2024 IEEE International Conference on Robotics and Automation (ICRA 2024)
May 13-17, 2024. Yokohama, Japan

Multi-Profile Quadratic Programming (MPQP) for Optimal Gap Selection and Speed Planning of Autonomous Driving

Alexandre Miranda Añon[1], Sangjae Bae[1], Manish Saroya[1], David Isele[1]

***Abstract*— Smooth and safe speed planning is imperative for the successful deployment of autonomous vehicles. This paper presents a mathematical formulation for the optimal speed planning of autonomous driving, which has been validated in high-fidelity simulations and real-road demonstrations with practical constraints. The algorithm explores the inter-traffic gaps in the time and space domain using a breadth-first search. For each gap, quadratic programming finds an optimal speed profile, synchronizing the time and space pair along with dynamic obstacles. Qualitative and quantitative analysis in Carla is reported to discuss the smoothness and robustness of the proposed algorithm. Finally, we present a road demonstration result for urban city driving.**

I. Introduction

Trajectory planning has been a vibrant area of research engagement in autonomous driving. While foundational planning algorithms have existed for decades [1], [2], calls to navigate efficiently, precisely, and smoothly through highly dynamic and uncertain scenes where safety is critical have pushed researchers to probe the limits of existing algorithms in an effort to prioritize often conflicting objectives [3], [4], [5]. Due to both their interpretability and theoretical guarantees, formal optimization methods such as model predictive control (MPC) and quadratic programming (QP) still remain popular choices for trajectory planning and control tasks for autonomous driving [6], [7].

Motion planning in the presence of dynamic and uncertain agents can result in highly non-convex problems [8], [9]. These can either 1) be slow to solve thereby limiting the optimality or time horizon of the solution, or 2) make use of approximations that get stuck in local optima, resulting in shaky or otherwise undesirable trajectories [10]. Thus, path and speed decomposition methods have been actively adopted by relieving planning subtasks in terms of complexity [11], [8], [12], [13]. By decoupling the path planning problem from the speed planning problem, we can restrict the optimization space, resulting in faster high-quality solutions [14].

For decades, there have been collective efforts to improve the decomposition method [12], [15]. Especially for autonomous driving applications, a space-time (ST) graph [16], [17] has been a popular choice to refine lower and upper bounds, thus reducing the search space. The ST graph also helps address dynamic obstacles and their associated ordering problems [18] (see the motivational example in Fig. 1).

[1]Honda Research Institute, USA, Inc. San Jose, CA 95134, USA {alexandre_mirandaanon, sbae, manish_saroya, disele}@honda-ri.com

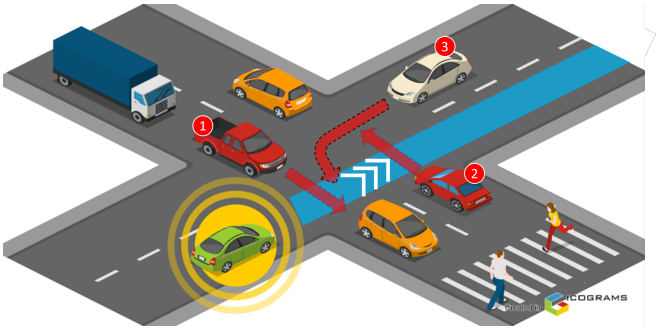

Fig. 1. Motivational example: there exists multiple timing options of crossing the intersection for the ego vehicle (in green) along with the crossing timings of cars 1, 2, and 3. Precisely synchronizing the timing and distance as well as choosing the best gaps is the key for a safe maneuver.

While the ST graph simplifies the problem, the synchronization of time and space remains challenging [19]. To address that, the authors in [13] added synchronized time as a decision variable and solved the non-convex problem with iterative QP approximations. However, the iterative process is often computationally heavy and approximations may lead to local optima [20]. Similarly, the author in [16] employed an ST graph and approximated the non-convex problem as QP, but solved it in the space domain to ease the integration of position-based speed limits (e.g., for curved paths). Yet, planning in the space domain cannot seamlessly consider zero speed as it causes the time evaluation to be infinity [21], problematic in stop-and-go scenarios. In [22], [17] the authors applied Dynamic Programming (DP) to directly optimize non-convex problems on the ST graph. However, DP typically suffers from the curse of dimensionality, and thus is often limited in number of states and control inputs. Customization [22] and heuristic solutions [17] (e.g., using spiral curve interpolations [22]) are inevitable for real-time implementation, while the number of states and controls remain limited. Overall, the existing methods are promising and mathematically sound, but have yet to address rigorous ways of designing speed planners that simultaneously: (i) search solutions in temporal space, (ii) secure global optimality without approximations, (iii) have high computational efficiency without heuristics, and (iv) address space-time synchronization along with dynamic obstacles. Importantly, thorough validations in various scenarios, often missing in academic literature, are essential before deploying a method in an automated vehicle.

