Real-Time Safe Stop Trajectory Planning via Multidimensional Hybrid A*-Algorithm

Lingguang Wang[1], Zhenkang Wu[1], Jiakang Li[1] and Christoph Stiller[1]

***Abstract*— A reliable autonomous driving system should be capable of performing safe stop when proceeding the normal driving becomes impossible. It is essential for safe stop planning to be able to provide a trajectory that leads the vehicle to a specific stopped goal in real-time. With this as a main challenge and distinction, the safe stop planning should still be able to avoid collision with static and dynamic obstacles like normal motion planning. To guarantee a meaningful solution and be real-time capable, we propose to utilize path-velocity decomposition, which provides a non-globally optimal solution but reduces the computational burden. Firstly, by sampling piecewise quintic polynomials that connect a series of sampled points from the current location to the goal, a set of path candidates in Frenét frame are generated. The path that meets kinematic constraints, maximizes comfort, and avoids static obstacles is selected. Afterward, we generate the length-time (ST) graph by projecting the dynamic obstacles on our driving corridor along the chosen path in space and time domain. The velocity planning is performed on the ST-graph with an extended multidimensional Hybrid A-Star (A*) Algorithm. Finally, our approach is evaluated in several simulation scenarios and also in CoInCar-Simulation framework, which shows a real-time capability and promising driving behaviors.**

I. INTRODUCTION

Nowadays, the requirements for autonomous vehicles stay not only at efficiency and convenience level, but more at safety level. In the recent years, many major vehicle manufactures and research institutions have invested huge manpower and resources to develop fully self-driving vehicles. However, most of them focus on how to bring the vehicle to the goal more efficiently and comfortably, but rarely consider a back-up plan for cases whenever their designed systems fail. This might include situations when the self-driving system experiences a internal problem, the vehicle is involved in a tiny collision, or when the environmental conditions change in a way that would affect normal driving within our operational design domain. After encountering those situations, the system should determine an appropriate response to keep the vehicle and its passengers safe, including pulling over or coming to a safe stop.

In comparison to the emergency braking, the events that trigger the safe stop are usually not that urgent, i.e. the vehicle still maintains part of its automation ability such that it can still follow one pre-calculated trajectory until stopping. However, assuming that the vehicle has lost its main perceptional and computational ability, replanning the safe stop trajectory while following one is in this case not considered. Therefore, the safe stop planning should be executed in parallel with the normal planner in a sufficient operating frequency and whenever the system failure occurs, the calculated safe stop trajectory before it happens, which was verified as safe based on the previous perceived and anticipated environment, will be pursued until the end. This raises several new questions in addition to the normal trajectory planning:

- How to generate trajectories that stops the vehicle exactly on the chosen location with certain orientation while avoiding static and dynamic obstacles, e.g. moving pedestrians, bicycles, as shown in Fig. 1?
- How to ensure safety given only single-shot perception and prediction of the environment?
- How to meet the kinematic constraints (jerk, acceleration and velocity limits, etc.) of the vehicle?
- How to fulfill the real-time requirements while meeting all the aforementioned constraints?

By answering these questions, we could state the main contributions of this paper:

- Our safe stop planning approach could generate trajectory that guides the vehicle to a specific goal location/pose with zero final velocity on arbitrary road topology.
- Our approach is prediction algorithm agnostic, i.e. we can incorporate the Responsibility-Sensitive Safety

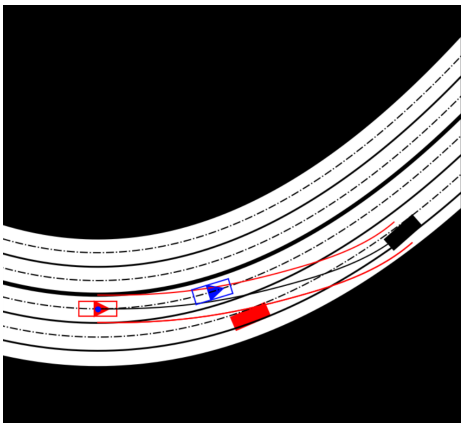


Fig. 1: Safe stop trajectory (center of the red corridor) from the current position (non-solid red rectangle) to the goal (solid black rectangle) while avoiding static (solid red rectangle) and dynamic obstacles (blue rectangle).

*This work is accomplished within the project "UNICARagil" (FKZ 16EMO0287) and the financial support from the Federal Ministry of Education and Research of Germany (BMBF) is acknowledged.

