Real-Time Mixed-Integer Quadratic Programming for Vehicle Decision-Making and Motion Planning

Rien Quirynen, Sleiman Safaoui, *Member, IEEE*, and Stefano Di Cairano, *Senior Member, IEEE*

***Abstract*— We develop a real-time feasible mixed-integer programming-based decision-making (MIP-DM) system for automated driving (AD). Using a linear vehicle model in a road-aligned coordinate frame, the lane change constraints, collision avoidance, and traffic rules can be formulated as mixed-integer inequalities, resulting in a mixed-integer quadratic program (MIQP). The proposed MIP-DM performs maneuver selection and trajectory generation by solving the MIQP at each sampling instant. While solving MIQPs in real time has been considered intractable in the past, we show that our recently developed solver `BB-ASIPM` is capable of solving MIP-DM problems on embedded hardware in real time. The performance of this approach is illustrated in simulations in various scenarios, including merging points and traffic intersections, and hardware-in-the-loop (HIL) simulations in dSPACE Scalexio and MicroAutoBox-III (MABX-III). Finally, we show experiments using small-scale vehicles.**

***Index Terms*— Autonomous driving, decision-making, mixed-integer programming, motion planning, predictive control.**

I. Introduction

AUTOMATED transportation systems, even in the case of partial automation, may lead to reduced road accidents and more efficient usage of the road network. However, the complexity of automated driving (AD) and advanced driver-assistance systems (ADASs) and their real-time requirements in resource-limited automotive platforms [1] requires the implementation of a multilayer guidance and control architecture [2], [3]. Thus, the ADAS/AD system consists of multiple interconnected components executing at different sampling rates, where the integrated system must satisfy the overall driving specifications [4], [5].

A typical guidance and control architecture is illustrated in Fig. 1(a), see [6], [7]. Based on a route given by a navigation system, a decision-making module decides when to perform maneuvers such as lane changing, stopping, waiting, and intersection crossing. Given these decisions, a motion planning system generates a state trajectory to execute the maneuvers, and a vehicle control system computes the commands to track the trajectory.

Manuscript received 27 January 2024; revised 3 June 2024; accepted 31 July 2024. Date of publication 6 September 2024; date of current version 31 December 2024. Recommended by Associate Editor E. Hellstrom. *(Corresponding author: Stefano Di Cairano.)*

Rien Quirynen and Sleiman Safaoui were with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: rien.quirynen@gmail.com; snsafaoui@gmail.com).

Stefano Di Cairano is with Mitsubishi Electric Research Laboratories, Cambridge, MA 02139 USA (e-mail: dicairano@ieee.org).

Digital Object Identifier 10.1109/TCST.2024.3449703

Optimization-based motion planning and control techniques, such as model predictive control (MPC), account for dynamics, constraints, and objectives in a model-based design framework [8]. Leveraging hybrid system models, including both discrete and continuous decision variables [9], hybrid MPC can tackle a large range of problems, including switching control [10], obstacle avoidance [11], logic rules, and temporal logic specifications [5]. However, the mixed-integer optimal control problem (MIOCP) to be solved at each step is nonconvex and $\mathcal{NP}$-hard [12]. For a linear-quadratic objective, linear or piecewise linear dynamics, and inequality constraints, the MIOCP results in a mixed-integer quadratic program (MIQP).

Recent work [13] indicates that, by exploiting the particular structure of the MIOCPs, real-time solvers can achieve performance comparable to commercial tools, e.g., `GUROBI` [14] and `MOSEK` [15], especially for small-to-medium-scale problems. Therefore, we apply the tailored `BB-ASIPM` solver [13] that uses a branch-and-bound (B&B) method with reliability branching and warm starting [16], block-sparse presolve techniques [13], and early termination and infeasibility detection [17] within a fast convex quadratic programming (QP) solver based on an active-set interior point method (`ASIPM`) [18].

In this article, we design a mixed-integer programming-based decision-making (MIP-DM) module that simultaneously computes a sequence of discrete decisions and a continuous motion trajectory for the vehicle in a hybrid MPC framework. This approach eliminates the need for a separate motion planner in the ADAS/AD architecture as long as an advanced vehicle control algorithm is used, e.g., based on nonlinear MPC (NMPC), see Fig. 1(b). We demonstrate the proposed MIP-DM approach in simulations in various scenarios, including merging points and traffic intersections, and we confirm its real-time feasibility in dSPACE Scalexio and MicroAutoBox-III (MABX-III) rapid prototyping units commonly used for automotive development. Finally, we present results from hardware experiments using MIP-DM in combination with NMPC-based reference tracking on a setup with small-scale automated vehicles.

A. Relation With Existing Literature

In the DARPA Urban Challenge [19], most teams implemented rule-based decision-making systems involving hand-tuned heuristics for different urban-driving scenarios. Some recent works on vehicle decision-making are based on

1063-6536 © 2024 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

machine learning [20], [21], [22], which often lacks interpretability and may lack safety guarantees without additional safety layers [23]. Ahn et al. [7] proposes automata combined with set reachability; however, it aims to determine only maneuver feasibility, not accounting for performance. Esterle et al. [24] proposes simultaneous trajectory generation and maneuver selection, but its complexity grows rapidly with the number of obstacles.

Our prior work [5] defined traffic rules as signal temporal logic (STL) formula that are converted into mixed-integer inequalities resulting in MIQPs. This provides formal guarantees at the price of an optimization problem often too large for real-time implementation, in part due to the automated STL formula translation. Motivated by the latter results, this present article proposes a real-time feasible MIQP formulation for vehicle decision-making and motion planning. An overview on MIP-based decision-making, motion planning, and control problems may be found in [25] and [26]. For ADAS/AD systems, Ballesteros-Tolosana et al. [27] and Miller et al. [28] propose MIPs for vehicle lane changing and overtaking maneuvers, e.g., solving separate optimization problems for longitudinal and lateral motion planning. In [29] and [30], motion planning in Cartesian coordinates is achieved by MIQP using disjunctive region-based linearization.

B. Contributions

This article provides three main contributions. First, we present a guidance and control architecture that integrates the MIP-DM with a reference tracking NMPC controller. The MIP-DM provides a safe trajectory based on a simplified vehicle model in the MIQP formulation, which is used for efficient computations in NMPC by sequential QP (SQP). Second, we present simulation results of MIP-DM with our embeddable `BB-ASIPM` solver in a range of traffic scenarios while operating in a dynamic environment with potentially changing traffic rules and compare against state-of-the-art software tools `GUROBI` and `MOSEK`. We also validate the method in experiments using a mini car. Third, we demonstrate the real-time feasibility of the proposed approach in rapid prototyping hardware (dSPACE Scalexio and MABX-III) and present experimental results using small-scale automated vehicles. To the best of our knowledge, this article presents the first MIP for decision-making with an embedded solver that is real-time feasible in hardware-in-the-loop (HIL) simulations and vehicle experiments.

C. Outline and Notation

This article is structured as follows. Section II introduces the objectives and problem formulation, followed by a description of the MIP-DM in Section III. The MIQP solver is described in Section IV, and the integration with NMPC is discussed in Section V. The simulation results are shown in Section VI. Finally, Section VII presents the results from the hardware experiments and our conclusions are established in Section VIII.

Notation: $\mathbb{R}$, $\mathbb{R}_+$, and $\mathbb{R}_{0+}$ ($\mathbb{Z}$, $\mathbb{Z}_+$, and $\mathbb{Z}_{0+}$) are the set of real, positive real, and nonnegative real (integer) numbers, respectively; $\mathbb{B} = \{0, 1\}$; and $\mathbb{Z}_a^b = \{a, a+1, \ldots, b-1, b\}$. The logical operators AND, OR, XOR, and NOT are $\wedge$, $\vee$, $\veebar$, and $\neg$, respectively; and the logical operators *implies* and *equivalent (if and only if)* are $\implies$ and $\iff$, respectively. Inequalities between vectors are intended componentwise.

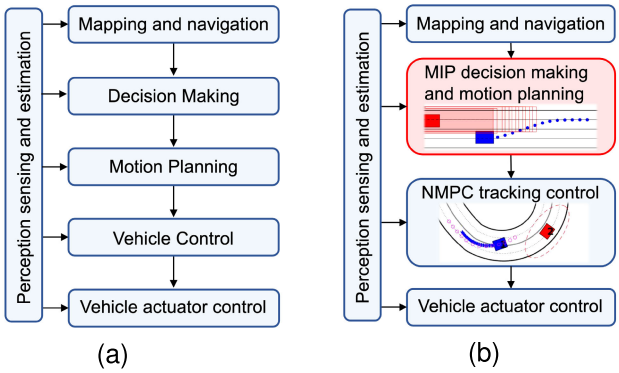


Fig. 1. Multilayer control architecture for ADAS/AD. (a) Typical architecture, e.g., [7]. (b) MIP-DM architecture.

II. Problem Setup and Formulation

This section briefly describes common components in a multilayer guidance and control architecture for ADAS/AD and then introduces the MIOCP formulation for MIP-DM.

A. Multilayer Control Architecture for AD

A typical guidance and control architecture is illustrated in Fig. 1(a). A perception, sensing, and estimation module uses various onboard sensor information, such as radar, light detection and ranging (LiDAR), camera, and global positioning system (GPS) information, to estimate the vehicle states, parameters, and parts of the surroundings relevant to the driving scenario [31]. Based on a route given by a navigation system, a decision-making module determines what maneuvers to perform, e.g., lane changing, stopping, waiting, and intersection crossing [7]. Then, a motion planning system generates a collision-free and kinematically feasible trajectory to perform the maneuvers, see [32]. A vehicle control system computes the commands to execute the motion planning trajectory, see [33]. Additional low-level controllers operate the vehicle actuators.

B. Architecture for MIP Decision-Making

In this article, an autonomous vehicle must reach a desired destination while obeying the traffic rules. This requires the vehicle to adjust its velocity to comply with speed limits, to avoid collisions, to follow and change lanes, and to cross intersections following right-of-way rules. We propose an alternative architecture to that in Fig. 1(a) using MIP-based vehicle decision-making, see Fig. 1(b). In this architecture, the MIP-based decision-making determines the sequence of next maneuvers and a coarse reference trajectory that satisfies traffic rules and avoids obstacles along a future horizon, and the NMPC tracks such a trajectory while avoiding collisions, with a shorter horizon and a higher control rate to compensate for model errors, disturbances, and sensing and prediction errors on other vehicles.

A detailed comparison of the architectures in Fig. 1 may be difficult and beyond the scope of this article. Here, we aim at

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

demonstrating the feasibility of the architecture in Fig. 1(b) and briefly discuss some potential benefits next.

The typical architecture in Fig. 1(a) separates decision-making and motion planning and computes trajectories for each selected sequence of maneuvers. Conservative approaches may only select a single maneuver, while computationally expensive approaches may generate trajectories for every sequence of maneuvers. For motion planning in Fig. 1(a), since decision-making may only provide maneuvers and not reference trajectories, using optimization-based methods may be hard, due to the nonconvexity and the lack of a good initial guess. Instead, the architecture based on MIP-DM in Fig. 1(b) integrates decision-making and motion planning. The maneuver sequences are generated in the B&B method of the MIP solver, and although the worst case complexity is still combinatorial, B&B and integer presolving techniques provide a computationally efficient way of searching for feasible and optimal solutions [34]. Such a solution provides a coarse reference trajectory and an action sequence that may be a suitable initial guess for optimization-based control involving nonconvex problems.

From an implementation perspective, fewer layers usually reduce the complexity of the software architecture, inter-module communication, and potential negative side effects. In Fig. 1(b), the main two layers are optimization-based, and large portions of code and libraries may be shared. On the other hand, the MIP-DM may treat obstacles in a more computationally burdensome way than other methods, i.e., with additional integer variables, and mixed-integer solvers, no matter how simplified, are typically more complex than the rule- or graph-based decision-makers and well-known motion planning methods, e.g., sampling based. Thus, the final choice has to be based on the application specifications and computational platform, but our results demonstrate that the architecture in Fig. 1(b) may be a valid alternative to that in Fig. 1(a).

The problem setup in this work requires the following simplifying assumptions.

Assumption 1: There exists a prediction time window along which the following is known:

1) the position and orientation for each of the obstacles in a sufficiently large neighborhood of the ego vehicle;
2) the map information, including center lines, road curvature, and lane widths within the current road segment;
3) the current traffic rules and any changes to the rules, e.g., traffic light timings and/or speed zone changes. □

Assumption 1-1 requires the ego vehicle to be equipped with sensors to detect static and dynamic obstacles within a given range and to locate itself in the environment. Furthermore, the ego vehicle must be equipped with a module that provides conservative predictions for future trajectories of the dynamic obstacles, e.g., using techniques referenced in [2] and [3]. Such predictions are not expected to be exact since the feedback mechanism of the architecture proposed here will adjust the trajectory, in the presence of prediction errors. Assumption 1-2 requires the availability of map information and/or the use of online updates and corrections to such map information [31]. Assumption 1-3 requires a combination of map information, online perception [31], and/or vehicle-to-infrastructure (V2I) communication [35]. Based on these assumptions, we define the problem statement and objectives.