Thus, we further explore a rigorous speed planner design by encompassing collective efforts in optimal speed planning under the path-speed decomposition scheme. We adopt the

space-time graph to refine the lower and upper bounds, while the refinement is along with the temporal steps. This allows for a direct and efficient integration of lower and upper bounds into a QP. The QP, as a convex optimization, yields a globally optimal solution that secures comfort and safety. While robust, hard constraints often lead to overly conservative behaviors (if not infeasible solutions) that leave a real-time system with no course of action. We choose to combine hard and soft constraints to produce behaviors that remain safe while dynamically adapting to the complexities of the current scene.

In short, this paper contributes to the literature by presenting a mathematical framework that (i) explores multiple combinations of spatial and temporal scenarios associated with dynamic obstacles and (ii) optimizes the speed profile for the scenarios explored. Importantly, the formulation remains in the temporal domain and addresses the space-time synchronization in a convex optimization. We also introduce techniques to integrate lateral acceleration limits along with curvatures on a given path (and thus account for kinematic limits). The proposed framework has been heavily validated in both simulation and road tests.

II. Space-time (ST) Graph Construction and Domain Initialization

A. Overview

This section details the algorithm that sequentially (i) generates the space-time graph, (ii) finds cases of passage orders among traffic agents with respect to the ego vehicle (we call it "**profile**"), and (iii) obtains the lower and upper bounds associated with each profile. We assume the knowledge of the current state, path-to-follow, and predictions on other traffic.

To illustrate the algorithm, see Fig. 2. Given the ego car's path (in yellow), the predictions on other traffic (with constant speed predictions) indicate the ego vehicle intersecting with three vehicles (veh 1, veh 2, and veh 3). The vehicles are shaped as a polygon and we draw a (time t, distance s) trajectory where the polygon intersects the given path – thus, the ST graph. As an instance of the ST graph, the plot at the top center conveys: veh 1 currently intersects with the path and will drive along the path; veh 2 is approaching with a lower speed and will merge into the path later time; veh 3 is closest to the ego vehicle and will cross the path in a very low speed. The ST Cell Planner then segmentizes the continuous ST graph and finds "viable cells" (representing a discretized feasible space). Within the segmentation, Breadth-First Search (BFS) finds the possible cases of passage order with respect to the ego vehicle (i.e., profile). Given the profile, the associated lower and upper bounds are generated considering the kinematic limits (green space in the bottom left plot).

When multiple profiles exist, the best profile is chosen with respect to the minimum cost of QP. Note that the profiles are independent of each other, thus applicable for parallel computations.

B. Construction of Space-Time Graph

Aligned with the prior efforts in [22], [16], the construction of the ST graph is an essential step for MPQP as it is the base for determining the feasible space of the quadratic programming in a later step.

For each agent, we compute a polygon based on its dimensions and predicted positions. This polygon can be oversized by parameterized margins or uncertainty measures (e.g., noise, risk, or prediction noise). If the polygon intersects with the path of the ego vehicle, the intersecting area (time and distance) is marked on the ST graph. Note that the ST graph is in Frenet frame [23], and thus we convert the intersecting area to Frenet frame[1]. After all, each agent is represented as two line segments in the (time, distance) coordinate. Note that these lines can be shifted to account for safety margins.

C. Viable Cell Generation

A "viable cell" represents a feasible set in the ST graph, excluding the space occupied by other agents (represented by "occupied cells"). At each time segment, a cell is defined by the bounds in distance s, i.e., cell $= [s_{\min}, s_{\max}]$. For brevity, we call the algorithm of the segmentation and cell generation a "Cell Planner". The cell planner first identifies occupied cells associated with each agent at each time segment. It is possible that multiple occupied cells exist. If the occupied cells overlap, they are combined. The viable cells are then found as a complementary set of the occupied cells. For instance, at the time segment with $t = 2$, suppose there exist three vehicles (i.e., three occupied cells):

$$\text{occupied cells}(t=2) = \begin{cases} oc_1 = [4, 6] \\ oc_2 = [5, 8] \\ oc_3 = [20, 25]. \end{cases} \tag{1}$$

As oc_1 and oc_2 are overlapped, they are combined:

$$\text{occupied cells}(t=2) = \begin{cases} \widetilde{oc_1} = [4, 8] \\ \widetilde{oc_2} = [20, 25] \end{cases} \tag{2}$$

The viable cells are the complementary set of $\widetilde{oc_1}$ and $\widetilde{oc_2}$:

$$\text{viable cells}(t=2) = \begin{cases} vc_1 = [0, 4] \\ vc_2 = [8, 20] \\ vc_3 = [25, \bar{s}] \end{cases} \tag{3}$$

where the $\bar{s}$ is the upper bound in distance.