[1]Authors are with the Institute of Measurement and Control Systems, Karlsruhe Institute of Technology (KIT), Karlsruhe, Germany lingguang.wang, stiller@kit.edu, wuzhenkang111@gmail.com, mazeljk@gmail.com

(RSS) [1] along with the reachable set based prediction [2] of other road users as the underlying prediction concept to ensure provable safety.
- Our approach meets the real-time requirements and avoids static and dynamic obstacles given their states and predictions.
- The trajectory comes out from the safe stop planner is comfort in the sense of minimizing jerk, and feasible w.r.t. the vehicle kinematic constraints, which is controllable by a highly close-to-real controller in CoInCar-Simulation [3].

The remainder of the paper is organized as follows: Section II discusses the related work. In Section III, some preliminaries of safe stop planning are given. The planning approach is then described in Section IV. In Section V, we evaluate the approach in some simulation scenarios and in CoInCar-Simulator. Finally, we conclude this paper and discuss the future work in Section VI.

II. RELATED WORK

Some surveys report on the state-of-art of autonomous vehicle in the field of motion planning are [4], [5]. Trajectory planning is referred as the subfield of motion planning, which generates reference trajectories for the ego vehicle. In the field of robotics, graph search method is widely used. The space is discretized into grid and graph searching algorithms visit the different states in the grid and find the path connecting the initial and end state. A*-Algorithm, which is an extension of Dijkstra's graph search algorithm, is used to search in high-dimensional space [6]. Ziegler et.al. [7] implemented the A* with Voronoi cost function for planning in unstructured environment and shows good and fast result for practical path planning problem. However, the planning is only performed in a static environment without taking dynamic obstacles into account. In [8], they proposed hybrid A*, which captures the continuity state of the vehicle in the discrete of A* and guarantees the kinematic feasibility of the planned path. Sampling-based planners draw samples either in deterministic [9] or probabilistic [10] manner.

As for sampling with predefined-pattern, Pivtoraiko et.al. [11] proposed the concept of state lattice, which embeds a discrete graph composed of kinematically-feasible motion primitives. However, the trajectory candidate pool grows fast due to refined sampling resolution and a long horizon. In addition, randomized sampling like RRT* [12], BiRRT* with hybrid curvature steer [13] combine graph construction and graph searching process, which further guarantee asymptotic optimality. Despite this, a slower convergence due to random sampling occurs, which inevitably introduces higher computation time.

Optimization-based methods formulate motion planning as a mathematical optimization problem. The planning results are continuous, optimal and spatiotemporal. The state-of-the-art work at this field is done by Mercedes-Benz [14], which utilized Sequential Quadratic Programming (SQP) to solve the nonlinear and non-convex problem. Chen et.al. [15] proposes the Constrained Iterative Linear Quadratic Regulator (CILQR) to handle the general form of constraints for Iterative LQR. It has superiority on time efficiency, which makes it a potential method that can be utilized in real-time. However, there is no generally well-defined formulation of the optimization problem in the context of automated driving. For each specific scenario, it should be designed appropriately.

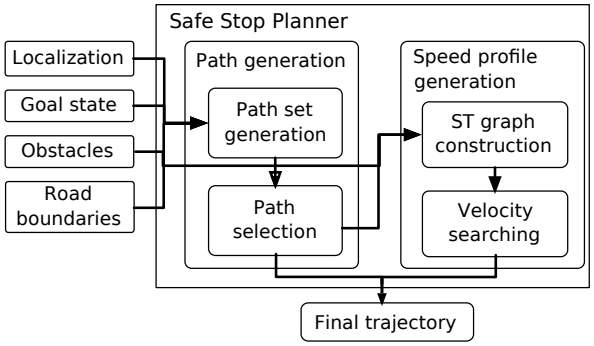


Fig. 2: Pipeline.

In contrast to the normal trajectory planning problem, safe stop trajectory planning receives limited interest of researchers. One related work is the contingency planner presented by Salvado et al. in [16]. The planner uses graph search with the help of precomputed motion primitives and is capable of planning reasonable and fast trajectory. But suboptimality is difficult to guarantee and the algorithm does not have a specific stop location. In the recent paper [17], considering the high complexity and time consumption of solving the non-convex optimization problem, the authors sample some goal positions on the road and discrete the vehicle's initial state. Trajectory planning problem is formulated as an optimal control problem (OCP). They generate a large set of trajectories with different initial state and goal positions by solving OCP offline. At runtime, the subset of trajectories associated to the current vehicle state is selected and the trajectory with the lowest cost w.r.t. comfort and safety in the subset will be executed. They run their algorithm only on straight road and it is almost impossible to setup trajectory set to cover all initial states. Therefore, this approach is not flexible enough.

