in the Frenet frame and stay in their current lanes.

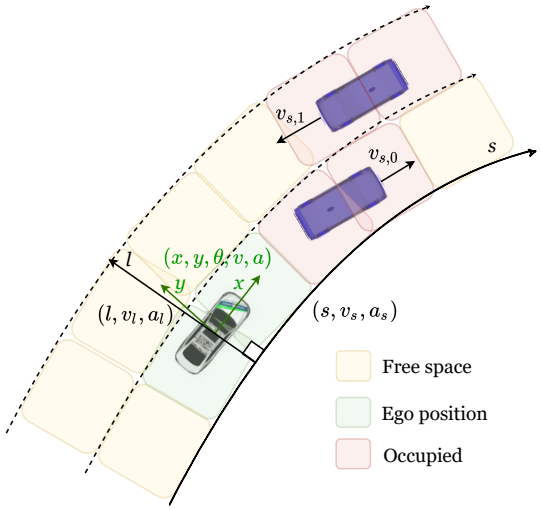


Fig. 3: Illustration of motion planning in a Frenet frame. The vehicle states in a Cartesian frame include (x,y,θ,v,a) , which denote vehicle position, heading, velocity, and acceleration respectively. They are projected onto the driving reference line for motion decoupling. (l,v_l,a_l) represent lateral distance, velocity and acceleration respectively, while (s,v_s,a_s) represent longitudinal quantities. $v_{s,0}$ and $v_{s,1}$ are the longitudinal velocity of tracked object 0 and 1 respectively.

Even if the real movements of other vehicles deviate from the predicted ones, the planned trajectory should be not only collision-free but also have low collision risk. Therefore a frequent re-planning is performed, and collision risk is evaluated during the trajectory generation of A* search. This concept was verified reliable in experiments even when a lead vehicle frequently accelerates and decelerates.

Fig. 4 shows a 3D grip map transformed from the oncoming traffic scenario in Fig. 3. It describes a 3D space Ω by a triple $(s,t,l) \in [s_{min},s_{max}] \times [0,t_{max}] \times \{0,\cdots,N_l\}$. s_{min} and s_{max} are the minimal and maximal distance horizon determined by the rear and front sensor detection range respectively. t_{max} is limited by the accuracy of motion prediction algorithms. The right-most lane index has a value 0, while the allowed left-most lane index is N_l. The center of gravity of a vehicle determines which lane it currently belongs to.

3D grid maps are able to model different traffic semantics, such as crossings, roundabouts, signalized and unsignalized

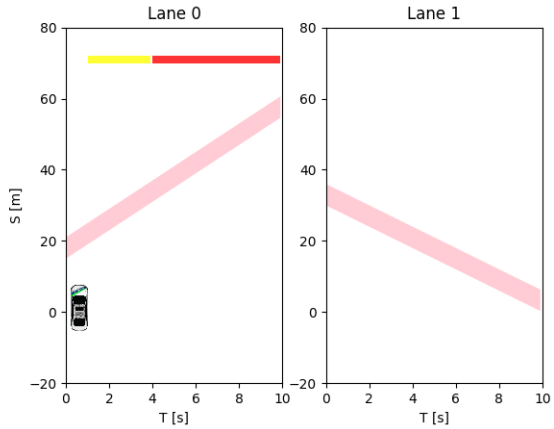


Fig. 4: Illustration of a 3D grid map. The ego vehicle is posed at time step 0 s and distance 0 m. The pink strips represent the current and predicted motions of other vehicles. Yellow and red traffic lights are modeled as yellow and red static obstacles respectively.

intersections [13]–[17]. Therefore it presents a general map representation for different urban environments. In addition, it takes apparently less time to construct a 3D grip map than a spatio-temporal state lattice. Another advantage of using grid map is that it enables quick collision check and risk assessment for A* search.

C. Risk assessment

The risk evaluation undertaken in this paper is based on the concept of RSS. We attach high importance to two "common-sense" rules from RSS: Autonomous vehicles should avoid hitting someone from behind and should not cut in recklessly [20].