D. Profile Search

We adopt BFS [24] to discover possible profiles in the ST graph. The search process reads:

1) Starting from the origin, progress one step forward in time.
2) Expand a profile for each viable cell if the cell overlaps with any viable cells in the previous step.
3) Otherwise, terminate and discard the profile.

[1]Note that we use "distance" and "space" interchangeably in Frenet frame.

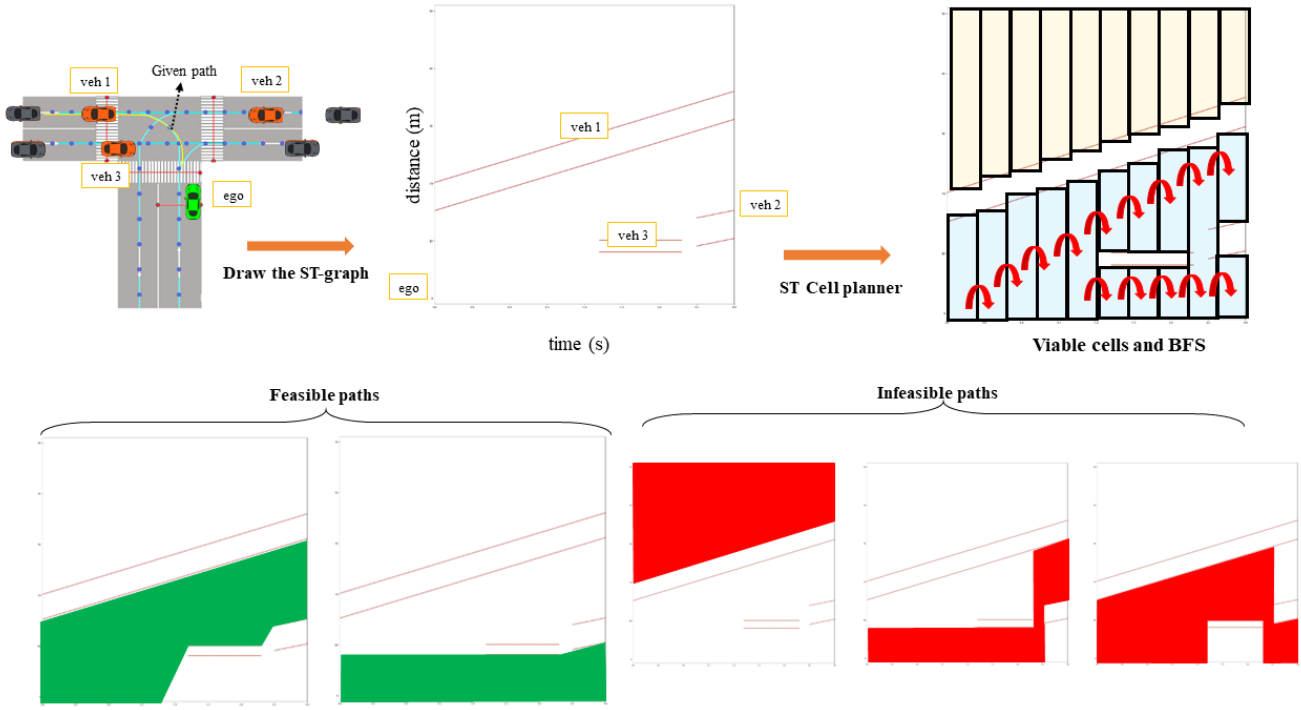

Fig. 2. T-junction scenario (top left), corresponding ST-graph (top center) and ST Cell Planner (top right). ST Cell Planner shows also the possible paths found using the Breadth First Search Algorithm (BFS). Bottom images shows the feasible and infeasible paths found in the ST-graph.

4) If the time step reaches the end (i.e., planning horizon), return all valid profiles.

The step 2 indicates that multiple viable cells can be connected in each time step (see the top right plot in Fig. 2). If no viable cell exists at the origin, the ST graph is not traversable, and thus no profile exists.

E. Integration to QP

Recall, constructing the ST graph is eventually to facilitate the underlying speed optimization. There are three techniques we employ: **First**, we segmentize the ST graph with the equivalent time step (0.1 sec) to that of QP. This is straightforward while being highly effective in synchronizing the path and the speed. **Second**, we generate a lower bound of each profile by connecting the lower end of viable cells at each time step. Similarly, an upper bound is generated by connecting the upper end of viable cells. **Third**, we post-filter all valid profiles with kinematic limits (in speed, acceleration, and jerk). This helps reduce the number of profiles as well as ensure the (longitudinal) kinematic limits.