This paper aims to provide a safe stop planning approach that can lead the vehicle to a specific stop location smoothly on arbitrary road topologies in real-time while fulfilling all the internal and external constraints.

III. PRELIMINARIES

The vehicle has nonlinear dynamics

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{1}$$

where $\mathbf{x} \in \mathbb{R}^m$ and $\mathbf{u} \in \mathbb{R}^n$ are the state and input vector of the system. To avoid collision with static and dynamic obstacles, constraints on space at specific time t can be expressed by

$$\Gamma(\mathbf{x}(t)) \in \Omega_{free}(t) \tag{2}$$

$\Gamma(\mathbf{x}(t)) : \mathbb{R}^m \rightarrow \mathbb{R}^2$ denotes the occupancy of the ego vehicle at state $\mathbf{x}(t)$, which is derived from the m-dimensional states of the vehicle (position, orientation, etc.). $\Omega_{free}(t)$ denotes the free space that is not occupied by static and dynamic obstacles at t .

As mentioned in Section I, the localization of the ego vehicle is given as a prior, as well as the states and predictions of all the static and dynamic obstacles. In order to ensure safety in terms of RSS, set-based prediction of other traffic participants could be adopted. However, this will further limit the possibilities of finding a meaningful solution as more free spaces will be occupied in the future. Our approach is allowed to fail in case that all the space on the road is not available for driving, due to too conservative prediction of other road users. In this case, the back-up plan of full-braking on the ego lane always exists. On the contrary, if other road users are anticipated to keep their velocities and follow the traffic rules, the solution pool will be opened up such that obtaining more comfortable and feasible trajectories becomes possible.

Furthermore, as the approach focuses on trajectory planning, choosing optimal target location is not included in the task, which itself is a complex enough decision making problem. The goal location is assumed as given by the higher-level behavior planning module, which has access to the High-Definition (HD) Map frameworks that provides traffic rule information and the location of possible temporary parking slots, e.g. Lanelet2 [18].

IV. SAFE STOP TRAJECTORY PLANNING

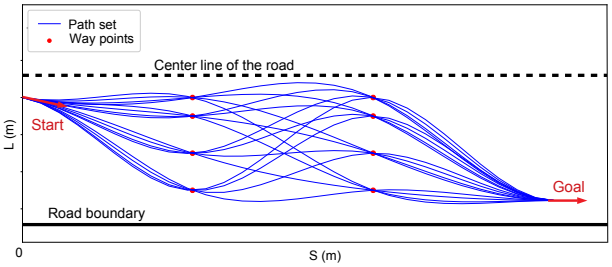
The safe stop trajectory planning is performed in two steps: path generation and speed profile generation. Fig 2 illustrates the whole pipeline. As inputs, the goal position, localization, all the information about the obstacles and the drivable corridor of the road are given.

A. Path Generation

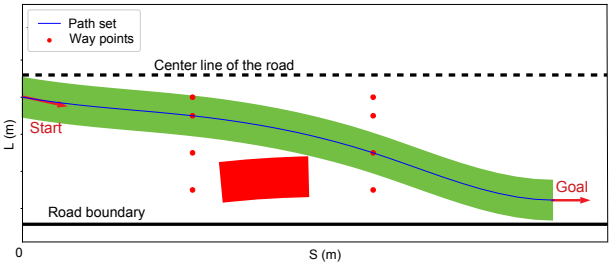
In the first step, we project the static obstacles, location and orientation of the ego vehicle, goal location and orientation and road boundaries from world coordinate to the Frenét frame [19] w.r.t. the centerline of the road. The static obstacles after projection will be represented as polygons (the red polygon in Fig 3(b)).

As shown in Fig 3(a), we sample a series of waypoints between the start state and the goal state (e.g. two columns and four points in each column) in Frenét frame. By generating piecewise quintic polynomials between points of each column, a set of paths connecting start and goal can be obtained. As for the boundary conditions, the first derivatives of the polynomials at the start and goal points are set to the corresponding yaw angles, and the second derivatives (yaw rate) are 0. After examining each path in the path set, the ones that have no collision with the road boundaries and the static obstacles will be firstly selected as path candidates.

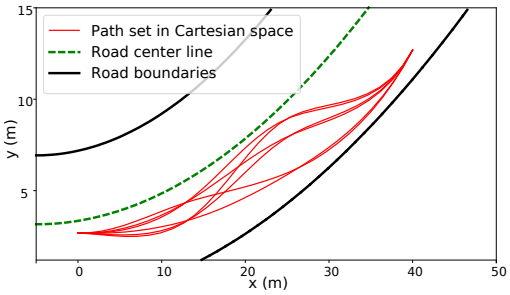
In order to check collision, we assume that the ego vehicle has negligible control error and small side slip angle while driving along the path. Thus, we sample some points on



(a) Generated path set in Frenét frame.