TTC is a widely accepted safety indicator, which shows the actual occurrence of hazardous events [23]. We use it to estimate collision risks because its calculation is simple but accounts for both spatial proximity and velocity difference. Another additional risk metric is the Minimal Safety Margin (MSM), defined as the required minimal safety distance within the vicinity of the ego vehicle. If the gap to other cars is smaller than this value, a crash is believed to have happened. We use this metric to overcome the measurement noises and discretization errors and ensure the collision-free requirement even in stand-still situations.

The formula of TTC of vehicle i is defined as follows:

$$TTC_i = \begin{cases} \frac{d_{s,i}-MSM}{v_{s,i}-v_{s,j}} & \frac{d_{s,i}-MSM}{v_{s,i}-v_{s,j}} > 0 \\ +\infty & \text{otherwise} \end{cases} \tag{1}$$

where $d_{s,i}$ is the longitudinal distance of vehicle j from vehicle i at one time step. $v_{s,i}$ and $v_{s,j}$ denote the projected velocity in a Frenet frame for vehicle i and vehicle j respectively. Vehicle j drives in front of vehicle i longitudinally.

A car in urban environments has two basic maneuvers: car-following (CF) and lane-change (LC). LC is further classified into two phases: LC preparation and LC execution, which are determined by whether the center of gravity of the vehicle crosses lane borders (as shown in Fig. 5). In this paper, LC preparation has the same risk assessment as CF as the ego vehicle still stays in the same lane and concentrates on maintaining a safe distance to the vehicle in front.

We introduce front-rear collision risk $R(i)$ of the collision between vehicle i and its lead vehicle. The formula is inspired by [24], in which collision probability is modeled as an exponential function of TTC. We want to keep collision risk under a limit, and refrain from planning a trajectory with collision risk over a threshold. Therefore collision risk is defined as follows:

$$R(i) = e^{-w(TTC_i - TTC_{safe})} \tag{2}$$

where TTC_{safe} is a safety TTC that human accept, and w is a positive tuning parameter. Safety TTC can be obtained from traffic regulations or real traffic data. For example, in [25] the TTC less than 1 second is viewed as "intensive interaction" based on distributions of real traffic data. In this case, the planner should avoid trajectories causing TTC value smaller than 1 second, as it might lead to hard brakes of other vehicles.

During CF or LC preparation (in the upper side of Fig. 5), we focus on the interaction between the ego vehicle and the lead vehicle. The collision risk is denoted by $R(e)$, where e means the ego vehicle.

During LC execution (in the lower side of Fig. 5), we take care of time-to-collision both forward and backward, so as to avoid reckless cut-in. The collision risk is hence defined as $R(e)+R(i)$, where e denotes the ego vehicle and i represents a predecessor in the target lane of LC. We will further use the collision risk in A* search.

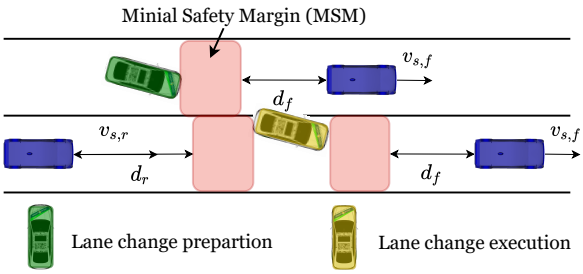


Fig. 5: Illustration of risk assessment in LC preparation and LC execution. d_f is the front longitudinal distance subtracted by MSM, while d_r is the rear longitudinal distance subtracted by MSM. $v_{s,f}$ and $v_{s,r}$ are longitudinal velocities of the lead and following vehicle respectively.

D. Extended hybrid A* approach

The hybrid A* search proposed for the 2007 DARPA urban challenge exploits the advantages of grid discretization and sampling in a control space [26]. Thus it can handle general path-planning tasks such as parking and executing U-turns with acceptable computation time. Traditionally it utilizes a 2D grid map and solves spatial planning problems. In this paper,

we extend it to solve spatio-temporal planning problems based on a 3D grid map.