Definition 1 (MIP-DM): Under Assumption 1 and for given navigation information, at each sampling instant, the MIP-DM module solves an MIOCP on embedded hardware and under strict timing requirements. The solution provides desired maneuvers that the vehicle should execute, and a coarse trajectory, i.e., a sequence of waypoints and target velocities, over a horizon of several seconds for the vehicle control module to execute the maneuver. □

C. Integration of MIP-DM With NMPC Tracking Controller

Based on Definition 1, the MIP-DM computes a safe trajectory that is executed by the NMPC tracking controller in Fig. 1(b), i.e., the vehicle control module. The MIP-DM trajectory is used in NMPC as an effective warm start for the nonconvex optimization problem, e.g., using SQP with the real-time iteration (RTI) [36]. The MIP-DM plans for long horizon trajectories with coarse granularity by solving a nonconvex optimization problem formulated on a simplified linear vehicle model, with a long sampling time period, $T_{\mathrm{s}}^{\mathrm{mip}} \approx 0.4$–$2$ s, and a relatively long prediction horizon, $T^{\mathrm{mip}} \approx 10$–$20$ s. The NMPC tracks such trajectories with finer granularity while enforcing constraints, by solving a convex problem formulated on a more accurate nonlinear kinematic model, with a short sampling time period $T_{\mathrm{s}}^{\mathrm{mpc}} \approx 0.02$–$0.1$ s, and a relatively short prediction horizon, $T^{\mathrm{mpc}} \approx 2$–$4$ s. Thus, the NMPC compensates for MIP-DM prediction model errors and disturbances, corrects possible MIP-DM intersampling constraint violations, and provides faster reactions to sensing and prediction errors of other vehicles, thus improving the overall collision avoidance.

D. Mixed-Integer Optimal Control Problem

At each sampling time instant, the proposed MIP-DM solves the following MIOCP:

$$\min_{X,\,U} \sum_{i=0}^{N} \frac{1}{2}\begin{bmatrix} x(i) \\ u(i) \end{bmatrix}^{\top} H(i) \begin{bmatrix} x(i) \\ u(i) \end{bmatrix} + \begin{bmatrix} q(i) \\ r(i) \end{bmatrix}^{\top} \begin{bmatrix} x(i) \\ u(i) \end{bmatrix} \tag{1a}$$

$$\text{s.t.} \quad x(i+1) = \begin{bmatrix} A(i)\, B(i) \end{bmatrix} \begin{bmatrix} x(i) \\ u(i) \end{bmatrix} + a(i), \quad \forall i \in \mathbb{Z}_0^{N-1} \tag{1b}$$

$$\begin{bmatrix} \underline{x}(i) \\ \underline{u}(i) \end{bmatrix} \leq \begin{bmatrix} x(i) \\ u(i) \end{bmatrix} \leq \begin{bmatrix} \overline{x}(i) \\ \overline{u}(i) \end{bmatrix}, \quad \forall i \in \mathbb{Z}_0^{N} \tag{1c}$$

$$\underline{c}(i) \leq \begin{bmatrix} C(i)\, D(i) \end{bmatrix} \begin{bmatrix} x(i) \\ u(i) \end{bmatrix} \leq \overline{c}(i), \quad \forall i \in \mathbb{Z}_0^{N} \tag{1d}$$

$$u_j(i) \in \mathbb{Z}, \quad \forall j \in \mathcal{I}(i), \quad \forall i \in \mathbb{Z}_0^{N} \tag{1e}$$

where $i \in \{0, 1, \ldots, N\}$ is the time, N is the horizon length, the state variables are $x(i) \in \mathbb{R}^{n_{\mathrm{x}}^i}$, the control and auxiliary variables are $u(i) \in \mathbb{R}^{n_{\mathrm{u}}^i}$, and $\mathcal{I}(i)$ denotes the index set of integer decision variables $u_j(i)$, $\forall j \in \mathcal{I}(i)$, $\forall i \in \mathbb{Z}_0^N$, i.e., the cardinality $|\mathcal{I}(i)| \leq n_{\mathrm{u}}^i$ denotes the number of control and auxiliary variables that are integer valued. The objective in (1a) defines a linear-quadratic function with positive semi-definite Hessian matrix $H(i) \succeq 0$ and gradient vectors $q(i) \in \mathbb{R}^{n_{\mathrm{x}}^i}$ and $r(i) \in \mathbb{R}^{n_{\mathrm{u}}^i}$. The constraints include dynamic constraints in (1b), simple bounds in (1c), affine inequalities in (1d),

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

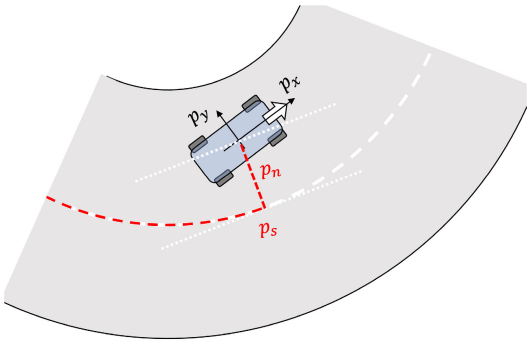


Fig. 2. Road-aligned curvilinear coordinate system for a curved segment; p_s is the arc length along the center line and p_n is the lateral deviation.

and integer feasibility constraints in (1e). The initial state constraint $x(0) = \hat{x}_t$, where $\hat{x}_t$ is a current state estimate at time t, can be enforced using the simple bounds in (1c). The MIOCP (1) includes control variables on the terminal stage, $u(N) \in \mathbb{R}^{n_u^N}$, due to possibly needing auxiliary variables to formulate the mixed-integer inequalities. A binary optimization variable $u_j(i) \in \{0, 1\}$ can be defined as an integer variable $u_j(i) \in \mathbb{Z}$ in (1e), including the simple bounds $0 \le u_j(i) \le 1$ in (1c). For compactness, we denote $X = [x(0)^\top, \dots, x(N)^\top]^\top$ and $U = [u(0)^\top, \dots, u(N)^\top]^\top$. The MIOCP (1) can be reformulated as a block-sparse structured MIQP [13] and solved with the corresponding algorithms.

III. Mixed-Integer Quadratic Programming for Vehicle Decision-Making and Motion Planning

Next, we describe the MIP-DM for real-time feasible AD in real-world scenarios. Traffic rules and switching dynamics are formulated using logical operators that can be implemented using mixed-integer inequalities [37].

A. Linear Vehicle Model in Road-Aligned Frame

The curvilinear coordinate system used in the prediction model of the MIOCP (1) is shown in Fig. 2. A similar coordinate system has been used for predictive control, e.g., [38], [39]. The vehicle position is described by (p_s, p_n), where p_s denotes the progress along the center line of the lane in which the ego vehicle is driving and p_n denotes the normal distance of the vehicle position from the center line.

Assumption 2: The turning radius is much larger than the wheelbase of the vehicle such that the steering and slip angles are relatively small and their difference for the outside and inside wheels is negligible. □

Based on Assumption 2, which is common in vehicle motion planning [6], [32], we use a simplified linear vehicle model in the curvilinear coordinate system with decoupled longitudinal and lateral kinematics

$$\begin{aligned} p_s(i+1) &= p_s(i) + T_s\, v_s(i) \\ v_s(i+1) &= v_s(i) + T_s\, a_s(i) \\ p_n(i+1) &= p_n(i) + T_s\, v_n(i) \end{aligned} \tag{2}$$

where the control inputs are the longitudinal acceleration $a_s(i)$ and the lateral velocity $v_n(i)$ at each time step $i \in \mathbb{Z}_0^{N-1}$. To approximate the nonholonomic constraints for vehicles, similar to [40], we enforce the linear inequalities on the lateral and longitudinal velocities

$$-\alpha\, v_s(i) \le v_n(i) \le \alpha\, v_s(i), \quad i \in \mathbb{Z}_0^{N-1} \tag{3}$$

where $\alpha > 0$, and we assume that $v_s(i) \ge 0$ at all time steps.

Proposition 1: The constraint in (3) is a linear approximation of a vehicle steering limit using a kinematic bicycle model and given the vehicle's minimum turning radius $R^{\min}$

$$\alpha = \sin\left(\tan^{-1}\left(\frac{l_r}{R^{\min}}\right)\right) \approx \frac{l_r}{R^{\min}} \tag{4}$$

where l_r is the distance from the center of gravity to the rear axle.

Proof: Considering the kinematic bicycle model [41]

$$\dot{p}_X = v\cos(\psi+\beta), \quad \dot{p}_Y = v\sin(\psi+\beta) \tag{5a}$$

$$\dot{\psi} = v\,\frac{\cos(\beta)}{L}\tan(\delta), \quad \beta = \tan^{-1}\left(\frac{l_r\tan(\delta)}{L}\right) \tag{5b}$$

where (p_X, p_Y) is the position of the vehicle's center of gravity in an absolute frame and $L = l_f + l_r$ is the wheelbase. For a constant radius R or road curvature $(1/R)$, the yaw rate is $\dot{\psi} = (v/R)$ [41, Sec. 2.2] such that $\tan(\delta) \approx (L/R)$ and $\beta = \tan^{-1}(l_r/R)$. The lateral velocity in the car body frame is $\dot{p}_y = v\sin(\beta)$. Given a minimum turning radius $R^{\min} > 0$, the steady-state lateral velocity is $v_y^{\max} = v\sin(\tan^{-1}(l_r/R^{\min}))$, and therefore, in (3), $\alpha_R^{\max} = \sin(\tan^{-1}(l_r/R^{\min})) \approx (l_r/R^{\min}) > 0$. □

The vehicle model (2) is an approximation of more precise models, see [42], which are usually nonlinear. Additional constraints on the lateral acceleration may be included but are omitted in this article for simplicity. The MIP-DM (1) operates in normal driving conditions, and in those conditions, some vehicle nonlinearities, such as those in the road–tire friction curve, are not excited, while others can be neglected because of the long horizon and coarse sampling period. In addition, errors due to the simplified model in MIP-DM are compensated by the NMPC tracking control layer in Fig. 1(b).

Remark 1: Given a time-varying road radius $R(i)$, which may be positive or negative depending on the direction of the road curvature, the lateral velocity in (2) is bounded as $v_n(i) \le v_y^{\max} - v_y^R(i)$, where $v_y^{\max} = v\,\alpha_R^{\max}$ and $v_y^R(i) = v\,\alpha_R(i)$ denotes the steady-state lateral velocity to follow the center of the road with radius $R(i)$. Equation (3) may be replaced by

$$\big(-\alpha_R^{\max} - \alpha_R(i)\big)v_s(i) \le v_n(i) \le \big(\alpha_R^{\max} - \alpha_R(i)\big)v_s(i) \tag{6}$$

where $\alpha_R^{\max} = (l_r/R^{\min}) > 0$ is the maximum steering and $\alpha_R(i) = (l_r/R(i))$ is the steering needed to follow the center of the road with radius $R(i)$, see Proposition 1. □

Remark 2: The vehicle model in (2) and (3) is simplified and conservative so that the NMPC can track the MIP-DM reference by leveraging its higher control rate while using a more precise vehicle model. As a consequence, the solutions obtained by MIP-DM may not be optimal among all vehicle trajectories. However, for AD in everyday conditions and general roads, optimality is not a key requirement, as safety, consistent and explainable operation, robustness, and simple and efficient computations are favored. The simplifications developed here trade the former for the latter ones.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

B. Lane Change and Timing Delay Constraints

For proper driving, we enforce lane bound constraints

$$-\frac{w_\mathrm{l}}{2} \leq p_\mathrm{n}(i) - p_\mathrm{n}^\mathrm{ref}(i) \leq \frac{w_\mathrm{l}}{2}, \quad i \in \mathbb{Z}_0^N \tag{7}$$

where w_l denotes a lane width given by the map and $p_\mathrm{n}^\mathrm{ref} \in \mathbb{R}$ is an auxiliary state variable that denotes the lateral position of the center line of the current lane of the vehicle. For equal lane width values w_l, the vehicle is in lane j if $p_\mathrm{n}^\mathrm{ref} = (j-1)\, w_\mathrm{l}$ for $j \in \{1, \ldots, n_\mathrm{l}\}$, where n_l is the number of lanes in the current traffic scenario. Even though the reference lane value may jump from one time step $p_\mathrm{n}^\mathrm{ref}(i)$ to the next $p_\mathrm{n}^\mathrm{ref}(i+1)$, it may take multiple time steps for the lateral position to transition from the center line of one lane to the next, i.e., $p_\mathrm{n}(i-l) \approx p_\mathrm{n}^\mathrm{ref}(i)$ and $p_\mathrm{n}(i+k) \approx p_\mathrm{n}^\mathrm{ref}(i+1)$, where $l \geq 0$ and $k \geq 1$.

1) Lane Change Decision Constraints: We use two binary variables $b_\mathrm{c}^\mathrm{u}(i), b_\mathrm{c}^\mathrm{d}(i) \in \{0, 1\}$ that denote whether the vehicle performs a lane change left or right, respectively, at time step $i \in \mathbb{Z}_0^{N-1}$. We also introduce an auxiliary variable $\Delta_\mathrm{c}(i) \in \mathbb{R}$ defined by $b_\mathrm{c}^\mathrm{u}(i)$ and $b_\mathrm{c}^\mathrm{d}(i)$ for $i \in \mathbb{Z}_0^{N-1}$ through

$$\begin{aligned} b_\mathrm{c}^\mathrm{u}(i) = 1 &\implies \Delta_\mathrm{c}(i) = w_\mathrm{l} \ \wedge \ b_\mathrm{c}^\mathrm{d}(i) = 0 \\ b_\mathrm{c}^\mathrm{d}(i) = 1 &\implies \Delta_\mathrm{c}(i) = -w_\mathrm{l} \ \wedge \ b_\mathrm{c}^\mathrm{u}(i) = 0 \\ b_\mathrm{c}^\mathrm{u}(i) = 0 \ \wedge \ b_\mathrm{c}^\mathrm{d}(i) = 0 &\implies \Delta_\mathrm{c}(i) = 0. \end{aligned} \tag{8}$$

The implications in (8) are implemented using mixed-integer inequality constraints, e.g., by the big-M technique [34]. The auxiliary state dynamics are

$$p_\mathrm{n}^\mathrm{ref}(i+1) = p_\mathrm{n}^\mathrm{ref}(i) + \Delta_\mathrm{c}(i) \tag{9a}$$

$$n_\mathrm{LC}(i+1) = n_\mathrm{LC}(i) + (b_\mathrm{c}^\mathrm{u}(i) + b_\mathrm{c}^\mathrm{d}(i)) \tag{9b}$$

where $n_\mathrm{LC}(i)$ counts the number of lane changes over the prediction horizon and is initialized to $n_\mathrm{LC}(0) = 0$.