(b) Collision check of the DC with static obstacle (red polygon).



(c) Collision free splines (red curves) in Cartesian space.

Fig. 3: Path generation, collision check and coordinate transformation

each path equidistantly and offset them perpendicularly to the left and right side to construct two new borders, which envelope the occupancy area of the vehicle while driving along the path. We call it driving corridor (DC) later on. The offset distance is half of the vehicle's width added by a designed safe distance. In Fig 3(b), the DC is represent by the green polygon. Thus, those paths whose DCs overlap with the obstacles' polygons in Frenét frame can be screened out from the path candidates. Among those left paths, the one that potentially provides more comfort behavior and is safer is preferable.

We transform all the left path candidates back to Cartesian space based on continuous curvature derivative reference line [19], as shown in Fig 3(c) (red splines). The cost for each path π is formulated as:

$$J(\pi) = w_1 \int_0^S \kappa(s) d_s + w_2 \int_0^S v^2(s)\kappa(s) d_s + w_3 \frac{1}{d(\pi)} \tag{3}$$

where S is the total length of the path. $\kappa(s)$ represents the curvature in Cartesian space. $v(s)$ provides a rough velocity profile along the path by assuming the vehicle decelerates constantly and stops at S, in order to estimate the lateral

acceleration for each path. d represents the minimal distance to the static obstacles. w_1 to w_3 are the weights that should be manually fine-tuned via experiments. Apparently, the path with the lowest cost can be derived using (3).

B. Velocity-Profile Generation

After having a path, the remaining problem is to know where exactly on the path the vehicle should be at each time step. This can be represented by a length-time-profile on the ST-graph, which is simplified as s-profile later on.

We construct the DC of the ego vehicle along the selected path in world coordinate using the aforementioned method. In order to avoid dynamic obstacles while following the path, when and where these obstacles will interfere with our DC should be known. This is straightforward given their predictions. By checking collision between the DC and the predicted obstacles at each time step t, the corresponding arc length S along the path where the collision happens can be derived. On the ST-graph (first row of Fig 4(b)), the dynamic obstacles that interfere with our DC are represented as polygons (red polygons in Fig. 4(b)). The remaining area is the collision free region where the vehicle can enter.

There are several constraints for the s-profile which connects the origin and target arc length line: 1) initial and goal position, i.e. the s-profile should start from origin and end in the goal arc length. 2) initial and end velocity, i.e. the first derivative of the s-profile should be the initial velocity at the origin and 0 at goal. 3) velocity limits, i.e. the first derivative of the s-profile should neither exceed the speed limits nor lower than $0m/s$. 4) acceleration limits, i.e. the second derivative of the s-profile should not exceed the dynamic limits of the vehicle. 5) collision free, i.e. the s-profile should not intersect with the obstacles' polygons.

We would like to get the final s-profile via solving a time-invariant linear optimal control problem (OCP):

$$\begin{aligned} \min \sum_{k=1}^{N-1} & (x_k - x_r)^T Q (x_k - x_r) + u_k^T R u_k \\ s.t. \;\; & x_{k+1} = A x_k + B u_k \\ & x_{min} \le x_k \le x_{max} \\ & u_{min} \le u_k \le u_{max} \\ & x_0 = \overline{x}, x_N = \underline{x} \end{aligned} \tag{4}$$

where k is the time step and N is the total number of time steps. Q and R are weight matrices. $x \in \mathbb{R}^3$ is the state vector with $x_k = [s_k, v_k, a_k]^T$ (distance, velocity and acceleration). $x_r = [s_{k,r}, 0, 0]^T$ is reference state vector, i.e. the deviation of s_k from the $s_{k,r}$ on the searched s-profile is penalized. $u \in \mathbb{R}$ is the scalar control input (jerk) with $u_k = j_k$. A is the simplified longitudinal system matrix which represents a constant acceleration model:

$$A = \begin{bmatrix} 1.0 & d_t & \frac{d_t^2}{2} \\ 0.0 & 1.0 & d_t \\ 0.0 & 0.0 & 1.0 \end{bmatrix} \tag{5}$$

d_t is the discrete time interval between steps. B is the control vector with $B = [\frac{d_t^3}{3} \quad \frac{d_t^2}{2} \quad d_t]^{\mathrm{T}}$. $\overline{x}$, $\underline{x}$ are the initial and end constraints for initial state x_0 and end state x_N. The constraint for state vector x_k is:

$$\begin{bmatrix} s_{k_{min}} \\ v_{min} \\ a_{kmin} \end{bmatrix} \le x_k = \begin{bmatrix} s_k \\ v_k \\ a_k \end{bmatrix} \le \begin{bmatrix} s_{k_{max}} \\ v_{max} \\ a_{max} \end{bmatrix}, 1 < k < N-1 \tag{6}$$