1) Search space: In this paper, we use a discrete search space(t,s,l,v_s,dir). Time is denoted by t. States s, and v_s follow the same definition in Fig. 3. v_s is naturally constrained by the discretization range. State l is a discrete lane index the same as it is in a 3D grid map (in Fig 4). The last state is the discrete driving direction for LC or CF: $dir \in \{left, right, forward\}$.

The lateral motion is not present in the search space. We adopt the hypothesis in [14], which uses a constant lateral velocity with a fixed LC duration of 5 seconds. The authors in [14] claimed that the LC duration corresponds to human driving habits and is feasible for vehicle kinematics except for driving with a very low velocity. If a lane width is 4 meters, then linear lateral velocity v_l is 0.8 m/s. In normal urban scenarios (e.g. 9km/h - 60km/h), v_l is much smaller than v_s, hence we have $v \approx v_s$.

The proposed search space is the minimal but necessary state space representation for driving in structured environments with a normal velocity. If low-velocity driving like navigating a parking lot is required, another free-space planner will be called.

2) Cost function: Inspired by control theory, the cost function of A* search consists of a penalty for deviation from the desired velocity and accelerations, which is defined as

$$cost = w_1(v_s - v_d)^2 \Delta T + w_2 \bar{a_s}^2 \Delta T \tag{3}$$

where v_d is the desired velocity, ΔT is the expanded time step, $\bar{a_s}$ is the average longitudinal acceleration, w_1 and w_2 are positive tuning parameters. The desired velocity is expressed as

$$v_d = min\{v_c, \sqrt{a_{lat}^{des}/k(s)}\} \tag{4}$$

where v_c is a user-defined cruise velocity, a_{lat}^{des} is the desirable lateral acceleration in body frame, and $k(s)$ is the curvature of the reference line in a Cartesian frame.

3) Heuristic value function: We propose an admissible heuristic to accelerate search and a traffic heuristics to forbid undesired maneuvers during driving. For the cost function definition in Eq. 3, we get an admissible heuristic by solving a single variable optimization problem in 1D, assuming a point mass model with a constant acceleration without constraints. The problem is defined as follows:

$$\min_{a} \quad w_1 \int_0^T (v_s + at - v_d)^2 \, dt + w_2 \int_0^T a^2 \, dt \tag{5}$$

where $T = (v_d - v_s)/a$.

The admissible heuristics $h_a(n)$ is hence written as

$$h_a(n) = \frac{2w_1(v_s - v_d)^2}{\sqrt{12 w_1 / w_2}} + \frac{\sqrt{3} w_1^2 (v_s - v_d)^2}{3 w_2 (w_1/w_2)^{\frac{3}{2}}} \quad (w_1, w_2 \neq 0) \tag{6}$$

Heuristics $h_t(n)$ handles dynamic traffic flow, which incorporates collision risk assessment and encourages driving in the desired lane. It is defined as

$$h_t(n) = \begin{cases} R(e) + R(i) + \varepsilon |l - l_d| & \text{LC execution} \\ R(e) + \varepsilon |l - l_d| & \text{otherwise} \end{cases} \tag{7}$$

where $R(e)$ and $R(i)$ are the collision risk of ego vehicle and follower respectively, ε is a small positive number, and l_d is the desired lane index.

The effect of $R(e)$ and $R(i)$ is that it prunes search branches causing TTC to be lower than a safety TTC. Notice that we use a big weight in the exponent. So this term makes risky LC (with small TTC) have extremely high costs but only has minimal effects if TTC is larger than the safety TTC. This can further ensure collision-free even in the existence of control errors and prediction errors. The second part $\varepsilon |l - l_d|$ gives a small penalty for not driving in the desired lane.

4) Node Expansion and pruning: We follow the expand function in [14], which expands the node both in s axis and t axis. The following branches are removed when expanding a node: next position is occupied by obstacles between the current and next time step in the 3D grid map, or its average longitudinal acceleration exceeds a limit.