Depending on the step size, the overall process might be computationally expensive. However, processing each profile in Section II-D and II-E is completely separable, thus parallel computations resolve the issue. A resulting profile is represented by a pair of boundaries with synchronized time. That can be seamlessly integrated into QP.

III. Optimization Problem

Given the piecewise linear bounds obtained from the ST graph, we formulate quadratic programming [20] that optimizes speed and keeps safety.

1) System Dynamics: The state includes [distance, speed, acceleration] denoted by $[p, v, a]$, and the control is jerk j. We aim to find the optimal speed profile along with the longitudinal direction, and thus we leverage the simplified longitudinal dynamics:

$$p(t+1) = p(t) + v(t)dt, \quad \forall t \in \{0, \dots, N-1\}, \tag{4}$$
$$v(t+1) = v(t) + a(t)dt, \quad \forall t \in \{0, \dots, N-2\}, \tag{5}$$
$$a(t+1) = a(t) + j(t)dt, \quad \forall t \in \{0, \dots, N-3\}. \tag{6}$$

With the augmented variable $x = [p, v, a, j]$, we write the dynamics in the compact form:

$$x(t+1) = f(x(t)), \tag{7}$$

where $f(\cdot)$ is the linear mapping with Eqn. (4)-(6).

A. Objective Function

The objective is to enhance passenger comfort and reduce travel time. Formally, the objective function F reads:

$$F = \frac{1}{2} x^\top H x + x^\top h, \tag{8}$$

where $H \in R^{\dim(x)\times\dim(x)}$ is a diagonal matrix where the weight w_a is imposed on accelerations (for all $t \in \{0, \dots, N-2\}$) and the weight w_j is imposed on jerks (for all $t \in \{0, \dots, N-3\}$) while it is zero everywhere else. The vector h has zero value for all elements except for the final displacement at $t = N$ (i.e., $p(N)$), which is set to $-w_f$ ($w_f > 0$) to encourage a large displacement with respect to the initial position (i.e., a shorter travel time).

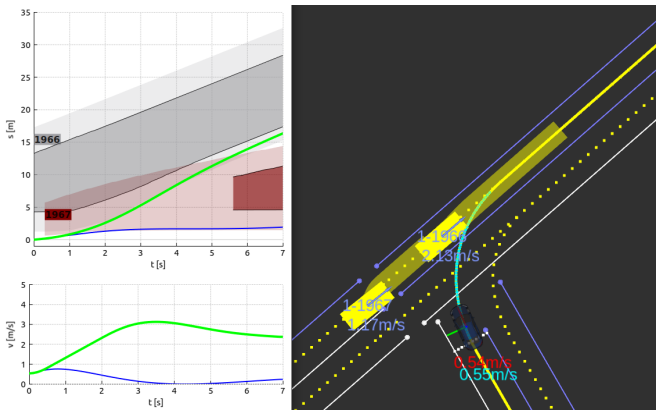

Fig. 3. T junction scenario with 2 agents (Car 1966 in grey, Car 1967 in dark red). The hard constraints can be seen in a solid color fill, the soft constraints can be seen in a semi-transparent color fill.

B. Constraints

There are two types of constraints: (i) initial conditions and (ii) lower and upper bounds. The initial conditions read:

$$p(0) = p_0, \tag{9}$$
$$v(0) = v_0 \tag{10}$$

where p_0 and v_0 denote the displacement and speed measured at the current time, respectively. In compact form, we can rewrite it with the augmented variable x, i.e., $x(0) = x_0$.

Recall that the lower and upper bounds are obtained from the ST graph in Section II-B and thus they are given and known. The corresponding inequality constraints read:

$$lb(t) \leq x(t) \leq ub(t), \quad \forall t \in \{0, \dots, N\}. \tag{11}$$

Note that the lower and upper bounds can be engineered with customized margin choices (such as parameters), which can potentially cause infeasibility issues. Details are discussed in Section III-D.

C. Complete problem

The complete optimization problem reads:

$$\min_{x} \frac{1}{2} x^\top H x + x^\top h \tag{12a}$$
subject to:
$$x(t+1) = f(x(t)), \quad \forall t \in \{0, \dots, N-1\}, \tag{12b}$$
$$lb(t) \leq x(t) \leq ub(t), \quad \forall t \in \{0, \dots, N\}, \tag{12c}$$
$$x(0) = x_0. \tag{12d}$$

Note that this is a generic form of QP without any (iterative) approximations – it retains nice convex optimization properties [20], e.g., optimality and convergence rate.