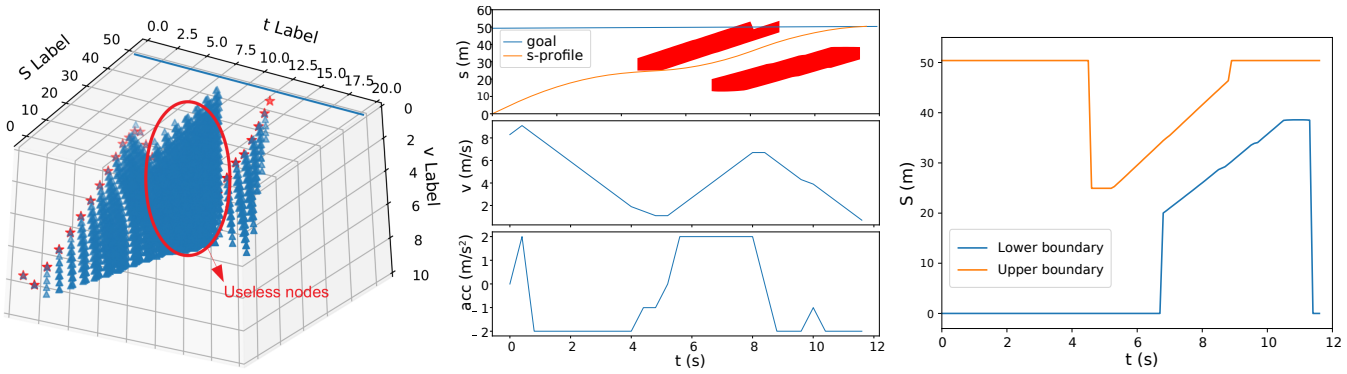
For v_k, a_k and control input j_k, we can set constant lower and upper bounds according to vehicle' kinematic limitations. However, there is no clear continuous bounds for s_k on ST-graph at each time k which makes the problem highly non-convex, i.e. it is not straightforward to set $s_{k_{min}}$ and $s_{k_{max}}$.

To overcome this problem, we extend the Hybrid A*-Algorithm to search a coarse s-profile on the ST-graph that is not directly controllable for the vehicle but at least meets part of the constraints. Afterwards, based on the first result, we could construct convex bounds for s_k. However, searching only on length-time domain can hardly ensure final velocity constraint. Therefore, we add an additional dimension - velocity - to the ST graph and convert the searching space from 2D (length-time) to 3D-space (length-velocity-time), where the state vector $\mathbf{s}$ consists of (s, v, t). The algorithm is shown in Algorithm 1.

Algorithm 1: Extended Hybrid A*

Data: Initial continuous state $\mathbf{s}_{start}$ and $\mathbf{s}_{goal}$
Result: Path from $\mathbf{s}_{start}$ to $\mathbf{s}_{goal}$
Discretize $\mathbf{s}_{start}$ and s_{goal} to n_{start} and $\{n_{goal}\}$
$OpenSet \leftarrow n_{start}$
$CloseSet \leftarrow n_{start}$
Assign $g(n_{start}) = 0$, $g(n) = +\infty$ for all graph nodes
Calculate $h(n_{start})$ for initial node n_{start}
while $OpenSet \neq \emptyset$ **do**
  $n_{current} = \underset{n \in OpenSet}{arg\ min}\ f(n)$
  $OpenSet$.pop($n_{current}$)
  if $n_{current} \in \{n_{goal}\}$ **then**
    return $traversedPath(n_{current})$
  end if
  for each
  $n_{child} \in nextNode(n_{current}, ControlInputs)$ **do**
    if $n_{child} \in CloseSet$ or $n_{child} \in Obstacles$ or $outOfLimits(n_{child})$ **then**
      Continue
    else if $n_{child} \in OpenSet$ **then**
      $f(n_{child})$.update()
    else
      $OpenSet \leftarrow n_{child}$
    end if
  end for
end while
return $NoPathFound$

The $ControlInputs$ are sampled acceleration values acc, e.g. $(-4, -3, \ldots, 1, 2)m/s^2$. The cost function is $f(n) = \gamma_1 g(n) + \gamma_2 h(n)$. n is one discrete node with discrete distance, velocity and time information (s_n, v_n, t_n). $g(n)$ is



(a) Nodes in closeSet. (b) s-profile, v-profile and acc-profile . (c) Constraint for length s_k.