In our hybrid A* search, once an expanded node has states exceeding s_{max} or t_{max}, the search is stopped and provides a solution as a sequence of continuous states $(t,s,l,\Delta\theta,v_s,v_l)$ in the Frenet frame. It is not guaranteed that hybrid A* search finds a minimal cost solution, due to the coarse discretization for long horizon and merging of continuous-coordinate states occupying the same grid in the discrete search space. However, it provides feasible, collision-free, and low-risk reference waypoints and velocity for further optimization. Also, as shown in practical experiments of [26], the resulting trajectory from hybrid A* search lies near the global optimum. This allows us to refine the trajectory and velocity profile with relatively low costs, which is elaborated in the following sections.

E. Trajectory refinement

The A* outputs in the Frenet frame are firstly transformed to states (t,x,y,θ,v) in the Cartesian frame. Directly connecting waypoints of (x,y) leads to a serrated path. Therefore we refine the trajectory with two steps: (1) points smoothing and (2) minimal jerk polynomial interpolation. The waypoints are firstly smoothed using the conjugate gradient (CG) method [6].

Due to the nature of discretization, similar continuous-coordinate states might have the same grid index. For example, 0.1 m and 0.9 m are located in the same grid if s is discretized with a grid size of 1 m. As a result, our smoothed waypoints have minor differences from the original ones for discretized planning and remain collision-free.

Next, we use polynomial trajectories developed for unmanned aerial vehicles to efficiently interpolate between waypoints while minimizing the sum of squared jerks [27].

The minimal jerk trajectory generation in one axle is formulated as a constrained QP:

$$\begin{aligned} \min_{\mathbf{p_1},\cdots,\mathbf{p_M}} \quad & \begin{bmatrix} \mathbf{p_1} \\ \vdots \\ \mathbf{p_M} \end{bmatrix}^T \begin{bmatrix} Q_1(T_0) & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & Q_M(T_M) \end{bmatrix} \begin{bmatrix} \mathbf{p_1} \\ \vdots \\ \mathbf{p_M} \end{bmatrix} \\ s.t. \quad & A_{eq} \begin{bmatrix} \mathbf{p_1} \\ \vdots \\ \mathbf{p_M} \end{bmatrix} = \begin{bmatrix} \mathbf{d_1} \\ \vdots \\ \mathbf{d_M} \end{bmatrix} \end{aligned} \tag{8}$$

In this expression, $\mathbf{p_i}$ is the coefficient vector of the quintic polynomial describing the i^{th} segment. $Q_i(T_i)$ is the Hessian matrix for the i^{th} segment. A_{eq} is a mapping matrix. $\mathbf{d_i}$ is a vector containing the i^{th} segment's derivative values for the beginning (d_0) and end (d_T).

The outputs from the A* search provide sufficient information to solve this QP problem. For example, the i^{th} segment of polynomial connects the $(i\text{-}1)^{th}$ waypoint and the i^{th} waypoint, which have states $(t_{i\text{-}1}, x_{i\text{-}1}, y_{i\text{-}1}, \theta_{i\text{-}1}, v_{i\text{-}1})$ and $(t_i, x_i, y_i, \theta_i, v_i)$ respectively. Thus, T_i is given by $t_i - t_{i\text{-}1}$. For x axle, $\mathbf{d_i}$ is constructed as $[x_{i\text{-}1}, v_{i\text{-}1}\cos\theta_{i\text{-}1}, 0, x_i, v_i\cos\theta_i, 0]^T$. Note that here 0 represents the continuous constraint for an endpoint connecting two polynomial segments rather than a fixed value. The same projection is applied to the y axle as well.

Luckily, the aforementioned constrained QP can be converted to an unconstrained QP, and solved directly for endpoint derivatives instead of polynomial coefficients. The optimal closed-form solution is written as [27]:

$$\mathbf{d_P}^* = -R_{PP}^{-1} R_{FP}^T \mathbf{d_F} \tag{9}$$

where $\mathbf{d_F}$ is a grouped vector of specified derivatives, and $\mathbf{d_P}$ is a grouped vector of unspecified derivatives. R_{PP} and R_{FP} are intermediate matrices for calculation.