D. Soft-constraints for relaxed problem

The inequality constraints (11) can be violated depending on the constructed ST graph, leading to a problem that is infeasible to solve. Examples include: (i) two predicted agent trajectories overlapping, causing no-existing feasible throughput within the planning horizon, and (ii) the initial position of the ego vehicle being outside the feasible space defined by the two bounds (due to excessive margin parameters and/or perception/localization noises). Infeasible solutions may cause jerky and uncomfortable maneuvering (depending on the pre-defined actions for infeasible solutions).

One well-received solution is to reformulate the problem by relaxing the constraints, leaving the problem soft-constrained. We add a slack variable for each bound (i.e., s_{lb} and s_{ub}) and penalize non-zero slack variables in the objective function. The problem now reads:

$$\min_{x, s_{lb}, s_{ub}} \frac{1}{2} x^\top H x + x^\top h + w_b^\top (s_{lb} + s_{ub}) \tag{13a}$$
subject to:
$$x(t+1) = f(x(t)), \quad \forall t \in \{0, \dots, N-1\}, \tag{13b}$$
$$lb(t) - s_{lb}(t) \leq x(t) \leq ub(t) + s_{ub}(t), \quad \forall t \in \{0, \dots, N\}, \tag{13c}$$
$$x(0) = x_0, \tag{13d}$$
$$0 \leq s_{lb}(t) \leq s_{\max lb}(t), \tag{13e}$$
$$0 \leq s_{ub}(t) \leq s_{\max ub}(t), \tag{13f}$$

where w_b denotes the penalty weight on the boundary slack variables. Note that the slack variables are temporal for all t and only non-zero values will be penalized. Also note that the slack variables are bounded (by $s_{\max lb}$ and $s_{\max ub}$). The bounds on slack variables ensure the safety guarantee, restricting the slack variables being overly penalized. Figure 3 illustrates the soft margins for a simple T-Junction scenario. The soft margins enable to obtain the feasible trajectory (green line) which cannot be found with the hard margins; there is no feasible gap between the two agents.

E. Funnel Technique to Impose Lateral Acceleration Limits

One caveat of longitudinal speed planning is that it disregards lateral accelerations and associated kinematic limits associated with the given path. However, directly evaluating the lateral accelerations leads the problem to non-convex, resulting in a higher complexity problem to solve [16], [13]. We introduce an easily implementable yet effective method: the funnel technique.

Given the path, we compute the curvature K at each position step s. With a parametric lateral acceleration limit a_ℓ, the associated speed limit is:

$$v_s = \sqrt{\frac{a_\ell}{K_s}}. \tag{14}$$

With a parametric distance lookahead L, the speed limit v_ℓ is set to the minimum of v_s, i.e., $v_\ell = \min(v_s, v_{\max}), \forall s \in \{0, \dots, L\}$, where $v_{\max}$ is the traffic speed limit. Yet, the speed limit v_ℓ cannot be immediately used to upper bound the speed in QP in Eqn. (13) since the QP becomes infeasible if the current speed is higher than v_ℓ. Instead, the upper bound is funneled down from the traffic speed limit to v_ℓ by linearly decreasing with a fixed rate. Note that a high decreasing rate might overly tighten the upper bounds (which in turn restricts the feasible set). We found that half the minimum acceleration, i.e., $-\frac{a_{\min}}{2}$, is a comfortable rate.

Fig. 4. Testing scenarios (from left): (i) highway merging, (ii) T-junction, (iii) four-way intersection, and (iv) lane changing in dense traffic.

Scenario	Time	Succ. (%)	Coll. (%)	Time out (%)	Brake Avg	Thr. Avg	Acc. Max	Brake Jerk Avg	Thr. Jerk Avg	Ang. Acc. Avg	Ang. Acc. Max	Ang. Jerk Avg	Ang. Jerk Max
Highway merging	20.17	100	0	0	-0.64	0.75	1.87	-0.73	0.9	1.36	8.22	2.31	15.43
T-junction	22.18	100	0	0	-0.71	0.66	1.97	-0.86	0.85	2.44	12.26	3.46	23.45
Four-way intersection	31.34	100	0	0	-0.32	0.42	1.96	-0.47	0.45	0.97	10.59	4.24	47.36
Lane change in dense traffic	41.77	100	0	0	-0.19	0.25	1.4	-0.34	0.32	1.5	18.84	2.17	30.29

TABLE I

SIMULATION RESULTS WITH THE FOUR DIFFERENT SCENARIOS. THE RESULT OF EACH SCENARIO IS WITH 50 RUNS. THE IDENTICAL SYSTEM IS USED FOR ALL FOUR SCENARIOS, AND POSITIONS AND DRIVING BEHAVIORS (E.G., TARGET SPEED) OF THE OTHER VEHICLES ARE RANDOMLY INITIALIZED IN EACH RUN. A RUN STOPS IF A COLLISION OCCURS OR A TIME LIMIT (80 SECONDS) IS REACHED.