Fig. 4: Results of the extended Hybrid A*-Algorithm.

the cost from the initial node to the node n with: $g(n) = g(n_{parent}) + \gamma_3 * |acc| + \gamma_4 * \kappa(s_n){v_n}^2$, where $g(n_{parent})$ is cost of the parent node. $\kappa(s_n)$ is the curvature at s_n. This cost penalizes longitudinal and lateral acceleration. γ_1, γ_2, γ_3 and γ_4 are weights that need to be tuned. The heuristic function $h(n) = s_{n_{goal}} - s_n$ represents the distance from the current node to the goal nodes. Note that the goal in 3D-space is rather a line (set of nodes $\{n_{goal}\}$) instead of a single node. The explored nodes in $CloseSet$ after solving the ST-graph is shown in Fig. 4(a). The red nodes construct the 3D path. The s-profile can be obtained by projecting the red nodes to ST plane, as depicts in Fig. 4(b), where the v-profile and acc-profile are also plotted.

Having a first s-profile helps conduct the lower and upper bounds for s_k, which was infeasible before. This is done by extracting the largest possible drivable area around the s-profile on ST-graph, as depicts in Fig. 4(c). In other words, the extended A*-Algorithm finds the appropriate gap for the vehicle to enter.

As we have the linear constraints for all the state variables and control variable in (4), the optimal control problem can be converted to standard quadratic problem (QP) and can be solved efficiently, for which the OSQP (Operator Splitting Quadratic Program) solver [20] is utilized. Some of the optimized s-profiles are shown in Fig. 5. Finally, the safe stop trajectory can be resampled from the selected path and the s-profile.

V. EVALUATION

A. Evaluation in three Scenarios in Simulation

To evaluate the present approach, three scenarios are simulated. The resulting trajectories and their corresponding velocity, acceleration and jerk profiles are shown in Fig. 5. The three scenarios have different configurations in term of the road structure and the number of obstacles. The initial velocity and velocity limit of the ego vehicle is $8.3m/s$ and $10m/s$ for all scenarios respectively. Other traffic participants are predicted to drive with constant velocity along one known path. But if we want to be safer, other more conservative predictions mentioned in Section III can also be adopted, which however limit the number of feasible safe stop trajectories. Before implementing our approach in real traffic, a good compromise should be found first.

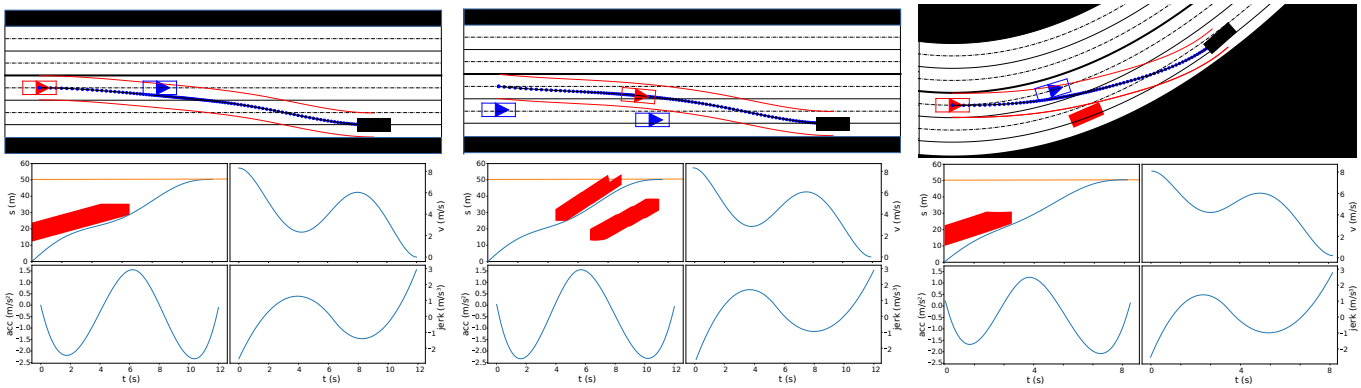
In Scenario 1 (Fig. 5(a)), there is one slow moving dynamic obstacle in front. The ego vehicle decelerates to not collide into it and moves to the right side simultaneously. In scenario 2 (Fig. 5(b)), there are two dynamic obstacles on the adjacent lane. The safe stop trajectory tries to reach the goal by passing through the gap between the two cars. The scenario 3 (Fig. 5(c)) is a replication of scenario 1 on curved road while having another static obstacle and the trajectory shows similar behavior.