Remark 3: The state $n_\mathrm{LC}(i) \in \mathbb{Z}$ is an integer variable, but it can be relaxed to be continuous because the sum in (9b) is guaranteed to be integer. Similarly, $p_\mathrm{n}^\mathrm{ref}$ and Δ_c could be reformulated as $p_\mathrm{n}^\mathrm{ref} = w_\mathrm{l}\, \tilde{p}_\mathrm{n}^\mathrm{ref}$ and $\Delta_\mathrm{c} = w_\mathrm{l}\, \tilde{\Delta}_\mathrm{c}$, where $\tilde{p}_\mathrm{n}^\mathrm{ref} \in \{0, 1, \ldots, n_\mathrm{l} - 1\}$ and $\tilde{\Delta}_\mathrm{c} \in \{-1, 0, 1\}$. State-of-the-art MIP solvers can possibly use these integer feasibility constraints to reduce the computational effort [34]. For simplicity, we only use continuous and binary optimization variables. □

2) Timing Delay Constraints for Lane Changes: We enforce a minimum time delay of t_min between two consecutive lane changes. The lane change variables $b_\mathrm{c}^\mathrm{u}(i), b_\mathrm{c}^\mathrm{d}(i) \in \{0, 1\}$ reset a timer $t_\mathrm{c}(i)$ as

$$t_\mathrm{c}(i+1) = \begin{cases} t_\mathrm{c}(i) + T_\mathrm{s}, & \text{if } b_\mathrm{c}^\mathrm{u}(i) = b_\mathrm{c}^\mathrm{d}(i) = 0 \\ 0, & \text{otherwise} \end{cases} \tag{10}$$

which can be implemented by mixed-integer inequality constraints, e.g., using again the big-M technique. Given $t_\mathrm{c}(i)$, we impose a minimum time between lane changes

$$b_\mathrm{c}^\mathrm{u}(i) = 1 \ \vee \ b_\mathrm{c}^\mathrm{d}(i) = 1 \implies t_\mathrm{c}(i) \geq t_\mathrm{min}, \ i \in \mathbb{Z}_0^{N-1}. \tag{11}$$

C. Polyhedral Obstacle Avoidance Constraints

The MIP-DM enforces obstacle avoidance constraints to avoid a region of collision risk around other traffic participants, e.g., vehicles, bicycles, or pedestrians. The position and dimensions of the safety region may be time-varying and adapted to a prediction of the behavior for each of the traffic participants. In addition, obstacle avoidance constraints enforce stopping maneuvers, e.g., in case of a stop sign or a red traffic light at an intersection. Per Assumption 1, the prediction of obstacle motions, the map information and the traffic rules are known. For simplicity, we use axis-aligned rectangular collision regions, see Fig. 3. Alternatively, any polyhedral representation of the collision regions could be used, see [25]. The size of the collision region around the obstacle is increased with the geometric shape of the ego vehicle and includes an additional safety margin for robustness to discretization errors, model mismatch, and/or disturbances.

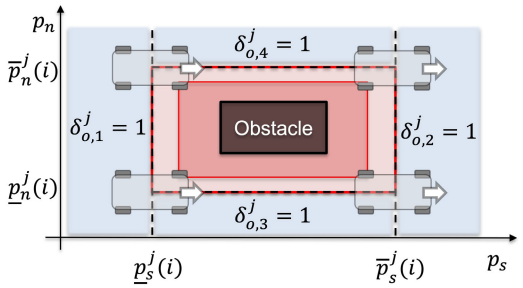


Fig. 3. Obstacle avoidance constraints using binary variables and an axis-aligned rectangular collision region. The extent of the region is increased by the geometric shape of the ego vehicle and includes an additional safety margin. The light red shaded region is enforced as soft constraints, while the dark region is enforced as hard constraints.

As shown in Fig. 3, obstacle avoidance for an axis-aligned rectangular region results in four disjoint feasible sets. We introduce four auxiliary binary variables $b_\mathrm{o}^j(i) = [b_{\mathrm{o},k}^j(i)]_{k \in \mathbb{Z}_1^4}$ for $j \in \mathbb{Z}_1^{n_\mathrm{obs}}$ to implement the logical implications

$$\begin{aligned} b_{\mathrm{o},1}^j = 1 &\iff p_\mathrm{s} \leq \underline{p}_\mathrm{s}^j - t_\mathrm{safe} v_\mathrm{s} + \nu_\mathrm{s}^\mathrm{c} \\ b_{\mathrm{o},2}^j = 1 &\iff p_\mathrm{s} \geq \overline{p}_\mathrm{s}^j + t_\mathrm{safe} v_\mathrm{s} - \nu_\mathrm{s}^\mathrm{c} \\ b_{\mathrm{o},3}^j = 1 &\iff \underline{p}_\mathrm{s}^j - t_\mathrm{safe} v_\mathrm{s} + \nu_\mathrm{s}^\mathrm{c} \leq p_\mathrm{s} \leq \overline{p}_\mathrm{s}^j + t_\mathrm{safe} v_\mathrm{s} - \nu_\mathrm{s}^\mathrm{c} \\ &\quad \wedge \ p_\mathrm{n} \leq \underline{p}_\mathrm{n}^j + \nu_\mathrm{n}^\mathrm{c} \\ b_{\mathrm{o},4}^j = 1 &\iff \underline{p}_\mathrm{s}^j - t_\mathrm{safe} v_\mathrm{s} + \nu_\mathrm{s}^\mathrm{c} \leq p_\mathrm{s} \leq \overline{p}_\mathrm{s}^j + t_\mathrm{safe} v_\mathrm{s} - \nu_\mathrm{s}^\mathrm{c} \\ &\quad \wedge \ p_\mathrm{n} \geq \overline{p}_\mathrm{n}^j - \nu_\mathrm{n}^\mathrm{c} \end{aligned} \tag{12}$$

where we omit the index $i \in \mathbb{Z}_0^N$ for readability and use slack variables $\nu_\mathrm{s}^\mathrm{c}(i) \geq 0$ and $\nu_\mathrm{n}^\mathrm{c}(i) \geq 0$ to ensure feasibility. In addition, $t_\mathrm{safe} v_\mathrm{s}$ corresponds to an adaptive velocity-dependent safety margin, where t_safe is the safety distance in seconds, where in general, it is recommended $t_\mathrm{safe} \approx 2$ s, and v_s is the longitudinal velocity of the ego vehicle, as predicted in the optimal control problem. Alternatively, v_s may be fixed to the current value and kept constant along the prediction horizon, for simplicity. We impose that the ego vehicle is in one of the feasible sets by $\sum_{k=1}^4 b_{\mathrm{o},k}^j(i) = 1$. Hard obstacle avoidance constraints can be defined by enforcing upper bounds on the slack variables $0 \leq \nu_\mathrm{s}^\mathrm{c}(i) \leq \overline{\nu}_\mathrm{s}^\mathrm{c}$ and $0 \leq \nu_\mathrm{n}^\mathrm{c}(i) \leq \overline{\nu}_\mathrm{n}^\mathrm{c}$, see Fig. 3. To reduce the number of variables in the MIP, a single slack variable $\nu_\mathrm{s}^\mathrm{c}(i) = a_\mathrm{sn}\, \nu_\mathrm{n}^\mathrm{c}(i)$ may be used, where $a_\mathrm{sn} > 0$ is a constant.

Remark 4: For each obstacle $j \in \mathbb{Z}_1^{n_\mathrm{obs}}$ in (12), we predict its position based on a constant velocity profile in curvilinear coordinates. Future work may include the use of a more advanced prediction model, e.g., a switching dynamical model [43] or a neural network classifier [44]. □

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

Remark 5: The predicted trajectory of the ego vehicle cannot "jump over" an obstacle driving in the same lane between two subsequent discretization points due to intersampling constraint violation if the safety distance in (12) is at least $t_{\text{safe}} > (T_\text{s}/2)$ in front and behind the obstacle, where T_s is the discretization time of the MIP-DM. A similar margin can be added also laterally to avoid "cutting corners" while noting that by enforcing constraints in a finer time grid and updating at a higher frequency, NMPC corrects constraint violations in the intersampling due to incorrect obstacle predictions.

1) Traffic Intersection Crossing Constraints: The obstacle avoidance constraints in (12) are also used to prevent the ego vehicle from crossing a traffic intersection, e.g., forcing the vehicle to stop during a particular time window. As in Fig. 3, the avoidance region is defined by the dimensions of the intersection, enlarged to account for the physical shape of the ego vehicle and with additional safety margins to account for modeling errors. If the intersection is controlled by traffic lights and the traffic light changes are known, e.g., using V2I communication [35], the intersection crossing constraints are time-varying within the prediction horizon. For example, if it is known that a traffic light will turn red, the intersection crossing constraints (12) force the ego vehicle to slow down and plan a stopping maneuver. The constraints are relaxed at future time steps within the prediction horizon when the traffic lights are predicted to become green. Alternatively, the intersection crossing constraints may be implemented based on map information and/or the perception system [31].

D. Zone-Dependent Traffic Rules

In real-world scenarios, certain traffic rules change in different road zones as follows:

1) speed limit, e.g., the vehicle enters a low-speed zone;
2) lane changes, e.g., lane changes become forbidden;
3) available lanes, e.g., a three-lane road transition to become two-lane road or the vehicle must merge in a new road.

We introduce binary variables $b_\text{z} = [b_\text{z}^1, \dots, b_\text{z}^{n_\text{z}}]$, where n_z denotes the number of position-dependent zones. Each zone is represented by a range $[\underline{p}_j, \overline{p}_j]$ for $j \in \mathbb{Z}_1^{n_\text{z}}$ in the longitudinal p_s-direction. We detect whether the vehicle is in zone j as

$$b_\text{z}^j(i) = 1 \quad \Longleftrightarrow \quad \underline{p}_j(i) \le p_\text{s}(i) \le \overline{p}_j(i). \tag{13}$$

Because the position-dependent zones are disjoint, the vehicle needs to be in exactly one zone, i.e., $\sum_{j=1}^{n_\text{z}} b_\text{z}^j = 1$.

The auxiliary binary variables b_z and constraints in (13) enable implementing the zone-dependent traffic rules. For example, changing speed limits can be enforced by

$$v_\text{s}(i) \le \sum_{j=1}^{n_\text{z}} b_\text{z}^j \, \overline{v}_\text{s}^j(i) \tag{14}$$

where the speed limit $\overline{v}_\text{s}^j(i)$ corresponds to zone $j \in \mathbb{Z}_1^{n_\text{z}}$. Similarly, the constraints on feasible lanes can be adjusted as

$$\sum_{j=1}^{n_\text{z}} b_\text{z}^j \, \underline{p}_\text{n}^{\text{ref},j}(i) \le p_\text{n}^\text{ref}(i) \le \sum_{j=1}^{n_\text{z}} b_\text{z}^j \, \overline{p}_\text{n}^{\text{ref},j}(i). \tag{15}$$

The constraints in (15) can be used to enforce a maximum position at which the ego vehicle needs to perform a lane change maneuver, e.g., as requested by a routing module in order to make a turn at the next traffic intersection. Fig. 4 shows the transition from a three-lane road segment into a two-lane road segment using (15).

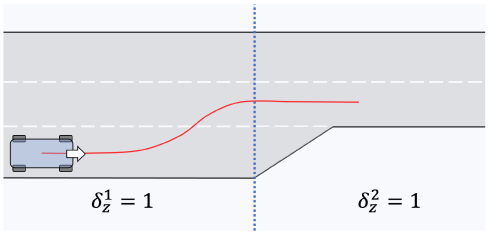


Fig. 4. Zone-dependent traffic rule: transition from a zone with three lanes ($b_\text{z}^1 = 1$) to a zone with two lanes ($b_\text{z}^2 = 1$), using MIP inequalities in (13) and (15).