IV. VALIDATIONS

A. Simulation Overview

We leverage CARLA [25] for the high-fidelity simulation environment which has been broadly adopted for validating autonomous driving research. Scenarios are generated using the "scenario runner", the CARLA built-in simulation generation platform.

We test MPQP with four different scenarios: (i) highway merging, (ii) T-junction, (iii) four-way intersection, and (iv) lane changing in dense traffic, as presented in Fig. 4. Each scenario has different road structures, target lanes, and traffic participants.

Highway merging scenario: The ego vehicle starts on the ramp and merges into the lane. Other vehicles in the target lane drive fast with randomness in their yielding behaviors toward the ego vehicle.

T-junction scenario: The ego vehicle makes an unprotected left turn while maintaining safety for crossing pedestrians.

Four-way intersection scenario: The ego vehicle makes a left turn. The increased complexity is due to additional vehicles approaching from the opposite direction.

Lane changing scenario: The goal is to lane change into dense traffic to avoid an emergency vehicle stopped in front. There exists no safe space for the ego vehicle to merge without interacting with other cars (i.e., without the cooperation of other vehicles).

This set of scenarios and their inherent challenges evaluate the general performance of the proposed algorithm. Note that we use the same algorithm for all scenarios, while positions and driving behaviors (e.g., target speed) of other vehicles are randomly initialized in each run. Other vehicles are modeled as intelligent driver model (IDM) [26] with some modifications to detect the ego vehicle in the adjacent lane. The yielding behavior is triggered according to a pre-set Bernoulli parameter. Interested readers are referred to [5] for details. Overall, the vehicles are designed to be aggressive in order to challenge our algorithm. The behavior model for the agents is lane following (except for pedestrians that randomly change directions) and traffic does not change lanes.

MPQP's processing time averages 20 ms with a maximum time cycle of 84 ms on a laptop with an Intel® Core™ i9-12900H and 32 GB RAM. This processing time includes all the aforementioned steps: ST graph construction (using ClipperLib [27]), ST Cell Planning, and quadratic programming (using qpOASES [28]). While the planned trajectory is sent to an underlying controller, MPQP finds a new solution. The planning horizon was set to 10 seconds for slow-speed scenarios and 15 seconds for high-speed scenarios. This indicates a strong potential for MPQP to be applicable to real-time systems even with a relatively long planning horizon.

B. Simulation Result

We first investigate the performance of MPQP in the four-way intersection scenario aligned with the motivational example in Fig. 1. Unlike the motivational example, the ego vehicle takes a left turn to interact with unexpected pedestrian crossing. The general behaviors of MPQP were observed as: (i) the ego vehicle approaches the intersection; (ii) the ego vehicle stays stopped until the oncoming vehicles cross - Figure 5 illustrates the ego vehicle slowly cruising until the oncoming cars (labeled as 258 and 257) pass on lane 98; (iii) if the car (car 264 in Fig. 5) on the merging lane does not slow down, the ego vehicle slows down and waits. Otherwise, it proceeds. (iv) While completing the turn, a pedestrian suddenly crosses the road and MPQP waits. Depending on the initial positions and aggressiveness of the traffic, the behaviors might vary, however, MPQP consistently secured safe and successful merging.

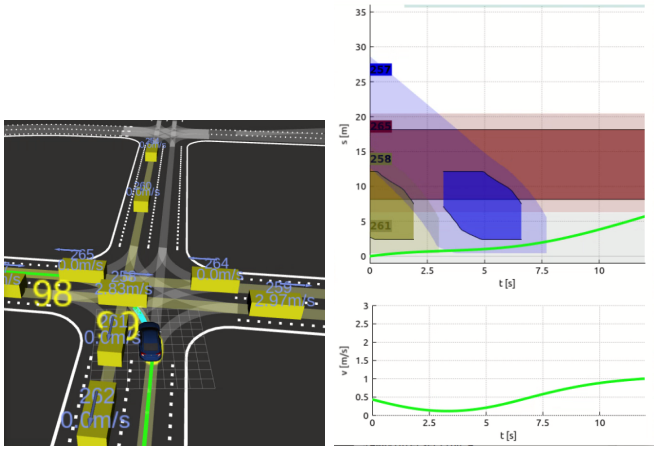

Fig. 5. Testing scenarios (from left): (i) highway merging, (ii) T-junction, (iii) four-way intersection, and (iv) lane changing in dense traffic.