The refined trajectory can be recovered from the resulting derivatives. One example of trajectory refinement for LC is shown in Fig. 6. Note that only a part of the trajectory is refined, as the trajectory tracking control system only utilizes the reference trajectory near the ego vehicle.

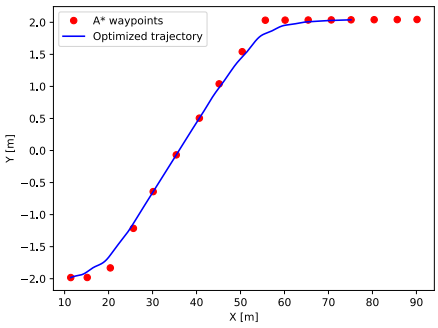


Fig. 6: Illustration of trajectory refinement.

F. Velocity profile refinement

Although the refined trajectory contains the velocity information in the x axle and y axle, direct utilization of the resultant velocity leads to a wavy velocity profile. Therefore we apply the refinement on v in a 1D manifold again.

Similarly, discrete velocities from the A* search are slightly smoothed. Next, we use cubic polynomials to connect discrete velocities and generate a minimal jerk velocity profile along the path. The problem formulation and solution is omitted for brevity, as only the number of coefficients and derivatives are different in Eq. 8 and Eq. 9. Fig. 7 shows one example of velocity profile refinement.

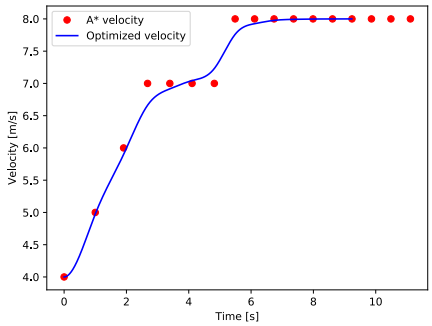


Fig. 7: Illustration of velocity profile refinement.

IV. EVALUATION

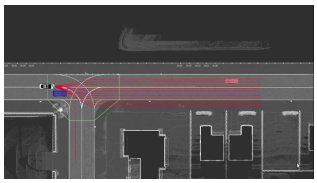
In this section, we evaluate the proposed approach in two challenging urban scenarios and benchmark it against OpenPlanner. Our proposed motion planning framework is implemented in C++ based on Robot Operation System (ROS) with a computer equipped with an Intel Core i7-9700 CPU. Particularly, the A* search dominated the run-time. Compared with [14], we use smaller or equal discretization (Δv = 1m/s, Δs = 5m, ΔT = 1 s) and longer horizon (200 m and 12 s). However, in dense traffic, the average run-time of our A* search is **4.7** ms, much less than the reported average search time in [14]. The worst-case computation time of the whole framework is **55** ms. In the following sections, we detail how the realistic simulation scenarios are created, and demonstrate qualitative and quantitative results in those scenarios.

A. Simulation environment

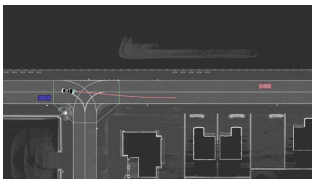
Most of the mentioned publications validate their motion planner in a "perfect" world, where no noise exists and all agents drive with a constant velocity. To get more realistic simulations, we use the simulation packages offered by Autoware, where measurement noises, localization errors as well stochastic agent behavior are available. The simulated agents attempt to drive with a maximal velocity but with noise and react to other agents. So they can create realistic traffic shocks in urban environments.

An internally developed tool – ViFWare, functions as the interface among autonomous driving functions, simulators, maps, sensors, and actuators. We use the vehicle model from WF Simulator of Autoware, but personalize it with the parameters of our experiment vehicle. This facilitates the further on-road test.