E. Extended Dynamic System With Auxiliary Variables

The vehicle kinematics (2) and the auxiliary dynamics (9) result in the prediction model (1b)

$$\begin{bmatrix} p_\text{s}(i+1) \\ p_\text{n}(i+1) \\ v_\text{s}(i+1) \\ p_\text{n}^\text{ref}(i+1) \\ n_\text{LC}(i+1) \end{bmatrix} = \begin{bmatrix} 1 & 0 & T_\text{s} & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} p_\text{s}(i) \\ p_\text{n}(i) \\ v_\text{s}(i) \\ p_\text{n}^\text{ref}(i) \\ n_\text{LC}(i) \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & T_\text{s} & 0 & 0 & 0 \\ T_\text{s} & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} a_\text{s}(i) \\ v_\text{n}(i) \\ \Delta_\text{c}(i) \\ b_\text{c}^\text{u}(i) \\ b_\text{c}^\text{d}(i) \end{bmatrix} \tag{16}$$

subject to simple bounds on states for $i \in \mathbb{Z}_0^N$

$$-\frac{w_\text{l}}{2} \le p_\text{n}(i) \le \left(n_\text{l} - \frac{1}{2}\right) w_\text{l}, \quad \underline{v}_\text{s}(i) \le v_\text{s}(i) \le \overline{v}_\text{s}(i)$$
$$0 \le p_\text{n}^\text{ref}(i) \le (n_\text{l} - 1)\, w_\text{l}, \quad 0 \le n_\text{LC}(i) \le n_\text{LC}^\text{max} \tag{17}$$

and on control inputs for $i \in \mathbb{Z}_0^{N-1}$

$$\underline{a}_\text{s}(i) \le a_\text{s}(i) \le \overline{a}_\text{s}(i), \quad \underline{v}_\text{n}(i) \le v_\text{n}(i) \le \overline{v}_\text{n}(i). \tag{18}$$

F. Objective for Decision-Making and Motion Planning

The objective function (1a) of the proposed MIP-DM is $\sum_{i=0}^{N} \ell_i(x(i), u(i))$, where the stage cost is

$$\begin{aligned} \ell_i = {} & w_1 \, \| p_\text{s}(i) - \overline{p}_\text{s}^\text{ref}(i) \|_2^2 + w_2 \, \| p_\text{n}(i) - p_\text{n}^\text{ref}(i) \|_2^2 \\ & + w_3 \, a_\text{s}(i)^2 + w_4 \, v_\text{n}(i)^2 + w_5 \, b_\text{c}(i) \\ & + w_6 \, | p_\text{n}^\text{ref}(i) - \overline{p}_\text{n}^\text{ref}(i) | + w_7 \, v^\text{c}(i). \end{aligned} \tag{19}$$

$b_\text{c}(i) = b_\text{c}^\text{u}(i) + b_\text{c}^\text{d}(i)$, $v^\text{c}(i) = v_\text{s}^\text{c}(i) + v_\text{n}^\text{c}(i)$, and $w_j \ge 0$ for $j = 1, \dots, 7$ are weights. The first term in (19) is the longitudinal tracking error with respect to a reference trajectory $\overline{p}_\text{s}^\text{ref}(i)$, e.g., computed based on a desired reference velocity. The second term minimizes the lateral tracking error with respect to the current center lane. The third and fourth terms penalize the control actions and the longitudinal acceleration and lateral velocity, respectively. The fifth term penalizes lane change decisions, avoiding unnecessary lane changes, and rendering unlikely aborting a lane change, unless necessary for constraint satisfaction, i.e., to preserve safety.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

The sixth term in (19) minimizes a tracking error of the current lane with respect to a given preferred lane value $\overline{p}_{\mathrm{n}}^{\mathrm{ref}}(i)$, e.g., the right lane in right-hand traffic or the left lane when a vehicle desires to make a left turn at a next traffic intersection. To handle the absolute value in (19), we minimize an auxiliary control variable $\Delta p_{\mathrm{n}}^{\mathrm{ref}}$, satisfying

$$\Delta p_{\mathrm{n}}^{\mathrm{ref}} \geq p_{\mathrm{n}}^{\mathrm{ref}} - \overline{p}_{\mathrm{n}}^{\mathrm{ref}}, \quad \Delta p_{\mathrm{n}}^{\mathrm{ref}} \geq \overline{p}_{\mathrm{n}}^{\mathrm{ref}} - p_{\mathrm{n}}^{\mathrm{ref}} \tag{20}$$

such that $\Delta p_{\mathrm{n}}^{\mathrm{ref}} \geq |p_{\mathrm{n}}^{\mathrm{ref}} - \overline{p}_{\mathrm{n}}^{\mathrm{ref}}|$ holds. The squared terms in (19) may be replaced by absolute values, which results in a mixed-integer linear program (MILP) instead of an MIQP. However, a quadratic cost is often preferred for vehicle applications, see [29], [40], because it results in smoother control actions and trajectories than a linear cost, due to the different properties of the optimal points. The last term in (19) corresponds to a penalty on the slack variables for soft constraint violations. The weight $w_7 \gg 0$ is chosen large enough to ensure that a feasible solution with $v^{\mathrm{c}}(i) = 0$ is found if and when it exists.

Remark 6: The weights $w_j \geq 0$ in (19) may be adapted to the traffic conditions, e.g., adjusting the balance between the time to execute a lane change and comfort. An in-depth discussion on this is outside the scope of this article.

The complete MIOCP of the proposed MIP-DM is

$$\begin{aligned}
&\min_{X,\,U} \quad \sum_{i=0}^{N} \ell_i(x(i), u(i)) \text{ in (19)} \\
&\text{s.t.} \quad x(0) = \hat{x}_t \\
&\qquad \text{Extended state dynamics in (16)} \\
&\qquad \text{Simple bound constraints in (17) and (18)} \\
&\qquad \text{Lateral velocity constraint in (6)} \\
&\qquad \text{Lane change constraints in (7) and (8)} \\
&\qquad \text{Time delay constraints in (10) and (11)} \\
&\qquad \text{Obstacle avoidance constraints: Section III-C} \\
&\qquad \text{Zone-dependent traffic rules: Section III-D.}
\end{aligned} \tag{21}$$

The state vector is $x = [p_{\mathrm{s}}, p_{\mathrm{n}}, v_{\mathrm{s}}, p_{\mathrm{n}}^{\mathrm{ref}}, n_{\mathrm{LC}}, t_{\mathrm{c}}]$, and the control and auxiliary input vector is $u = [a_{\mathrm{s}}, v_{\mathrm{n}}, \tilde{t}_{\mathrm{c}}, \Delta_{\mathrm{c}}, b_{\mathrm{c}}, b_{\mathrm{o}}, b_{\mathrm{z}}]$. The binary optimization variables include the lane change variables $b_{\mathrm{c}} = [b_{\mathrm{c}}^{\mathrm{u}}, b_{\mathrm{c}}^{\mathrm{d}}]$, the obstacle avoidance variables $b_{\mathrm{o}} = [b_{\mathrm{o}}^{1}, \ldots, b_{\mathrm{o}}^{n_{\mathrm{obs}}}]$, and the traffic zone variables $b_{\mathrm{z}} = [b_{\mathrm{z}}^{1}, \ldots, b_{\mathrm{z}}^{n_z}]$, while the remaining variables are continuous.

Remark 7: With an upper bound on the number of obstacles taken into account in a single problem, the MIOCP dimensions are fixed. This allows for static memory allocation, as required for implementing the MIP-DM in embedded platforms suitable for automotive applications, as discussed later. □

IV. Embedded MIQP Solver for Mixed-Integer MPC

The MIOCP (21) is converted into the MIQP

$$\min_{z} \quad \frac{1}{2} z^\top H z + h^\top z \tag{22a}$$

$$\text{s.t.} \quad G z \leq g, \quad F z = f \tag{22b}$$

$$z_j \in \mathbb{Z}, \quad j \in \mathcal{I} \tag{22c}$$

where z includes all optimization variables, and the index set $\mathcal{I}$ denotes the integer variables. Next, we summarize the main ingredients of the `BB-ASIPM` solver [13] that uses a B&B method with reliability branching and warm starting [16], block-sparse presolve techniques [13], and early termination and infeasibility detection [17] within a fast convex QP solver [18].

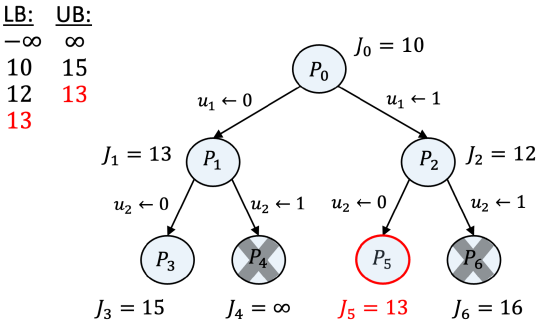


Fig. 5. B&B method as a binary search tree. A selected node can be either *branched*, resulting in two partitions for each binary variable $u_j \in \{0, 1\}$, or *pruned*, based on feasibility or the current upper bound.

Remark 8: The `BB-ASIPM` solver is not tailored to the MIP-DM problem (21), but rather designed to solve any MIOCP as in (1). However, the problem formulation is an important factor affecting the computational burden of any MIQP solver, see [10].

A. B&B Method and Search Heuristics

The B&B algorithm sequentially creates partitions of the MIQP, as shown in Fig. 5. For each partition, a local lower bound on the optimal objective value is obtained by solving a convex relaxation of the MIQP subproblem. If the relaxation yields an integer-feasible solution, the B&B updates the global upper bound for the MIQP solution, which is used to *prune* tree partitions. The B&B method terminates when the difference between the upper and lower bound is below a user-defined threshold. A key decision is how to create partitions, i.e., which node to choose and which discrete variable to select for branching. `BB-ASIPM` uses *reliability branching*, which combines strong branching and pseudo-costs [45].

B. Block-Sparse Exact Presolve Reduction Techniques

We refer to the parametric MIQP from (22) as $\mathcal{P}(\theta)$, in which the parameter vector θ includes the state estimate $\hat{x}_t$, and we denote the discrete variables in (22c) by $d \in \mathbb{Z}^{N_d}$. We use the compact notation $\mathcal{P}(\theta, d_{\mathcal{R}} = \hat{d})$ to denote the MIQP after fixing $d_j = \hat{d}_j$, $j \in \mathcal{R}$ where $\mathcal{R}$ is an index set.

Definition 2 (Presolve Step): Given problem $\mathcal{P}(\theta)$ and a set of integer values $\{\hat{d}_j\}_{j\in\mathcal{R}}$ for the index set $\mathcal{R} \subseteq \{1, \ldots, N_d\}$, the presolve step computes

$$\{\texttt{infeasible}, \hat{d}^+, \mathcal{R}^+\} \leftarrow \text{Presolve}(\mathcal{P}(\theta), \hat{d}, \mathcal{R}) \tag{23}$$

resulting in updated integer values $\{\hat{d}_j^+\}_{j\in\mathcal{R}^+}$ for the index set $\mathcal{R}^+ \subseteq \{1, \ldots, N_d\}$ such that the following conditions hold.

1) The new index set includes the original set, $\mathcal{R} \subseteq \mathcal{R}^+$.
2) $\mathcal{P}(\theta, d_{\mathcal{R}^+} = \hat{d}^+)$ is infeasible/unbounded only if $\mathcal{P}(\theta, d_{\mathcal{R}} = \hat{d})$ is infeasible/unbounded.
3) Any feasible/optimal solution of $\mathcal{P}(\theta, d_{\mathcal{R}^+} = \hat{d}^+)$ maps to a feasible/optimal solution of $\mathcal{P}(\theta, d_{\mathcal{R}} = \hat{d})$, with identical objective value. □

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

A presolve routine applied to a root node in B&B corresponds to Definition 2 with $\mathcal{R} = \emptyset$. In general, presolve cannot prune all of the binary or integer decision variables, but often, it leads to a reduced problem that is significantly faster to solve.

We use the tailored block-sparse presolve procedure [13, Sec. 4] that abides by the rules in Definition 2 and includes the following.

1) *Domain propagation* to strengthen bounds based on constraints of the MIQP, which may lead to fixing multiple integer variables. A tailored implementation for MIOCPs of the form in (1) based on an iterative forward-backward propagation is described in [13, Algorithm 2].
2) *Redundant constraints* are detected and removed based on updated bound values, which may also benefit *dual fixing* of multiple variables, see [13, Algorithm 4].
3) *Coefficient strengthening* to tighten the feasible space of the convex QP relaxation without removing any integer-feasible solution of the MIQP. A block-sparse implementation is described in [13, Algorithm 5].
4) *Variable probing* to obtain tightened bound values for multiple optimization variables by temporarily fixing a binary variable to 0 and 1, see [13, Algorithm 6].

The presolve procedure in [13] terminates if the problem is detected to be infeasible or if insufficient progress is made from one iteration to the next, but an upper limit on the number of presolve iterations or a timeout is typically included for efficiency.

C. Block-Sparse QP Solver for Convex Relaxations

A primal–dual interior point method (IPM) uses a Newton-type algorithm to solve a sequence of relaxed Karush–Kuhn–Tucker (KKT) conditions for the convex QP. We use the active-set based inexact Newton implementation of `ASIPM` [18], which exploits the block-sparse structure in the linear system with improved numerical conditioning, reduced matrix factorization updates, warm starting, early termination, and infeasibility detection [17]. If the convex QP relaxation is infeasible and has optimum exceeding the current global upper bound, then the node and the corresponding subtree can be pruned from the B&B tree. A considerable computational effort can be avoided if the above scenarios are detected early, i.e., before reaching the solution of the convex QPs. In [17], we describe an early termination method based on a tailored dual feasibility projection strategy applicable to `BB-ASIPM` to handle both cases and to reduce the computational effort of the B&B method without affecting the quality of the optimal solution.

D. Embedded Software Implementation of Hybrid MPC

In hybrid MPC, warm starting can be used to reduce the computational effort in the B&B method from one time step to the next as discussed in [46] and [47]. `BB-ASIPM` uses *tree propagation* [13], [16] to efficiently reuse the branching decisions and pseudo-costs from the previous MIQP solution. The `BB-ASIPM` solver is implemented in self-contained C code, which allows for real-time implementations on embedded microprocessors as shown next. An upper bound can be imposed on the number of B&B iterations to ensure a maximum computation time below a threshold. If an integer-feasible solution is found, the B&B method automatically provides a bound on the suboptimality. As in any real-world safety-critical application where a complex numerical algorithm is deployed, it is advisable to include simple backup strategies for when the algorithm is unable to converge to a feasible solution within the available time. Some examples are using a previously computed motion plan or a default maneuver, such as staying in the lane and following the vehicle ahead, or slowing down/braking to a stop. However, in all the simulations in Section VI and experiments in Section VII, `BB-ASIPM` never failed to compute a feasible solution within the maximum number of B&B iterations, and in most cases, the globally optimal solution was found.