Fig. 6. Public demonstration of MPQP on a closed course in Joso city, Japan. The white CR-V on the left of the image is the AD vehicle.

Now, we extend our analysis across the aforementioned four scenarios: highway merging, T-junction, four-way intersection, and lane change. Table I tabulates the overall performance for each scenario that was repeated 50 times with random initialization. Overall, MPQP successfully completes all scenarios without collisions or timeout. Note that MPQP does not have prior knowledge of what scenario will be running and of what behavior model the traffic is based upon. That is, MPQP demonstrates a capability of generalizability without situational knowledge and with the presence of uncertainties. Some observations from the results: (i) Interestingly, the T-junction scenario has a higher average on the brake than that in the highway (which has a higher speed throughput). This is explained by the uncertain behavioral changes of traffic in the T-junction scenario. Due to the unexpected acceleration of the vehicle at the merging point, the ego vehicle had to slow down strongly to keep the safety. In fact, the same behavior is observed in the four-way scenario, however, the average speed is much lower due to the density which results in less aggressive brake and jerk. (ii) The ego vehicle tends to drive smoother and slower for lane changing in dense traffic, waiting for their cooperation (i.e., making room for the ego vehicle). (iii) The ego vehicle tends to be more conservative in the T-junction and four-way scenarios as there exists priority between traffic agents. The priority is given to the pedestrians and oncoming cars (especially in the T-junction scenario). Accordingly, the ego vehicle tends to be defensive and merges to the target lane only if the car obviously gives way to the ego vehicle or if safety is guaranteed (e.g., no vehicle). The overall behaviors are aligned with an experienced human driver who balances between being defensive and not being too conservative.

C. On-road Demonstration

While the simulation effectively evaluates the general performance of MPQP, there still exists a sim-to-real gap. That is, the behavior model of other vehicles often fails to correctly represent the decision making process of other drivers. Simulation environments do not precisely represent localization and perception noises. Different controllers are used, and there are variations in the physics and system timing. We thus extended our validation in the real-world urban city course in Joso city, Japan, in July 2023. The course includes several driving scenarios, including 4 way intersection (snapshot in Fig. 6), T-junction, and lane change.

Public participants feedback was generally positive. Participants mentioned the ride was very smooth and had good user comfort. Some users said the crowded intersection felt complicated even for human drivers. Some participants said some maneuvers, like the lane change and driving around a stopped car, felt a little dangerous but expressed that they had trust in the autonomous driving system.

V. Conclusion

Inspired by the efficacy of path-speed decomposition methods, this paper provides a mathematical framework to optimally plan a speed profile for automated vehicles, namely MPQP. Given a path, we leverage the Breadth First Search algorithm to find timing combinations to follow the path when dynamic objects are present. Each profile is then integrated into Quadratic Programming as lower and upper bounds. The formulation and implementation are computationally efficient, being capable of providing a new solution within 20 ms on average for 10+ seconds of planning horizon. Simulation results in CARLA demonstrate the strong potential of MPQP, followed by a real-car demonstration with public participants. One caveat is that this work relies on deterministic predictions for traffic which precludes the ability to yield interactive behaviors. Handling uncertainties in multi-modal predictions remains for future work.

Acknowledgement

The authors would like to thank Dr. Yuji Yasui and Dr. Takayasu Kumano at Honda R&D Co., Ltd., Japan, for their general support for the research and vehicle test.

REFERENCES

[1] Lester E Dubins. On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, 79(3):497–516, 1957.

[2] James Reeds and Lawrence Shepp. Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, 145(2):367–393, 1990.

[3] Alexander Heilmeier, Alexander Wischnewski, Leonhard Hermansdorfer, Johannes Betz, Markus Lienkamp, and Boris Lohmann. Minimum curvature trajectory planning and control for an autonomous race car. *Vehicle System Dynamics*, 2019.

[4] Maxime Bouton, Alireza Nakhaei, David Isele, Kikuo Fujimura, and Mykel J Kochenderfer. Reinforcement learning with iterative reasoning for merging in dense traffic. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[5] Sangjae Bae, David Isele, Alireza Nakhaei, Peng Xu, Alexandre Miranda Añon, Chiho Choi, Kikuo Fujimura, and Scott Moura. Lane-change in dense traffic with model predictive control and neural networks. *IEEE Transactions on Control Systems Technology*, 2022.

[6] Boris Ivanovic, Amine Elhafsi, Guy Rosman, Adrien Gaidon, and Marco Pavone. Mats: An interpretable trajectory forecasting representation for planning and control. *arXiv preprint arXiv:2009.07517*, 2020.