This algorithm is implemented in C++ and runs on a laptop cpu (intel CORE i7 8th Gen). The time consumptions for the three scenarios are listed in Table I. Overall, the computation of the pipeline can be done within $30ms$. Note that in the second scenario, executing the extended Hybrid A* is significantly more expensive. We can find clues from Fig. 4(a), which illustrates all the nodes in $CloseSet$ that are visited by the algorithm. Firstly, the algorithm tries to find solution by overtaking the first vehicle in the adjacent lane to reach the goal as soon as possible. However, after realizing that it's impossible to stop at goal while ensuring collision-freeness, it traces back to wait until the adjacent vehicle passes through. This actually imitates one decision making process on determining which gap suites the safe stopping better, which certainly raises the need to explore more nodes in 3D-space and results in huge amount of useless visits.

B. Evaluation in CoInCar-Simulation

We also evaluated our algorithm in CoInCar-Simulation in order to prove that our trajectory is able to be followed by a close-to-real controller.

In Fig. 6, the vehicle first follows the normal trajectory and the safe stop planning is executed simultaneously (Fig. 6(a)). The goal location is picked on the right side of the road with certain safety distance to the border. The arc distance of the goal to the current location is selected depending on the current velocity. Note that the trajectory makes a large turning before entering the roundabout, which reduces the lateral acceleration. After manually triggering the safe stop mode, the ego vehicle performs harsh brake to avoid collision

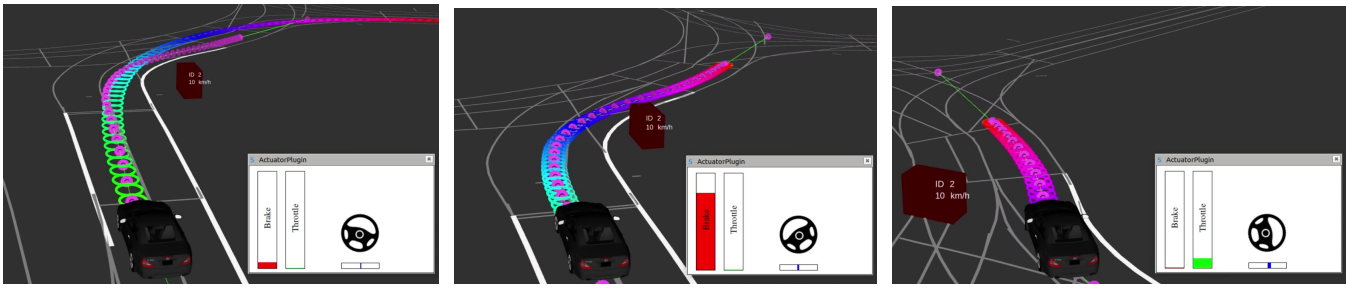


(a) Straight road with one dynamic obstacle (b) Straight road with two dynamic obstacles (c) Curved road with one dynamic obstacle

Fig. 5: Simulation results in different scenarios. Dynamic obstacles are shown in blue. Ego vehicle are represented by red rectangles with one red triangle inside. Black solid rectangles are goal states.

TABLE I: Time consumption in different scenarios

Time (ms)	Path generation & selection	Extended hybrid A*	s-profile optimization	Other processing	Total time
Straight one obstacle	2.2	6.2	1.2	1.3	10.9
Straight two obstacle	3.2	15.1	1.4	1.5	21.2
Curvy one obstacle	2.5	5.8	1.3	1.6	11.2



(a) (b) (c)

Fig. 6: Simulation in CoInCar-Simulator. (a) The circles represent the normal trajectory, while the pink line is the safe stop trajectory. Both circles and lines in (b) and (c) represent the safe stop trajectory as there is no normal trajectory anymore. The red bar and green bar in the right bottom window represent the braking and throttle strength.

with the cyclist (Fig. 6(b)). After that, it accelerates slightly to reach the goal faster (Fig. 6(c)) and then comes to a full stop. The controller is able to stop the vehicle within the limited braking and steering potential without damaging the overall comfort too much, even though the vehicle encounters an abrupt emergency situation and needs a harsh braking.