V. Integration of MIP-DM and NMPC Reference Tracking Controller

The NMPC reference tracking controller executes the motion plan of the MIP-DM, see Fig. 1(b). Based on the vehicle model in (2), the MIP-DM reference trajectory in curvilinear coordinates $[p_s(i), p_n(i), v_s(i)]^\top$, $i \in \mathbb{Z}_0^N$, is transformed to a Cartesian coordinate frame (p_X, p_Y) as in Fig. 2. With the approximation of the heading angle $\psi(i) \approx \arctan(p_Y(i+1) - p_Y(i)/p_X(i+1) - p_X(i))$, we obtain the reference trajectory $[p_X(i), p_Y(i), \psi(i), v(i)]^\top$ for $i \in \mathbb{Z}_0^N$. As in [33] and [48], we use a 3rd-order polynomial approximation, resulting in $\boldsymbol{y}^{\text{ref}}(\tau) = [p_X^{\text{ref}}(\tau), p_Y^{\text{ref}}(\tau), \psi^{\text{ref}}(\tau), v^{\text{ref}}(\tau)]^\top$ for $0 \le \tau \le T^{\text{mpc}}$, where T^{mpc} is the NMPC horizon length.

For the NMPC prediction model, we use the nonlinear kinematic model (5) with additional actuation dynamics [33], resulting in the continuous-time dynamics

$$\dot{p}_X = v\cos(\psi+\beta), \quad \dot{p}_Y = v\sin(\psi+\beta) \tag{24a}$$

$$\dot{\psi} = v\,\frac{\cos(\beta)}{L}\tan(\delta_f), \quad \dot{\delta}_f = \frac{1}{t_d}(\delta - \delta_f) \tag{24b}$$

$$\dot{v} = u_1, \quad \dot{\delta} = u_2 \tag{24c}$$

where p_X and p_Y denote the 2-D position in Cartesian coordinates; ψ is the heading angle; $\dot{\psi}$ the heading rate; v is the longitudinal velocity; δ and δ_f are the commanded and actual front wheel steering angle, respectively; and L and β are defined in (5). First-order front steering dynamics are included in (24b) for the steering actuation response, and inputs u_1 and u_2 are the acceleration and steering rate command, respectively.

At each control time step t, the NMPC solves

$$\min_{X,\,U} \quad \frac{1}{2}\sum_{i=0}^{N^{\text{mpc}}} \|\boldsymbol{y}(k) - \boldsymbol{y}^{\text{ref}}(t_k)\|_Q^2 + \|u(k)\|_R^2 + \qquad r_v\, v(k) \tag{25a}$$

$$\text{s.t.} \quad x(0) = \hat{x}_t \tag{25b}$$

$$x(k+1) = f_k(x(k), u(k)) \quad \forall k \in \mathbb{Z}_0^{N^{\text{mpc}}-1} \tag{25c}$$

$$\underline{c}_k \le c_k(x(k), u(k)) \le \overline{c}_k \quad \forall k \in \mathbb{Z}_0^{N^{\text{mpc}}} \tag{25d}$$

where the N^{mpc} control intervals are defined by an equidistant grid of time points $t_k = k(T^{\text{mpc}}/N^{\text{mpc}})$ for $k \in \mathbb{Z}_0^{N^{\text{mpc}}}$ over the NMPC horizon, $\hat{x}_t$ is the current state estimate at time t, and the constraints in (25c) are a discretization of the continuous-time dynamics in (24), e.g., using a 4th-order

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

Runge–Kutta method. The NMPC includes the squared output, $\boldsymbol{y}$, tracking error with respect the reference, $\boldsymbol{y}^{\text{ref}}$, and command effort, and an L_1 penalty on the slack variable $\nu(k) \geq 0$, where the weight $r_\nu \gg 0$ is sufficiently large to ensure that $\nu(k) = 0$ when a feasible solution exists [49].

Constraint (25d) includes hard bounds on the control inputs and soft constraints for limiting the distance to the reference trajectory, the velocity, and the steering angle

$$5 - \bar{e}_Y \leq e_Y + \nu, \quad -\bar{\delta}_f \leq \delta_f + \nu, \quad -\bar{v} \leq v + \nu \tag{26a}$$

$$e_Y \leq \bar{e}_Y + \nu, \quad \delta_f \leq \bar{\delta}_f + \nu, \quad v \leq \bar{v} + \nu \tag{26b}$$

$$-\bar{\dot{\delta}} \leq \dot{\delta} \leq \bar{\dot{\delta}}, \quad -\bar{\dot{v}} \leq \dot{v} \leq \bar{\dot{v}} \tag{26c}$$

where $e_Y(k) = \cos(\psi^{\text{ref}}(t_k))(p_Y(k) - p_Y^{\text{ref}}(t_k)) - \sin(\psi^{\text{ref}}(t_k))(p_X(k) - p_X^{\text{ref}}(t_k))$ is the distance to the reference trajectory. Instead of the rectangular collision region in curvilinear coordinates as in the MIP-DM (see Fig. 3), in NMPC, obstacle avoidance is enforced by ellipsoidal constraints in Cartesian coordinates

$$1 \leq \left(\frac{\tilde{p}_{x,j}(k)}{a_{x,j}}\right)^2 + \left(\frac{\tilde{p}_{y,j}(k)}{a_{y,j}}\right)^2 \tag{27}$$

where

$$\begin{bmatrix} \tilde{p}_{x,j} \\ \tilde{p}_{y,j} \end{bmatrix} = R(o_{\psi,j})^\top \begin{bmatrix} p_X - o_{X,j} \\ p_Y - o_{Y,j} \end{bmatrix}$$

is the rotated distance, $(o_{X,j}, o_{Y,j}, o_{\psi,j})$ is the obstacle's pose, and $(a_{x,j}, a_{y,j})$ are the lengths of the principal semi-axes of the ellipsoid that ensure a safety margin around each obstacle. Our efficient implementation of NMPC solves the nonlinear program in (25) using the RTI algorithm in [36], which performs a single SQP iteration at each control time step based on a shifted solution guess from the previous time step. An extensive discussion on the NMPC optimization algorithm and its implementation and computation times in small-scale vehicle experiments can be found in [48] and hence is not repeated here.

Remark 9: MIP-DM may be slow in reacting to sensing or prediction errors of other vehicles, due to its sampling period typically $T_s^{\text{mip}} \approx 0.4$–$2$ s. Therefore, obstacle avoidance is executed by both MIP-DM and by NMPC, which has a higher control rate with sampling period $T_s^{\text{mpc}} \approx 0.02$–$0.1$ s and hence ensures constraint enforcement on a finer time grid and a more rapid reaction to sensing or prediction errors of the other vehicles.

VI. Numerical Simulation Results

We present numerical simulation results for the MIP-DM described in Section III, in a variety of traffic scenarios. We also compare the `BB-ASIPM` solver in Section IV against state-of-the-art software tools, and we demonstrate its real-time feasibility on dSPACE rapid prototyping units.

A. Problem Formulation and Simulation Test Scenarios

In this section, we perform simulations of MIP-DM in MATLAB using the vehicle model in (2) to show the variety of traffic scenarios that can be handled using the MIQP in Section III. We use the simple model (2) to assess the behavior and the stand-alone computational load of MIP-DM. In these simulations, the obstacles do not react to the ego vehicle's behavior, i.e., they do not favor or actively harm the ego vehicle, but rather only pursue their own objectives while satisfying the traffic rules. While, in the experiments, there is some limited reaction of the traffic to the ego vehicle, e.g., the following and queuing behavior and the intersection rules, the evaluation and adaptation of the method to traffic participants that continuously react to the ego motion, i.e., interactively planning with traffic, is left for future studies. Also, in the experiments reported later, we validate robustness to model approximations, disturbances, sensing, and prediction errors.

TABLE I

Problem Dimensions and Parameters in the MIQP Formulation of Section III for Each of the Test Scenarios in Fig. 6. The Number of Binary Variables per Each Step in the MIOCP Prediction Horizon Is $n_b = 2 + 3\,n_{\text{obs}} + n_z$

	N	n_x	n_u	n_b	n_c	n_{obs}	n_z
① see Fig. 6a	15	6	20	14	60	3	3
② see Fig. 6b	15	6	18	12	56	3	1
③ see Fig. 6c	15	6	23	17	71	4	3
④ see Fig. 6d	15	6	24	18	73	4	4
⑤ see Fig. 6e	15	6	16	10	47	2	2
⑥ see Fig. 6f	15	6	17	11	49	2	3
⑦ see Fig. 6g	15	6	20	14	60	3	3
⑧ see Fig. 6h	15	6	20	14	60	3	3

Fig. 6 shows a snapshot of MATLAB simulations for eight test scenarios. Table I shows the problem dimensions and parameter values in the MIQP formulation of Section III for the scenarios in Fig. 6, where $N = 15$ is the horizon length; n_x is the number of state variables; n_u is the number of control variables; n_c is the number of inequality constraints, each per time step; and $n_b = 2 + 3\,n_{\text{obs}} + n_z$ is the number of binary variables per time step, with n_{obs} the maximum number of obstacles (see Section III-C) and n_z the number of zones (see Section III-D). Using a sampling time of $T_s = 1$ s, the MIP-DM time horizon is $T = N\,T_s = 15$ s.

Scenario 1 in Fig. 6(a) shows the ego vehicle overtaking three obstacles, where two obstacles are on lane 1 and a third obstacle is on lane 2, on a road segment with one-way traffic. Lane 1 refers to the rightmost lane with respect to the ego vehicle's direction of motion. Scenario 2 in Fig. 6(b) shows the ego vehicle swaying around two parked vehicles (stopped) on lane 1 while avoiding a third vehicle on lane 2. Scenario 3 in Fig. 6(c) shows the ego vehicle overtaking one vehicle on lane 1 before stopping at an intersection and then crossing after two other vehicles. Scenario 4 in Fig. 6(d) shows the ego vehicle overtaking three obstacles (two vehicles on lane 1 and one vehicle on lane 2) on a curved road segment with one-way traffic, followed by stopping and crossing an intersection. In test scenarios 1–4, lane 1 is the preferred lane $\overline{p}_n^{\text{ref}}$ in (19) so that the ego vehicle always returns to lane 1 after each overtaking or sway maneuver.

Scenario 5 in Fig. 6(e) shows the ego vehicle merging from lane 1 to lane 2 between three vehicles on lane 2, i.e., the preferred lane $\overline{p}_n^{\text{ref}}$ in (19) is lane 2. Scenario 6 in Fig. 6(f) shows the ego vehicle merging at the end of its lane onto a new lane while avoiding and/or overtaking three vehicles on the same lane. Scenario 7 in Fig. 6(g) shows the ego vehicle