[7] Faizan M Tariq, David Isele, John S. Baras, and Sangjae Bae. RCMS: Risk-aware crash mitigation system for autonomous vehicles. *IEEE Intelligent Vehicles Symposium*, 2023.

[8] Thierry Fraichard and Christian Laugier. Path-velocity decomposition revisited and applied to dynamic trajectory planning. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 40–45. IEEE, 1993.

[9] Piyush Gupta, David Isele, Donggun Lee, and Sangjae Bae. Interaction-aware trajectory planning for autonomous vehicles with analytic integration of neural networks into model predictive control. *2023 International Conference on Robotics and Automation (ICRA)*, 2023.

[10] Anahita Mohseni-Kabir, David Isele, and Kikuo Fujimura. Interaction-aware multi-agent reinforcement learning for mobile agents with individual goals. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 3370–3376. IEEE, 2019.

[11] Kamal Kant and Steven W Zucker. Toward efficient trajectory planning: The path-velocity decomposition. *The international journal of robotics research*, 5(3):72–89, 1986.

[12] Quang-Cuong Pham, Stéphane Caron, Puttichai Lertkultanon, and Yoshihiko Nakamura. Planning truly dynamic motions: Path-velocity decomposition revisited. *arXiv preprint arXiv:1411.4045*, 2014.

[13] Changliu Liu, Wei Zhan, and Masayoshi Tomizuka. Speed profile planning in dynamic environments via temporal optimization. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 154–159, 2017.

[14] Zeyu Yang, Jin Huang, Hui Yin, Diange Yang, and Zhihua Zhong. Path tracking control for underactuated vehicles with matched-mismatched uncertainties: An uncertainty decomposition based constraint-following approach. *IEEE Transactions on Intelligent Transportation Systems*, 23(8):12894–12907, 2021.

[15] Vasundhara Jain, Uli Kolbe, Gabi Breuel, and Christoph Stiller. Collision avoidance for multiple static obstacles using path-velocity decomposition. *IFAC-PapersOnLine*, 52(8):265–270, 2019.

[16] Wenda Xu. *Motion Planning for Autonomous Vehicles in Urban Scenarios: A Sequential Optimization Approach*. PhD thesis, Carnegie Mellon University, 02 2022.

[17] Till-Julius Krüger, Daniel Göhring, and Fritz Ulbrich. *Graph-Based Speed Planning for Autonomous Driving*. PhD thesis, Free University of Berlin Berlin, Germany, 2019.

[18] Christoforos Mavrogiannis, Jonathan A DeCastro, and Siddhartha S Srinivasa. Implicit multiagent coordination at unsignalized intersections via multimodal inference enabled by topological braids. *arXiv preprint arXiv:2004.05205*, 2020.

[19] Francesco Micheli, Mattia Bersani, Stefano Arrigoni, Francesco Braghin, and Federico Cheli. Nmpc trajectory planner for urban autonomous driving. *Vehicle system dynamics*, 61(5):1387–1409, 2023.

[20] Stephen P Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004.

[21] Chao Sun, Jacopo Guanetti, Francesco Borrelli, and Scott J Moura. Optimal eco-driving control of connected and autonomous vehicles through signalized intersections. *IEEE Internet of Things Journal*, 7(5):3759–3773, 2020.

[22] Xiaowei Shi and Xiaopeng Li. Trajectory planning for an autonomous vehicle with conflicting moving objects along a fixed path–an exact solution method. *Transportation Research Part B: Methodological*, 173:228–246, 2023.

[23] Moritz Werling, Julius Ziegler, Sören Kammel, and Sebastian Thrun. Optimal trajectory generation for dynamic street scenarios in a frenet frame. In *2010 IEEE international conference on robotics and automation*, pages 987–993. IEEE, 2010.

[24] Alan Bundy and Lincoln Wallen. Breadth-first search. *Catalogue of artificial intelligence tools*, pages 13–13, 1984.

[25] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[26] Arne Kesting, Martin Treiber, and Dirk Helbing. Enhanced intelligent driver model to access the impact of driving strategies on traffic capacity. *Philosophical Transactions of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 368(1928):4585–4605, 2010.

[27] Clipper2 - Polygon Clipping and Offsetting Library — angusj.com. http://www.angusj.com/clipper2/Docs/Overview.htm. [Accessed 27-Jun-2023].

[28] Hans Joachim Ferreau, Christian Kirches, Andreas Potschka, Hans Georg Bock, and Moritz Diehl. qpoases: A parametric active-set algorithm for quadratic programming. *Mathematical Programming Computation*, 6:327–363, 2014.