The first validation scenario is overtaking with on-coming traffic, where the time window for overtaking is limited and decisive action is required. The second one is multiple-lane dense traffic, where 5 agents drive around the ego vehicle along with frequent acceleration and deceleration. We show the qualitative results in the overtaking with the on-coming traffic scenario and the quantitative results in the second scenario.

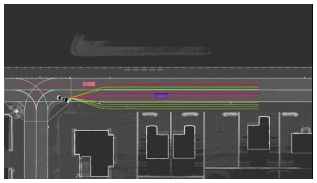


(a) OpenPlanner

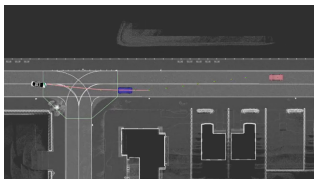


(b) Our approach

Fig. 8: Use Case 1: Overtaking with Limited Time Window



(a) OpenPlanner



(b) Our approach

Fig. 9: Use Case 2: Giving Right of Way

B. Overtaking with on-coming traffic

The map we used here is the vector map of Carla Town 01 from open-source Carla Simulator [28], which consists of 2 lanes traveling in opposite directions. In order to create intensive interactions between the ego vehicle and the on-coming vehicle, we design two use cases as follows:

1) Use Case 1: Overtaking with Limited Time Window: The lead vehicle drives with maximal 3 m/s, while the on-coming vehicle drives with maximal 10 m/s. In the simulation, the proposed motion planner foresees the possible crash with the on-coming vehicle and accelerates to come back to its own lane, as shown in Fig. 8(b). Due to the decoupling of velocity and path planning, the OpenPlanner firstly decelerates and then starts to overtake when getting close the slow lead vehicle. However, during overtaking it does not accelerate aggressively to come back to the original lane when the on-coming vehicle is quickly approaching. Since the time window of a successful overtaking is missed, all possible paths are blocked (in red) and hazards take place, as shown in Fig 8(a).

2) Use Case 2: Giving Right of Way: The lead vehicle drives with a maximum of 4.5 m/s, while the on-coming vehicle drives with a maximum of 12 m/s. As our planner estimates a high collision risk in the overtaking lane, the ego vehicle gives up overtaking and gives the right of way even though the distance to the on-coming vehicle is roughly 65 meters. This makes a smooth and comfortable coming back. On the contrary, OpenPlanner gives the right of way only when the distance to the on-coming vehicle is smaller than a fixed safety threshold without considering the high on-coming tempo. As a result, the steering is extremely harsh to avoid a collision and the ego vehicle drives off the road (in Fig 9(a)).

C. Multiple-lane dense traffic

This scenario is motivated by Carla Town 03, which consists of multiple lanes and curvy roads for urban environments. We simulate 5 agents around the ego vehicle, including 2 slow agents (with a maximal velocity of 4 m/s) and 3 normal agents (with a maximal velocity of 10 m/s). Due to the space limit, this scenario is only shown in the attached video.

We benchmark our approach against OpenPlanner with the same route (450 m) and the same parameters for simulated agents. Note that we carefully tuned OpenPlanner's parameters to ensure that its driving is collision-free and has similar motion constraints as ours (e.g. acceleration limits, safety border). The comparison result is shown in Table I, where DTC_{min} and TTC_{min} are minimal Distance-to-Collision and minimal Time-to-Collision respectively. The collision check is implemented by the three-disk approximation in [29] assuming a constant velocity and orientation. In addition, $|\kappa|_{max}$ denotes the maximal curvature of the vehicle's real trajectory, $\bar{v}$ represents the average velocity during the journey, and $\bar{e}$ is the average tracking error, which is expressed as the closest distance between the current position and reference trajectory. T_{worst} is the worst-case computation time of the planner in the presence of 5 agents. Some observations can be made

TABLE I: Comparison of results in a 450 m route

Planner	DTC_{min} (m)	TTC_{min} (s)	$\lvert\kappa\rvert_{max}$ (m^{-1})	$\bar{v}$ (m/s)	$\bar{e}$ (m)	T_{worst} (ms)
Ours	3.97	4.90	0.04	7.22	0.12	55.0 [1]
Open Planner	5.68	4.10	0.11	4.33	0.19	< 100 [2]

