2021 IEEE International Conference on Robotics and Automation (ICRA 2021)
May 31 - June 4, 2021, Xi'an, China

Lane-free Autonomous Intersection Management: A Batch-processing Framework Integrating Reservation-based and Planning-based Methods

Bai Li, Youmin Zhang, Tankut Acarman, Yakun Ouyang, Cagdas Yaman, and Yaonan Wang

Abstract—**Autonomous intersection management (AIM) refers to planning the trajectories for multiple connected and automated vehicles (CAVs) when they traverse an unsignalized intersection cooperatively. As an extension of the conventional AIM, lane-free AIM allows the CAVs to adjust their velocities and paths flexibly within the intersection. Nominally, one needs to formulate a centralized optimal control problem (OCP) to describe the concerned lane-free AIM scheme, but solving such an intractably scaled problem is challenging. This work proposes a batch-processing framework, which divides the traffic flow into batches. The cooperative trajectories within one batch are planned by numerically solving a small-scale OCP; all the batches are managed via a reservation-based method following the first-come-first-serve policy. The proposed batch-processing framework aims to run as fast as a reservation-based method at the macro level while taking care of the cooperative driving quality at the micro level. The proposed method is validated via simulation and preliminary experiments.**

I. Introduction

Connected and automated vehicles (CAVs) are important elements to form an intelligent transportation system [1]. With the developments in fast communication and high-quality planning modules, CAVs are promising to have sufficient cooperative driving ability [2–4]. Intersections are the main source of congestion in an urban transportation system [5]. An intersection is a typical scenario that reflects the underlying conflicts among the vehicles. This paper is focused on the multi-CAV cooperative trajectory planning in an unsignalized intersection, which is also known as autonomous intersection management (AIM) [6–9]. Despite the large number of studies in AIM, the cooperation capability of CAVs deserves to be further exploited.

A. Related Works

AIM studies are primarily classified as reservation-based and planning-based methods [10–13]. A reservation-based method runs fast, but the solution quality is not high. Conversely, a planning-based method generates high-quality cooperative trajectories while the computation is usually overburdened.

The reservation-based methods originated from the pioneering work of Dresner and Stone [13], which held the first-come-first-serve strategy to specify the trajectory of each single vehicle *sequentially*. The planning-based methods, on the other hand, generate the cooperative trajectories of multiple CAVs *simultaneously* [14]. Mirheli et al. [15] described the AIM scheme as a mixed-integer linear programming (MILP) problem and solved it via Monte Carlo Tree Search. Another work done by Mirheli et al. [16] is an iterative framework to determine the velocity of each vehicle along a given path, and the velocity is determined through solving a mixed-integer nonlinear programming (MINLP) problem. The iteration continues until consensus is achieved. Similarly, Levin and Rey [17] derived the passing order and intersection entrance time simultaneously for all of the CAVs by solving an MILP problem. Xu et al. [18] projected the vehicles into a virtual lane, determined the passing order via a geometry topology method, and then proposed a distributed controller to achieve conflict-free cooperation in the intersection. Malikopoulos et al. [19] decoupled the multi-vehicle velocity planning scheme and provided analytical solutions to each decoupled subproblem. Zohdy et al. [20] defined two preparation zones for each CAV before it enters the intersection and adopted a simulator-in-the-loop optimizer to plan the velocity. Kamal et al. [21] handled the cooperative velocity planning and passing order decision scheme via a model predictive control (MPC) approach. Similar works include [22–24].

The aforementioned methods share some common limitations. First, trajectory planning is degraded to velocity planning along a fixed path or path planning with specified velocity. Despite the fact that such a degradation makes the AIM scheme easier, the driving flexibility of each CAV is not fully activated. Second, the CAVs are not allowed to change their lanes flexibly when they traverse in the intersection.

To leverage the CAVs' cooperation potential better, the concept of lane-free AIM has been considered [8,25,26], which microscopically describes the intersection as a free space. However, an intractably scaled nonlinear programming (NLP) problem needs to be solved [27,28], thus making the existing lane-free AIM methods only able to handle tens of CAVs at the cost of thousands of CPU seconds.

B. Contributions

This work focuses on the lane-free AIM scheme. In contrast to the existing lane-free AIM methods, this work particularly cares about the following two aspects: 1) a long-lasting traffic

* Research supported by the Fundamental Research Funds for the Central Universities (No. 531118010509), the National Natural Science Foundation of China (No. 61833013), the Natural Sciences and Engineering Research Council of Canada (RGPIN-2017-06680), and the Galatasaray University Scientific Research Support (No. 19.401.005).

Bai Li and Yakun Ouyang are with the College of Mechanical and Vehicle Engineering, Hunan University, Changsha, China (e-mail: libai@zju.edu.cn, libai@hnu.edu.cn; yakun@hnu.edu.cn).

Youmin Zhang is with the Department of Mechanical, Industrial and Aerospace Engineering, Concordia University, Montreal, Canada (e-mail: ymzhang@encs.concordia.ca).

Tankut Acarman and Cagdas Yaman are with the Computer Engineering Department, Galatasaray University, Istanbul, Turkey (e-mail: tacarman @gsu.edu.tr, truckercy@hotmail.com).

Yaonan Wang is with the School of Robotics, Hunan University, Changsha, China (e-mail: yaonan@hnu.edu.cn).

flow rather than a small group of CAVs is dealt with; and 2) the CPU time should be short enough for online applications. The contribution of this work is the proposal of a batch-processing framework integrating the strengths of reservation-based and planning-based methods. Concretely, the CAVs in a continuous traffic flow are classified into batches, wherein each batch contains a small number of CAVs. The cooperative trajectories within one batch are microscopically optimized by a planning-based method while the batches are handled macroscopically in a reservation-based strategy.