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

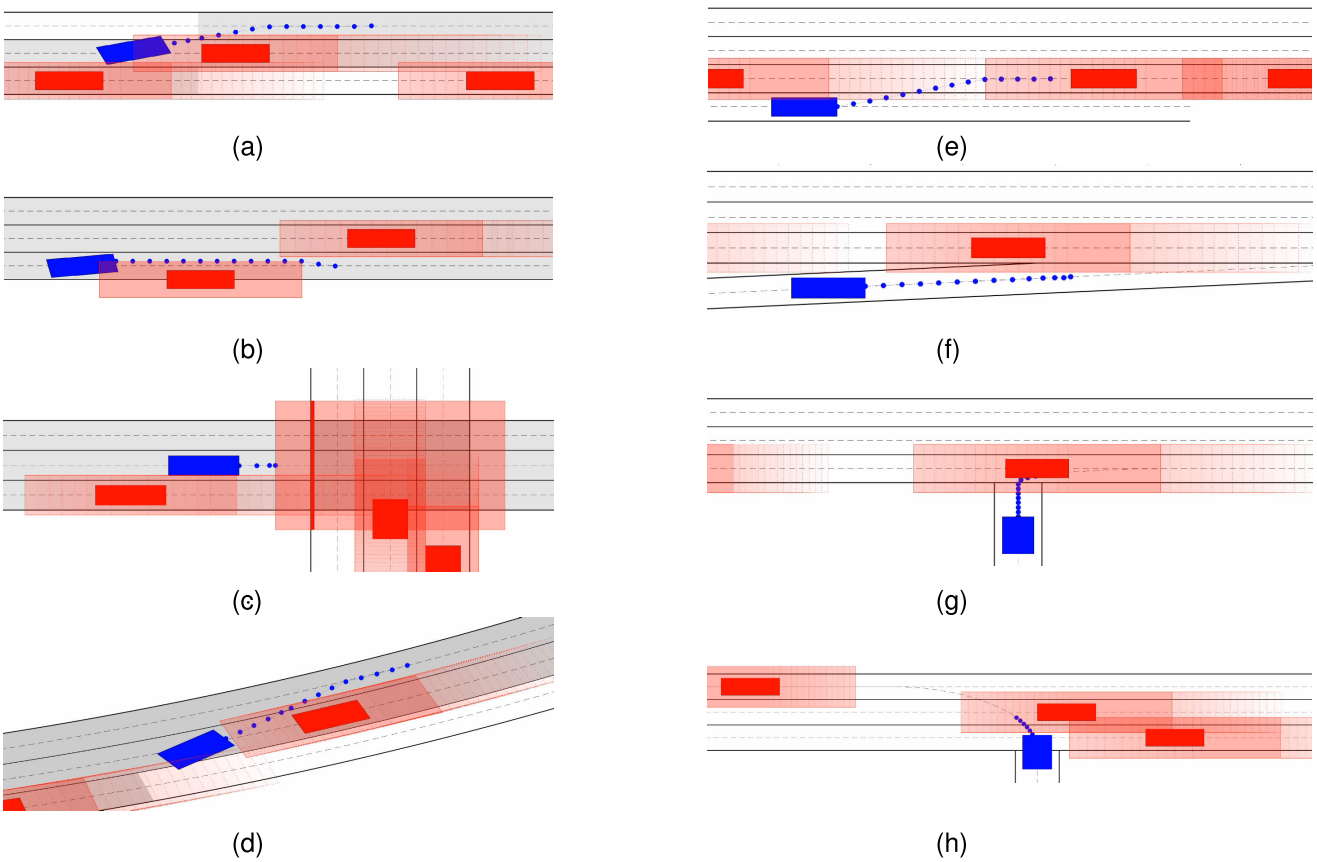


Fig. 6. Snapshot of the closed-loop MATLAB simulations using the MIP-DM in eight test scenarios. The ego vehicle is shown in blue, and other vehicles are shown in red. A video recording of the simulations is available at https://youtu.be/fe9lUOQUPoU. (a) Scenario 1: ego vehicle overtakes three obstacles on a one-way traffic road. (b) Scenario 2: ego vehicle sways for two parked vehicles (only one visible), avoiding a third vehicle on another one-way traffic lane. (c) Scenario 3: ego vehicle overtakes before stopping at intersection and then continues after two vehicles finish crossing the intersection. (d) Scenario 4: ego vehicle overtakes obstacles on a curved road with one-way traffic, followed by stopping and crossing an intersection. (e) Scenario 5: ego vehicle merges to lane 2 between three vehicles in one-way traffic. (f) Scenario 6: ego vehicle merges at the end of lane onto a new lane while avoiding/overtaking three vehicles (only one visible). (g) Scenario 7: ego vehicle performs right turn at a T-intersection, merging between two vehicles (only one visible) on the same lane. (h) Scenario 8: ego vehicle turns left at T-intersection, following one vehicle while avoiding two other vehicles driving in the opposite direction.

performing a right turn at a T-intersection, merging between two vehicles on the same lane of the new road segment. Scenario 8 in Fig. 6(h) shows the ego vehicle performing a left turn at a T-intersection, following one vehicle on the same lane while avoiding two other vehicles driving in the opposite direction. In test scenarios 5–8, after a merging or turning maneuver, the ego vehicle overtakes any other vehicle that is driving below the speed limit.

B. Computational Performance and Solver Comparisons

Table II shows the average and worst case computation times of MIP-DM for each of the eight simulation scenarios that are illustrated in Fig. 6, using the MIQP formulation as described in Section III and where the MIQPs at each control time step are solved using either `GUROBI`, `MOSEK`, or `BB-ASIPM`. It can be observed that the average and worst case computation times of `BB-ASIPM` are approximately five and four times faster than `MOSEK`, respectively. On the other hand, the average and worst case computation times of `GUROBI` are approximately 1.5 and 2.5 times faster than `BB-ASIPM`, respectively. All the default presolve options are enabled in the `GUROBI` solver, and the maximum number of B&B iterations in the `BB-ASIPM` solver was never reached in the scenarios in Table II. Computation times in Table II and Fig. 7 are obtained by executing the MIP-DM simulations in a single core of a 2.4-GHz Intel Core i9 processor.

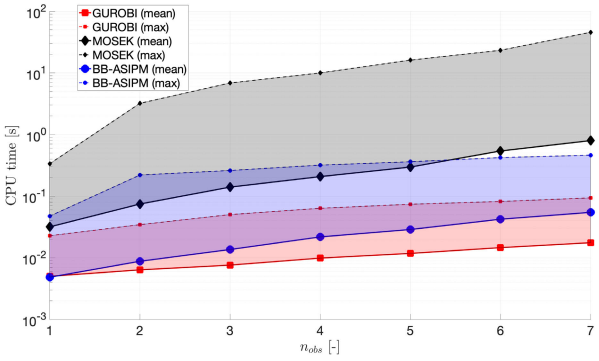


Fig. 7. Average and worst case computation times of MIP-DM for an overtaking scenario similar to Fig. 6(a). Results for 200 randomized simulations for each number of obstacles $n_{\text{obs}} \in \mathbb{Z}_1^7$, using `GUROBI`, `MOSEK`, and `BB-ASIPM`.

Fig. 7 shows the average and worst case computation times of MIP-DM for an overtaking scenario similar to Fig. 6(a), showing results for 200 randomized simulations for each number of obstacles $n_{\text{obs}} \in \mathbb{Z}_1^7$, using `GUROBI`, `MOSEK`, and `BB-ASIPM`. For each simulation, the initial position, the

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

TABLE II

AVERAGE AND WORST CASE COMPUTATION TIMES FOR EACH OF THE EIGHT SCENARIOS IN FIG. 6 FOR MIP-DM WITH THE MIQP FORMULATION IN SECTION III, FOR GUROBI, MOSEK, AND BB-ASIPM

	GUROBI		MOSEK		BB-ASIPM	
	Mean time	Max time	Mean time	Max time	Mean time	Max time
① see Fig. 6a	12.6 ms	31.5 ms	142.5 ms	558.6 ms	21.8 ms	85.1 ms
② see Fig. 6b	6.3 ms	18.9 ms	32.4 ms	191.2 ms	8.1 ms	49.7 ms
③ see Fig. 6c	7.8 ms	25.6 ms	60.9 ms	225.0 ms	10.6 ms	54.3 ms
④ see Fig. 6d	7.7 ms	25.5 ms	57.9 ms	200.6 ms	12.3 ms	65.9 ms
⑤ see Fig. 6e	5.8 ms	23.3 ms	47.5 ms	254.9 ms	8.6 ms	60.8 ms
⑥ see Fig. 6f	6.4 ms	23.7 ms	58.8 ms	262.7 ms	9.3 ms	57.5 ms
⑦ see Fig. 6g	4.9 ms	23.0 ms	31.9 ms	348.8 ms	7.0 ms	58.3 ms
⑧ see Fig. 6h	5.0 ms	23.0 ms	33.6 ms	153.6 ms	6.0 ms	45.7 ms

TABLE III

AVERAGE AND WORST CASE COMPUTATION TIMES, NUMBER OF B&B ITERATIONS, TOTAL NUMBER OF ASIPM ITERATIONS, AND MEMORY FOOTPRINT OF EMBEDDED BB-ASIPM ON DSPACE SCALEXIO AND DSPACE MABX-III, FOR HIL SIMULATIONS OF THE MIP-DM METHOD FOR THE EIGHT SCENARIOS IN FIG. 6

	BB-ASIPM solver				BB-ASIPM on **dSPACE Scalexio**				BB-ASIPM on **dSPACE MABX-III**			
	B&B iters		ASIPM iters		CPU time [ms]		Memory [KB]		CPU time [ms]		Memory [KB]	
	mean	max	mean	max	mean	max	text	data	mean	max	text	data
① see Fig. 6a	6.3	65	56.1	495	22.7	185.2	170	13633	99.9	774.0	132	12217
② see Fig. 6b	4.7	35	41.2	199	14.0	56.0	167	12639	63.4	240.1	132	11407
③ see Fig. 6c	4.3	25	42.2	153	16.9	60.9	170	18015	74.6	252.8	133	16209
④ see Fig. 6d	7.5	39	66.4	353	27.5	131.7	177	18258	119.6	542.8	134	16343
⑤ see Fig. 6e	4.1	27	37.6	241	13.2	71.8	170	14391	59.2	300.2	132	12975
⑥ see Fig. 6f	7.9	45	73.4	351	18.8	102.2	170	11896	83.9	425.6	133	10897
⑦ see Fig. 6g	3.8	37	41.1	204	12.5	61.4	170	12364	54.6	251.5	133	11283
⑧ see Fig. 6h	4.4	37	42.3	261	12.8	74.1	170	14391	55.0	308.1	132	12975

lane, and the velocity for each of the obstacle vehicles are selected randomly from a uniform distribution. Based on the results in Fig. 7, the MIP-DM using the BB-ASIPM solver is able to remain real-time feasible for $n_{\text{obs}} \in \mathbb{Z}_1^7$ on a standard computer. As expected, the gap between the average computation time of BB-ASIPM versus GUROBI grows for an increasing number of obstacles n_{obs}, but the worst case computation times remain below the sampling time of $T_s = 1$ s, which is not the case for MOSEK. Based on prior works in [21], [27], and [28], it is often sufficient to consider up to $n_{\text{obs}} = 5$ surrounding vehicles, for example, including the leading vehicle ahead in the current lane of the ego vehicle and the leading and following vehicles in the left and right adjacent lanes.

Given the relatively simple and compact algorithmic implementation in BB-ASIPM when compared to the extensive collection of advanced heuristics, presolve and cutting plane techniques in the commercial GUROBI [14] solver, it is reassuring to see that BB-ASIPM remains competitive with state-of-the-art software tools in Table II. On the other hand, the software implementation of BB-ASIPM [13] is relatively compact, on the order of 200 kB, and self-contained such that it can execute on an embedded microprocessor for real-time vehicle decision-making and motion planning. Instead, state-of-the-art optimization tools, such as GUROBI and MOSEK, typically are not designed for embedded control hardware with limited computational resources and available memory [1] since they are closed-source, have large program code (several MB), and extensively use features that are not compatible with hard real-time requirements, such as dynamic memory allocation and caching.

C. HIL Simulation Results on dSPACE Scalexio and MABX-III Rapid Prototyping Units

Next, we present the detailed results of running HIL simulations for each of the eight test scenarios shown in Fig. 6 on both the dSPACE Scalexio[1] and the dSPACE MABX-III[2] rapid prototyping units. Table III shows the average and worst case computation times, the number of B&B iterations, total number of ASIPM iterations, and the memory usage of the BB-ASIPM solver on Scalexio and MABX-III. The memory usage is categorized into *text* that contains code and constant data, which is typically stored in ROM, and *data* that are stored in RAM.

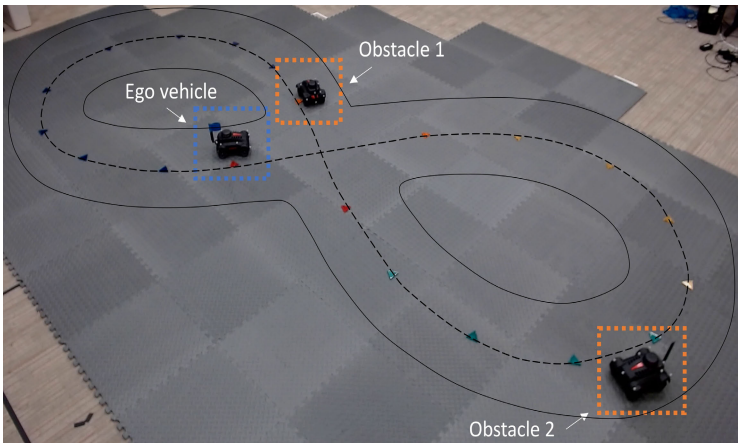
[1]dSPACE Scalexio DS6001 unit, with an Intel i7-6820EQ quad-core 2.8-GHz processor with 64-kB L1 cache per core, 256-kB L2 cache per core, 8-MB shared L3 cache, 4-GB DDR4 RAM, and 8-GB flash memory. In the presented results, MIP-DM executes in a single core.

[2]dSPACE MABX-III DS1403 unit, with four ARM Cortex-A15 processor cores with 32-kB L1 cache per core, 4-MB shared L2 cache, 2-GB DDR3L RAM, and 64-MB flash memory. In the presented results, MIP-DM executes in a single core.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

(a) Small-scale autonomous vehicle.

(b) OptiTrack motion capture camera.



(c) Experiments using three small-scale vehicles: the ego vehicle (blue flag) executing the MIP-DM and NMPC, and two obstacles (no flag).

Fig. 8. Experimental testbench that consists of (a) small-scale automated vehicles with onboard sensors and (b) OptiTrack motion-capture system. (c) Track and snapshot of the positions of the ego vehicle and the two obstacle vehicles.

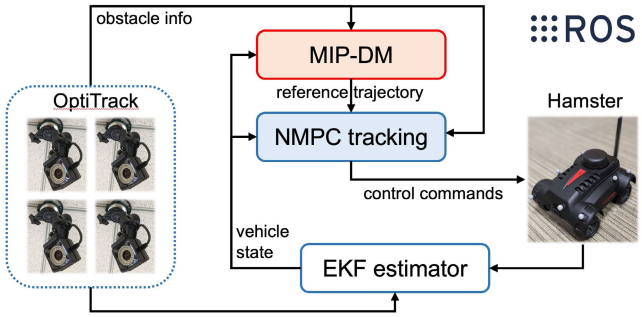


Fig. 9. Multilayer control architecture with MIP-DM, NMPC controller, and an extended Kalman filter (EKF) for state estimation based on measurements from the Optitrack system and onboard sensors of the Hamster.

From Table III, MIP-DM is real-time feasible using the proposed `BB-ASIPM` solver for each of the eight simulation scenarios on both the dSPACE Scalexio and MABX-III units, as the worst case computation time is below the sampling period $T_s = 1$ s at each time step. Considering all test scenarios, the computation times on the dSPACE Scalexio are always below 200 ms, below 100 ms 99% of the times, and in average 17.3 ms. On MABX-III, the computation times are always below 800 ms, below 400 ms 99% of the times, and in average 76.3 ms. The total memory usage for MIP-DM, solver, and fully pre-allocated memory is approximately 18 MB on Scalexio and 16.1 MB on MABX-III, due to the different compilers. As expected, Table III shows that for each test scenario, the number of iterations on Scalexio and MABX-III is identical.