[1] Time consists of: search time (54.6 ms), trajectory refinement (0.3 ms), velocity profile refinement (0.1 ms) and 3D grid map update time (0.1 ms).
[2] The estimated time is from [6].

out of Table I. The proposed planner and OpenPlanner both can finish the route without any collisions, but the proposed approach focuses more on a bigger TTC, while OpenPlanner tries to enlarge DTC due to the safety distance setup. Our planner provides a more comfortable trip for the passenger, as the vehicle motion of our approach has much smaller curvature than OpenPlanner's. Moreover, the proposed approach reduces average tracking errors by 37% compared to OpenPlanner while driving with a higher velocity in dense traffic. This shows that the proposed planner plans a trajectory closer to the global optimum even though noise and uncertainties exist. The computation time of the proposed approach is bounded by 55 ms in the worst-case, roughly half of the planning cycle time, therefore it can run on embedded hardware.

V. CONCLUSIONS

In this paper, we propose a novel motion planning framework for complex urban environments. The framework is capable of providing a safe and smooth trajectory in real-time and is robust in different scenarios such as overtaking with on-coming traffic and multiple-lane dense traffic. We demonstrate that the planner can correctly make the optimal lane-change

decisions and avoid reckless overtaking and cut-in while showing better driving performance. Moreover, the trajectory and velocity refinement reduces the trajectory tracking errors compared to the state-of-the-art OpenPlanner and improves driving comfort.

In the future we will extend the work in several promising directions: (1) A data-driven prediction model can be applied in the framework to predict the motions of other road users; (2) Reinforcement learning can be integrated into the A* search to further improve the efficiency; (3) A more complex vehicle motion model can be used during trajectory refinement to extend the operational design domain.

ACKNOWLEDGMENT

The publication was written at Virtual Vehicle Research GmbH within the scope of the projects ArchitectECA2030 and ESRIUM. The project ArchitectECA2030 has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 877539. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Germany, Austria, Czech Republic, Netherlands, Lithuania, Latvia, France, Sweden, Norway. In Austria the project was also funded by the program "IKT der Zukunft" of the Austrian Federal Ministry for Climate Action (BMK). The project ESRIUM has received funding from the European GNSS Agency under the European Union's Horizon 2020 research and innovation programme under grant agreement No 101004181. The publication was written at Virtual Vehicle Research GmbH in Graz and partially funded within the COMET K2 Competence Centers for Excellent Technologies from the Austrian Federal Ministry for Climate Action (BMK), the Austrian Federal Ministry for Labour and Economy (BMAW), the Province of Styria (Dept. 12) and the Styrian Business Promotion Agency (SFG). The Austrian Research Promotion Agency (FFG) has been authorised for the programme management. The content of this paper reflects only the authors' view. Neither the European Commission nor the European GNSS Agency and the JU is responsible for any use that may be made of the information it contains.

REFERENCES

[1] D. Watzenig and M. Horn, *Automated Driving*. Springer International Publishing, 2017.

[2] E. Yurtsever, J. Lambert, A. Carballo, and K. Takeda, "A survey of autonomous driving: Common practices and emerging technologies," *IEEE Access*, vol. 8, pp. 58 443–58 469, 2020.

[3] K. Tong, Z. Ajanovic, and G. Stettinger, "Overview of tools supporting planning for automated driving," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, 2020, pp. 1–8.

[4] S. Dixit, S. Fallah, U. Montanaro, M. Dianati, A. Stevens, F. Mccullough, and A. Mouzakitis, "Trajectory planning and tracking for autonomous overtaking: State-of-the-art and future prospects," *Annual Reviews in Control*, vol. 45, pp. 76–86, 2018.

[5] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 1, no. 1, pp. 187–210, 2018.