VI. CONCLUSION AND FUTURE WORK

In this paper, a path-velocity decoupled method for real-time safe stop trajectory planning is presented. Piecewise quintic splines is utilized to generate smooth path candidates in Frenét frame. By extending the Hybrid A*-Algorithm to 3D-space, a coarse s-profile that meets the required constraints can be generated on ST-graph. After conducting the boundaries for the s-profile, the OCP can be transformed to standard convex QP that can be solved efficiently.

Our approach limits the goal location and orientation explicitly, considers static and dynamic obstacles and is still able to provide feasible trajectory. It is potentially not only suitable for safe stop planning but also for other goal-oriented trajectory planning which might has non-zero constraint for final velocity.

Furthermore, our approach has large potential for maintaining stability in terms of computational time even with more dynamic obstacles. As explained in Section V, the extended Hybrid A*-Algorithm did part of the job for decision making while trying to find the best time gap to pass. We can parallelize them by introducing several heuristic cost functions which simulates distinct driving styles (aggressive, defensive) to explore different gaps. In case of several feasible gaps, the fastest or the most comfortable one will be chosen depending on the preference of the driver. Another parallelization that can be done for finding better trajectories is applying the extended Hybrid A*-Algorithm on more paths. By comparing the resulting trajectory of each path, we could find a solution that is closer to the globally optimal solution.

REFERENCES

[1] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *CoRR*, vol. abs/1708.06374, 2017. [Online]. Available: http://arxiv.org/abs/1708.06374

[2] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, June 2016.

[3] M. Naumann, F. Poggenhans, M. Lauer, and C. Stiller, "Coincar-sim: An open-source simulation framework for cooperatively interacting automobiles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, June 2018, pp. 1–6.

[4] 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, pp. 1135–1145, 2016.

[5] B. Paden, M. Čáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, March 2016.

[6] M. Likhachev and D. Ferguson, "Planning long dynamically feasible maneuvers for autonomous vehicles," *The International Journal of Robotics Research*, vol. 28, no. 8, pp. 933–945, 2009.

[7] J. Ziegler, M. Werling, and J. Schroder, "Navigating car-like robots in unstructured environments using an obstacle sensitive cost function," in *2008 IEEE Intelligent Vehicles Symposium*. IEEE, 2008, pp. 787–791.

[8] 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.

[9] U. Schwesinger, P. Versari, A. Broggi, and R. Siegwart, "Vision-only fully automated driving in dynamic mixed-traffic scenarios," *it-Information Technology*, vol. 57, no. 4, pp. 231–242, 2015.

[10] S. Klemm, J. Oberländer, A. Hermann, A. Roennau, T. Schamm, J. M. Zollner, and R. Dillmann, "Rrt-connect: Faster, asymptotically optimal motion planning," in *2015 IEEE International Conference on Robotics and Biomimetics (ROBIO)*. IEEE, 2015, pp. 1670–1677.

[11] M. Pivtoraiko, R. A. Knepper, and A. Kelly, "Differentially constrained mobile robot motion planning in state lattices," *Journal of Field Robotics*, vol. 26, no. 3, pp. 308–333, 2009.

[12] S. Karaman and E. Frazzoli, "Incremental sampling-based algorithms for optimal motion planning," *Robotics Science and Systems VI*, vol. 104, no. 2, 2010.

[13] H. Banzhaf, L. Palmieri, D. Nienhüser, T. Schamm, S. Knoop, and J. M. Zöllner, "Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–8.

[14] J. Ziegler, P. Bender, T. Dang, and C. Stiller, "Trajectory planning for bertha—a local, continuous method," in *2014 IEEE intelligent vehicles symposium proceedings*. IEEE, 2014, pp. 450–457.

[15] J. Chen, W. Zhan, and M. Tomizuka, "Constrained iterative lqr for on-road autonomous driving motion planning," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017, pp. 1–7.

[16] J. Salvado, L. M. Custódio, and D. Hess, "Contingency planning for automated vehicles," in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2016, pp. 2853–2858.

[17] L. Svensson, L. Masson, N. Mohan, E. Ward, A. P. Brenden, L. Feng, and M. Törngren, "Safe stop trajectory planning for highly automated vehicles: an optimal control problem formulation," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 517–522.

[18] F. Poggenhans, J.-H. Pauls, J. Janosovits, S. Orf, M. Naumann, F. Kuhnt, and M. Mayr, "Lanelet2: A high-definition map framework for the future of automated driving," 11 2018, pp. 1672–1679.

[19] M. Werling, J. Ziegler, S. Kammel, 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.

[20] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *ArXiv e-prints*, Nov. 2017.