VII. Experimental Results of MIP-DM and NMPC on Small-Scale Automated Vehicles

Next, we validate the performance of MIP-DM in combination with NMPC for reference tracking on experiments with small-scale vehicles, using ROS and an Optitrack motion-capture system [6]. First, we present the hardware and software setup, and then, we show the results from the experiments.

A. Hardware Setup and Software Implementation

The hardware setup, shown Fig. 8, is based on the *Hamster* [50] vehicle, see Fig. 8(a), a 25 × 20 cm mobile robot with electric steering and electric motor speed control. The robot is equipped with sensors such as a rotating 360° LiDAR, an inertial measurement unit, a GPS receiver, an HD camera, and motor encoders. It has Ackermann steering and its kinematic behavior emulates that of a regular vehicle. We use an Optitrack motion-capture system [51], see Fig. 8(b), to obtain position and orientation measurements for each of the Hamster vehicles, to focus our validation on the control performance.

Our test setup consists of three vehicles driving on a two-lane track shaped as a figure 8, causing a traffic intersection, see Fig. 8(c). Two Hamsters are designated as *obstacles*, executing a PID controller that tracks the center line of the current lane, and an adaptive cruise controller to ensure a safe following distance from the vehicle in front. A traffic intersection coordinator forces each of the obstacles to stop in front of the intersection for at least three seconds before continuing the execution of the lane keeping controller when the intersection is free. We adopt a first-come first-served rule in case multiple Hamsters arrive at the intersection around the same time, which is consistent with the driving rules in USA for nonsignalized four-way intersections. The third Hamster is the ego vehicle that is controlled by the multilayer control architecture shown in Fig. 9, i.e., the proposed MIP-DM method in combination with the NMPC for reference tracking as described in Section V. In the MIP-DM optimization problem, the obstacle vehicles are predicted to maintain a constant velocity, which contrasts with their actual behavior in the intersection and in following/queuing behind a leading vehicle. This allows us to evaluate the MIP-DM robustness to incorrect predictions of the other vehicles, which is achieved by feedback.

The sampling period of MIP-DM is reduced with respect to that of Section VI due to the scaling of the vehicles. MIP-DM executes with a sampling period of $T_s^{\text{mip}} = 0.3$ s and horizon length $N^{\text{mip}} = 15$. The NMPC problem in (25) includes $n_x = 6$ states, $n_u = 3$ control inputs, and

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

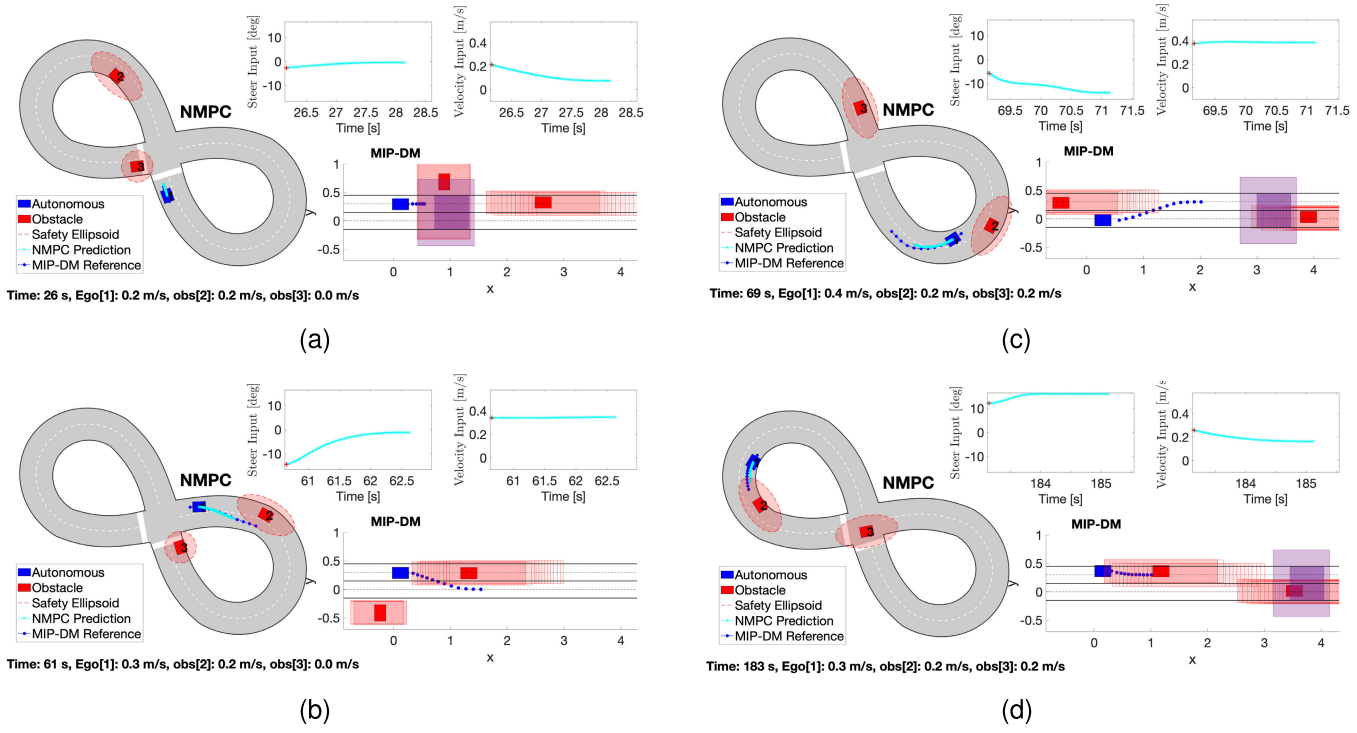


Fig. 10. Predicted trajectories of MIP-DM ($T_s^{\text{mip}} = 0.3$ s) and NMPC ($T_s^{\text{mpc}} = 0.025$ s) tracking the MIP-DM reference, at some instants of the experiments. The left side of each subfigure shows the 8-shaped track, the ego (blue) and two obstacles (red), safety ellipsoid around each obstacle (red dashed line), NMPC predicted trajectory (cyan cross markers), and MIP-DM reference (blue solid circles). The bottom-right side of each subfigure shows the ego (blue), two obstacles (red), traffic intersection (purple), and MIP-DM solution (blue solid circles) in curvilinear coordinates, and the top-right side shows the NMPC command trajectory (cyan cross markers). A video is available at https://youtu.be/fe9lUOQUPoU. (a) Trajectories for MIP-DM and NMPC at 26 s of experiment: ego vehicle stopping at traffic intersection. (b) Trajectories for MIP-DM and NMPC at 61 s of experiment: ego vehicle overtaking slower obstacle to achieve the desired velocity. (c) Trajectories for MIP-DM and NMPC at 69 s of experiment: ego vehicle returning to preferred lane after overtaking slower obstacle. (d) Trajectories for MIP-DM and NMPC at 183 s of experiment: ego vehicle slowing down behind slower obstacle because overtaking is not allowed.

$N^{\text{mpc}} = 80$ control intervals with a sampling period of $T_s^{\text{mpc}} = 25$ ms over a $T^{\text{mpc}} = 2$ s horizon length. The NMPC controller is implemented with a sampling frequency of 40 Hz, using the RTI algorithm [36] in the ACADO code generation tool [52] and the PRESAS QP solver [48]. Each of the components in Fig. 9 is executed in a separate ROS node on a single dedicated desktop computer.[3]

B. Experimental Results Using Small-Scale Vehicles

Based on the MIP-DM in Section III, the capabilities of the ego vehicle include lane and lead following, lane selection and lane change execution, swaying maneuvers, queuing behaviors, and stopping/crossing at a traffic intersection. For a qualitative assessment, the behavior in similar scenarios of an architecture as in Fig. 1(a) may be found in [7]. Based on the zone constraints in the MIP (see Section III-D), we implement a traffic rule that allows lane changes only in the bottom-right loop of the figure 8 track [see Fig. 8(c)].

Fig. 10 shows four snapshots of an experiment. The left side of each subfigure shows the location of the ego (blue) and two obstacles (red) on the 8-shaped track, the safety ellipsoid around each obstacle (red dashed line), the NMPC predicted trajectory (cyan cross markers), and the MIP-DM reference trajectory (blue solid circles). The bottom-right side of each subfigure in Fig. 10 illustrates the proposed MIP-DM, i.e., it shows the two-lane road in curvilinear coordinates, the location of the ego (blue), two obstacles (red), the intersection (purple), and the MIP solution trajectory (blue solid circles) over a $T^{\text{mip}} = 4.5$ s horizon length. For each obstacle, the dark red/purple region represents the physical shape, while the larger shaded area corresponds to the avoidance constraints in the MIP-DM that accounts for the ego vehicle shape and margins. A sequence of larger shaded areas is shown for each obstacle based on a prediction of the obstacle behavior over the MIP-DM horizon. The top-right side of each subfigure in Fig. 10 shows the steering angle and velocity command in the NMPC control input trajectory over a $T^{\text{mpc}} = 2$ s horizon.

Fig. 10(a) shows the trajectories for MIP-DM and NMPC at 26 s in the experiment, demonstrating the ego vehicle stopping at the traffic intersection. After the obstacle (Hamster 3) finishes crossing the intersection, the ego crosses at 33 s in the experiment. Fig. 10(b) shows the trajectories at 61 s, demonstrating the ego changing lane and overtaking a slower obstacle to achieve the desired velocity of 0.4 m/s. Fig. 10(c) shows the trajectories at 69 s, demonstrating the ego changing lane back to the preferred lane after overtaking the slower obstacle. Finally, Fig. 10(d) shows the trajectories at 183 s in the experiment, demonstrating the ego queuing behind a slower obstacle because overtaking is not allowed in that zone.

Fig. 11 shows the trace of ego positions (in blue) during the 200-s experiment and each of the locations where the ego vehicle came to a full stop as red dots. The ego vehicle consistently stops at a desired safety distance from the intersection before crossing. The one red dot away from the intersection is due to the queuing behavior in Fig. 10(d).

[3]The desktop for vehicle experiments is equipped with an Intel i7-6900K CPU @ 3.20 GHz ×8 processor, 64-GB RAM, and Ubuntu 16.04 LTS.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

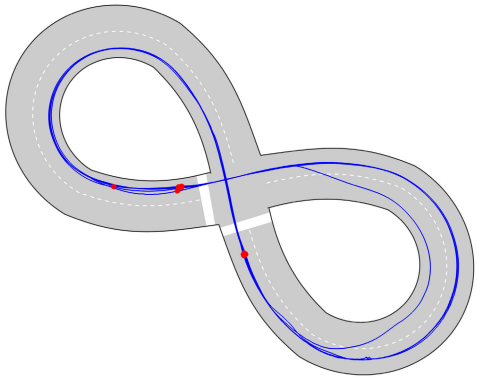


Fig. 11. Trace of ego vehicle positions during experiments in Fig. 10: red dots indicate positions at which the ego stopped, either at the intersection or queuing behind an obstacle.

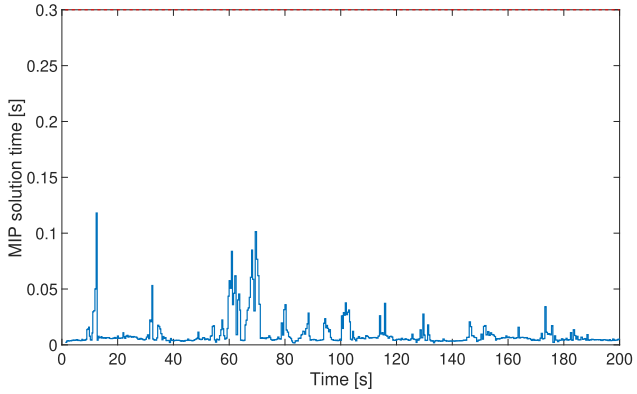


Fig. 12. CPU time of BB-ASIPM for MIP-DM ($T_s^{\text{mip}} = 0.3$ s) in the experiments in Fig. 10.

In addition, Fig. 11 confirms that the ego vehicle only makes lane changes in the bottom right loop of the track, demonstrating the zone-dependent traffic rules in Section III-D. Finally, Fig. 12 shows the CPU times for the BB-ASIPM solver to implement the MIP-DM during the experiment. The computation times are always below 120 ms and therefore real-time feasible, for $T_s^{\text{mip}} = 300$ ms.

VIII. Conclusion and Outlook

We developed an MIP-DM for AD that uses a linear vehicle model in a road-aligned coordinate frame, lane selection and lane change timing constraints, polyhedral collision avoidance and intersection crossing constraints, and zone-dependent traffic rule changes. We leveraged our recently developed embedded BB-ASIPM solver, using a B&B method with reliability branching and warm starting, block-sparse tailored presolve techniques, early termination, and infeasibility detection within an ASIPM. The performance of the MIP-DM method was demonstrated by simulations in various scenarios, including merging points and traffic intersections, and real-time feasibility was demonstrated by HIL simulations on dSPACE Scalexio and MABX-III rapid prototyping units. Finally, we presented results from experiments on a setup with small-scale vehicles, integrating the MIP-DM with an NMPC for reference tracking.

Future works will focus on more advanced behavior prediction models for other vehicles, possibly capturing the interactions between vehicle motion plans in dense traffic, and explicit handling of uncertainty in the modeling and perception of the environment.