[6] H. Darweesh, E. Takeuchi, K. Takeda, Y. Ninomiya, A. Sujiwo, L. Y. Morales, N. Akai, T. Tomizawa, and S. Kato, "Open source integrated planner for autonomous navigation in highly dynamic environments," *Journal of Robotics and Mechatronics*, vol. 29, no. 4, pp. 668–684, 2017.

[7] T. Gu, J. Atwood, C. Dong, J. M. Dolan, and J.-W. Lee, "Tunable and stable real-time trajectory planning for urban autonomous driving," in *2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2015, pp. 250–256.

[8] 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.

[9] Jingke Wang, Yue Wang, Dongkun Zhang, Yezhou Yang, and Rong Xiong, "Learning hierarchical behavior and motion planning for autonomous driving," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 25-29, 2020, Las Vegas, NV, USA*, 2020.

[10] J. 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*, 2009, pp. 1879–1884.

[11] M. McNaughton, C. Urmson, 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, 09.05.2011 - 13.05.2011, pp. 4889–4895.

[12] Ke Sun, Brent Schlotfeldt, Stephen Chaves, Paul Martin, Gulshan Mandhyan, and Vijay Kumar, "Feedback enhanced motion planning for autonomous vehicles," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), October 25-29, 2020, Las Vegas, NV, USA*, 2020.

[13] W. Zhan, J. Chen, C.-Y. Chan, C. Liu, and M. Tomizuka, "Spatially-partitioned environmental representation and planning architecture for on-road autonomous driving," in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 6/11/2017 - 6/14/2017, pp. 632–639.

[14] Z. Ajanovic, B. Lacevic, B. Shyrokau, M. Stolz, and M. Horn, "Search-based optimal motion planning for automated driving," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 4523–4530.

[15] 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, 11/1/2016 - 11/4/2016, pp. 1010–1016.

[16] W. Ding, L. Zhang, J. Chen, and S. Shen, "Safe trajectory generation for complex urban environments using spatio-temporal semantic corridor," *IEEE Robotics and Automation Letters*, vol. 4, no. 3, pp. 2997–3004, 2019.

[17] T. Zhang, M. Fu, W. Song, Y. Yang, and M. Wang, "Trajectory planning based on spatio-temporal map with collision avoidance guaranteed by safety strip," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–14, 2020.

[18] S. Noh and K. An, "Risk assessment for automatic lane change maneuvers on highways," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5/29/2017 - 6/3/2017, pp. 247–254.

[19] A. Pierson, W. Schwarting, S. Karaman, and D. Rus, "Navigating congested environments with risk level sets," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 5/21/2018 - 5/25/2018, pp. 5712–5719.

[20] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[21] M. Schratter, M. Hartmann, and D. Watzenig, "Pedestrian collision avoidance system for autonomous vehicles," *SAE International Journal of Connected and Automated Vehicles*, vol. 2, no. 4, 2019.

[22] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenét frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 03.05.2010 - 07.05.2010, pp. 987–993.

[23] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis & Prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[24] J. Eggert, "Predictive risk estimation for intelligent adas functions," in *17th International IEEE Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 10/8/2014 - 10/11/2014, pp. 711–718.

[25] Stefano Masi, Philippe Xu, and Philippe BONNIFAIT, "A curvilinear decision method for two-lane roundabout crossing and its validation under realistic traffic flow," in *2020 IEEE Intelligent Vehicles Symposium (IV), October 20-23, 2020, Las Vegas, NV, USA*, 2020.

[26] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Practical search techniques in path planning for autonomous driving," *Ann Arbor*, vol. 1001, no. 48105, pp. 18–80, 2008.

[27] C. Richter, A. Bry, and N. Roy, "Polynomial trajectory planning for aggressive quadrotor flight in dense indoor environments," in *Robotics Research*, ser. Springer Tracts in Advanced Robotics, M. Inaba and P. Corke, Eds. Cham: Springer International Publishing, 2016, vol. 114, pp. 649–666.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[29] J. Ziegler and C. Stiller, "Fast collision checking for intelligent vehicle motion planning," in *2010 IEEE Intelligent Vehicles Symposium*, 2010, pp. 518–522.