References

[1] S. Di Cairano and I. V. Kolmanovsky, "Real-time optimization and model predictive control for aerospace and automotive applications," in *Proc. Annu. Amer. Control Conf. (ACC)*, Jun. 2018, pp. 2392–2409.

[2] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Trans. Intell. Vehicles*, vol. 1, no. 1, pp. 33–55, Mar. 2016.

[3] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annu. Rev. Control*, vol. 45, pp. 18–40, Jan. 2018.

[4] S. Di Cairano, U. V. Kalabic, and K. Berntorp, "Vehicle tracking control on piecewise-clothoidal trajectories by MPC with guaranteed error bounds," in *Proc. IEEE 55th Conf. Decis. Control (CDC)*, Dec. 2016, pp. 709–714.

[5] Y. E. Sahin, R. Quirynen, and S. D. Cairano, "Autonomous vehicle decision-making and monitoring based on signal temporal logic and mixed-integer programming," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 454–459.

[6] K. Berntorp, T. Hoang, R. Quirynen, and S. Di Cairano, "Control architecture design for autonomous vehicles," in *Proc. IEEE Conf. Control Technol. Appl. (CCTA)*, Aug. 2018, pp. 404–411.

[7] H. Ahn, K. Berntorp, P. Inani, A. J. Ram, and S. Di Cairano, "Reachability-based decision-making for autonomous driving: Theory and experiments," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 5, pp. 1907–1921, Sep. 2021.

[8] D. Mayne and J. Rawlings, *Model Predictive Control*. San Francisco, CA, USA: Nob Hill, 2013.

[9] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints," *Automatica*, vol. 35, no. 3, pp. 407–427, Mar. 1999.

[10] T. Marcucci and R. Tedrake, "Mixed-integer formulations for optimal control of piecewise-affine systems," in *Proc. 22nd ACM Int. Conf. Hybrid Syst., Comput. Control*, 2019, pp. 230–239.

[11] B. Landry, R. Deits, P. R. Florence, and R. Tedrake, "Aggressive quadrotor flight through cluttered environments using mixed integer programming," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2016, pp. 1469–1475.

[12] A. D. Pia, S. S. Dey, and M. Molinaro, "Mixed-integer quadratic programming is in NP," *Math. Program.*, vol. 162, nos. 1–2, pp. 225–240, Mar. 2017.

[13] R. Quirynen and S. Di Cairano, "Tailored presolve techniques in branch-and-bound method for fast mixed-integer optimal control applications," *Optim. Control Appl. Methods*, vol. 44, no. 6, pp. 3139–3167, Nov. 2023.

[14] Gurobi Optimization LLC. (2023). *Gurobi Optimizer Reference Manual*. [Online]. Available: www.gurobi.com

[15] MOSEK ApS. (2023). *The MOSEK Optimization Toolbox for MATLAB Manual*. [Online]. Available: www.mosek.com

[16] P. Hespanhol, R. Quirynen, and S. Di Cairano, "A structure exploiting branch-and-bound algorithm for mixed-integer model predictive control," in *Proc. 18th Eur. Control Conf. (ECC)*, Jun. 2019, pp. 2763–2768.

[17] J. Liang, S. D. Cairano, and R. Quirynen, "Early termination of convex QP solvers in mixed-integer programming for real-time decision making," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1417–1422, Oct. 2021.

[18] J. Frey, S. Di Cairano, and R. Quirynen, "Active-set based inexact interior point QP solver for model predictive control," in *Proc. IFAC World Congr.*, 2020, pp. 6522–6528.

[19] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA Urban Challenge: Autonomous Vehicles in City Traffic*, 1st ed., Berlin, Germany: Springer, 2009.

[20] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, 2018.

[21] B. Mirchevska, C. Pek, M. Werling, M. Althoff, and J. Boedecker, "High-level decision making for safe and reasonable autonomous lane changing using reinforcement learning," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 2156–2162.

[22] Q. Liu, X. Li, S. Yuan, and Z. Li, "Decision-making technology for autonomous vehicles: Learning-based methods, applications and future outlook," in *Proc. IEEE Int. Intell. Transp. Syst. Conf. (ITSC)*, Sep. 2021, pp. 30–37.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.

[23] K. P. Wabersich et al., "Data-driven safety filters: Hamilton–Jacobi reachability, control barrier functions, and predictive methods for uncertain systems," *IEEE Control Syst.*, vol. 43, no. 5, pp. 137–177, Oct. 2023.

[24] K. Esterle, P. Hart, J. Bernhard, and A. Knoll, "Spatiotemporal motion planning with combinatorial reasoning for autonomous driving," in *Proc. 21st Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2018, pp. 1053–1060.

[25] A. Richards and J. How, "Mixed-integer programming for control," in *Proc. Amer. Control Conf.*, vol. 4, 2005, pp. 2676–2683.

[26] D. Ioan, I. Prodan, S. Olaru, F. Stoican, and S.-I. Niculescu, "Mixed-integer programming in motion planning," *Annu. Rev. Control*, vol. 51, pp. 65–87, Jan. 2021.

[27] I. Ballesteros-Tolosana, S. Olaru, P. Rodriguez-Ayerbe, G. Pita-Gil, and R. Deborne, "Collision-free trajectory planning for overtaking on highways," in *Proc. IEEE 56th Annu. Conf. Decis. Control (CDC)*, Dec. 2017, pp. 2551–2556.

[28] C. Miller, C. Pek, and M. Althoff, "Efficient mixed-integer programming for longitudinal and lateral motion planning of autonomous vehicles," in *Proc. IEEE Intell. Veh. Symp. (IV)*, Jun. 2018, pp. 1954–1961.

[29] K. Esterle, T. Kessler, and A. Knoll, "Optimal behavior planning for autonomous driving: A generic mixed-integer formulation," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Oct. 2020, pp. 1914–1921.

[30] T. Kessler, K. Esterle, and A. Knoll, "Mixed-integer motion planning on German roads within the Apollo driving stack," *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 851–867, Jan. 2023.

[31] J. Van Brummelen, M. O'Brien, D. Gruyer, and H. Najjaran, "Autonomous vehicle perception: The technology of today and tomorrow," *Transp. Res. C, Emerg. Technol.*, vol. 89, pp. 384–406, Apr. 2018.

[32] K. Berntorp, T. Hoang, and S. Di Cairano, "Motion planning of autonomous road vehicles by particle filtering," *IEEE Trans. Intell. Vehicles*, vol. 4, no. 2, pp. 197–210, Jun. 2019.

[33] R. Quirynen, K. Berntorp, K. Kambam, and S. Di Cairano, "Integrated obstacle detection and avoidance in motion planning and predictive control of autonomous vehicles," in *Proc. Amer. Control Conf. (ACC)*, Jul. 2020, pp. 1203–1208.

[34] G. L. Nemhauser and L. A. Wolsey, *Integer and Combinatorial Optimization*. New York, NY, USA: Wiley-Interscience, 1988.

[35] T. Ersal et al., "Connected and automated road vehicles: State of the art and future challenges," *Vehicle Syst. Dyn.*, vol. 58, no. 5, pp. 672–704, May 2020.

[36] S. Gros, M. Zanon, R. Quirynen, A. Bemporad, and M. Diehl, "From linear to nonlinear MPC: Bridging the gap via the real-time iteration," *Int. J. Control*, vol. 93, no. 1, pp. 62–80, Jan. 2020.

[37] J. N. Hooker and M. A. Osorio, "Mixed logical-linear programming," *Discrete Appl. Math.*, vols. 96–97, pp. 395–442, Oct. 1999.

[38] J. V. Frasch et al., "An auto-generated nonlinear MPC algorithm for real-time obstacle avoidance of ground vehicles," in *Proc. Europ. Control Conf.*, 2013, pp. 4136–4141.

[39] Y. Gao et al., "Spatial predictive control for agile semi-autonomous ground vehicles," in *Proc. 11th Int. Symp. Adv. Vehicle Control*, 2012, pp. 1–6.

[40] X. Qian, F. Altché, P. Bender, C. Stiller, and A. de La Fortelle, "Optimal trajectory planning for autonomous driving integrating logical constraints: An MIQP perspective," in *Proc. IEEE 19th Int. Conf. Intell. Transp. Syst. (ITSC)*, Nov. 2016, pp. 205–210.

[41] R. Rajamani, *Vehicle Dynamics and Control*. New York, NY, USA: Springer, 2012.

[42] K. Berntorp, B. Olofsson, K. Lundahl, and L. Nielsen, "Models and methodology for optimal trajectory generation in safety-critical road–vehicle manoeuvres," *Vehicle Syst. Dyn.*, vol. 52, no. 10, pp. 1304–1332, Oct. 2014.

[43] N. Suriyarachchi, R. Quirynen, J. S. Baras, and S. Di Cairano, "Optimization-based coordination and control of traffic lights and mixed traffic in multi-intersection environments," in *Proc. Amer. Control Conf. (ACC)*, May 2023, pp. 3162–3168.

[44] Y. Chen, U. Rosolia, C. Fan, A. Ames, and R. Murray, "Reactive motion planning with probabilisticsafety guarantees," in *Proc. Conf. Robot Learn.*, 2021, pp. 1958–1970.

[45] T. Achterberg, T. Koch, and A. Martin, "Branching rules revisited," *Oper. Res. Lett.*, vol. 33, no. 1, pp. 42–54, Jan. 2005.

[46] A. Bemporad and V. V. Naik, "A numerically robust mixed-integer quadratic programming solver for embedded hybrid model predictive control," in *Proc. 6th IFAC NMPC Conf.*, 2018, pp. 412–417.

[47] T. Marcucci and R. Tedrake, "Warm start of mixed-integer programs for model predictive control of hybrid systems," *IEEE Trans. Autom. Control*, vol. 66, no. 6, pp. 2433–2448, Jun. 2021.

[48] R. Quirynen and S. Di Cairano, "PRESAS: Block-structured preconditioning of iterative solvers within a primal active-set method for fast model predictive control," *Optim. Control Appl. Methods*, vol. 41, no. 6, pp. 2282–2307, Nov. 2020.

[49] R. Fletcher, *Practical Methods of Optimization*, 2nd ed., Hoboken, NJ, USA: Wiley, 1987.

[50] Cogniteam. (2018). *The Hamster*. Accessed: Jan. 8, 2018. [Online]. Available: www.cogniteam.com/hamster5.html

[51] Optitrack. (2018). *Prime 13 Motion Capture*. Accessed: Jan. 23, 2018. [Online]. Available: http://optitrack.com/products/prime-13

[52] R. Quirynen, M. Vukov, M. Zanon, and M. Diehl, "Autogenerating microsecond solvers for nonlinear MPC: A tutorial using ACADO integrators," *Optim. Control Appl. Methods*, vol. 36, no. 5, pp. 685–704, Sep. 2015.

Rien Quirynen received the bachelor's degree in computer science and electrical engineering and the master's degree in mathematical engineering from KU Leuven, Leuven, Belgium, in 2010 and 2012, respectively, and the joint Ph.D. degree from KU Leuven and the University of Freiburg, Freiburg im Breisgau, Germany, in 2016.

He worked as a Senior Research Scientist at Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, from early 2017 until late 2023. Currently, he is a Staff Software Engineer at Stack AV, Pittsburgh, PA, USA. He has authored and co-authored more than 75 peer-reviewed papers in journals and conference proceedings. He holds 25 patents. His research focuses on numerical optimization algorithms for decision-making, motion planning, and predictive control of autonomous systems.

Dr. Quirynen received the four-year Ph.D. Scholarship from the Research Foundation–Flanders (FWO) for the term of 2012–2016. He serves as an Associate Editor for *Optimal Control Applications and Methods* (Wiley) and the IEEE CCTA Editorial Board.

Sleiman Safaoui (Member, IEEE) received the B.S. degree in electrical engineering from The University of Texas at Dallas, Richardson, TX, USA, in 2019, and the M.S. and Ph.D. degrees in electrical engineering from the Control, Optimization, and Networks Laboratory (CONLab), The University of Texas at Dallas, in 2023.

From August 2021 to March 2022, he was an Intern at Mitsubishi Electric Research Laboratories (MERL), Cambridge, MA, USA, where he worked on ground and aerial vehicle autonomy research projects. He was a Research Associate with CONLab in 2024. In June 2024, he joined Exponent, Phoenix, AZ, USA, where he is currently an Associate in electrical engineering and computer science. His research interests include robot planning and control under uncertainty, autonomous vehicles, advance driver-assistance systems (ADASs), and multirobot systems.

Stefano Di Cairano (Senior Member, IEEE) received the master's (Laurea) and Ph.D. degrees in information engineering from the University of Siena, Siena, Italy, in 2004 and 2008, respectively.

From 2008 to 2011, he was with Powertrain Control R&A, Ford Research and Advanced Engineering, Dearborn, MI, USA. Since 2011, he has been with Mitsubishi Electric Research Laboratories, Cambridge, MA, USA, where he is currently the Deputy Director and a Distinguished Research Scientist. He has authored and co-authored more than 250 peer-reviewed papers in journals and conference proceedings. He holds 80 patents. His research focuses on optimization-based control and decision-making strategies for automotive, factory automation, transportation systems, and aerospace. His research interests include model predictive control, constrained control, path planning, hybrid systems, optimization, and particle filtering.

Dr. Di Cairano was the Chair of the IEEE CSS Technical Committee on Automotive Controls and the IEEE CSS Standing Committee on Standards. He is the inaugural Chair of the IEEE Technology Conferences Editorial Board. He was an Associate Editor of IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY.

Authorized licensed use limited to: University of Science & Technology of China. Downloaded on October 13,2025 at 10:08:35 UTC from IEEE Xplore. Restrictions apply.