C. Organization

In the rest of this paper, Section II introduces the batch-processing framework and all its technical details. Simulation and experimental results are reported and discussed in Section III, followed by the conclusions drawn in Section IV.

II. Batch-processing based Lane-free AIM Method

The lane-free AIM scheme is inherently identical to planning the cooperative trajectories for a large number of CAVs, which intend to traverse a free-space intersection. As depicted in Fig. 1(a), we temporarily assume that 1) each leg of the intersection is sufficiently long; and 2) the initial driving status and traverse intention of each CAV are known *a priori*. Nominally, this problem should be described as a large-scale centralized optimal control problem (OCP) to minimize delays in the intersection, subject to boundary-value conditions, vehicle kinematics, and exterior limitations from the environment. Instead of solving the OCP directly, a batch-processing method is introduced in Sections II.A–II.D. Thereafter, Section II.E introduces how to remove the aforementioned assumptions so that the proposed method is applicable to real-world cases.

A. Lane-free AIM Problem Statement

The concerned lane-free AIM problem is nominally formulated as a centralized OCP:

$$\begin{aligned} &\text{Minimize } J, \\ &\text{subject to} \\ &g_{\text{kinematics}}\left(\mathbf{x}(t),\ \mathbf{u}(t)\right)=0, \\ &\mathbf{x}(0)=\mathbf{x}_{\text{init}},\ \mathbf{u}(0)=\mathbf{u}_{\text{init}}, \\ &g_{\text{end}}\left(\mathbf{x}(t_{\text{f}}),\ \mathbf{u}(t_{\text{f}})\right)\le 0, \\ &g_{\text{collision}}\left(\mathbf{x}(t)\right)\le 0,\ t\in[0,t_{\text{f}}]. \end{aligned} \tag{1}$$

Herein, t denotes time, t_{f} denotes the moment that the last CAV traverses the intersection, $\mathbf{x}$ contains the state profiles of all the CAVs, $\mathbf{u}$ denotes the corresponding control profiles, $g_{\text{kinematics}}=0$ stands for the kinematic principle of each CAV; $[\mathbf{x}_{\text{init}}, \mathbf{u}_{\text{init}}]$ contains the initial status of each CAV at $t=0$, which is assumed to be fully known *a priori*; g_{end} contains the end-point restrictions imposed on the CAVs, $g_{\text{collision}}$ stands for the vehicle-to-environment and inter-vehicle collision avoidance constraints, and the cost function J contains the demands of time efficiency, comfort, energy efficiency, etc. The details may be found in [26].

The dimension of (1) is extremely large if thousands of CAVs are involved in the AIM activity. Under such conditions, directly solving (1) is impossible due to being computationally overburdened. A batch-processing method is therefore introduced to decouple the nominal problem (1) so that it can be handled in a tractable way.

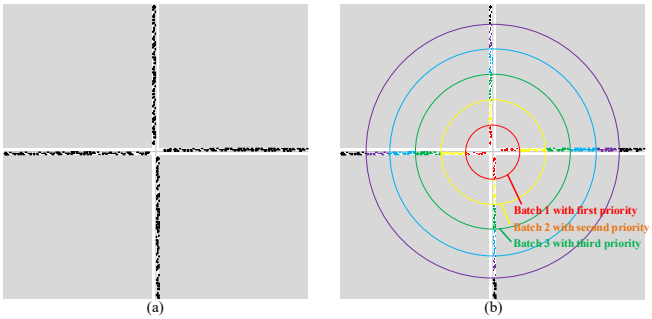


Fig. 1. Scenario setup: (a) an intersection with sufficiently long legs so that the number of CAVs can be large; (b) batch division of the traffic flow. This figure can be seen more clearly if zoomed in.

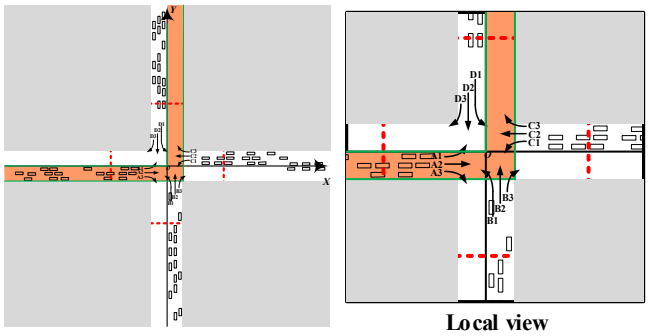


Fig. 2. Schematic diagram of intersection coordinate frame *XOY*, Threshold Line, CAV categories, and accessible region for A1-type vehicles.

B. Overall Batch-processing Framework

All the CAVs are classified into subgroups according to their distances to the intersection center. In the rest of this paper, we call each subgroup a *batch*. The CAVs in each batch run cooperatively, and all the batches are prioritized so that a lower-priority batch yields to the ones with higher priorities. As depicted in Fig. 1(b), a batch closer to the intersection center has a higher priority.

Each batch's motions during $[0,t_{\text{f}}]$ are classified into 3 stages. A batch prepares to enter the intersection at stage 1, travels through the intersection at stage 2, and continues to cruise ahead during stage 3 after leaving the intersection. As shown in Fig. 2, stage 1 begins from $t=0$ and ends at the moment that the first CAV in the batch touches the *Threshold Line* (plotted as dashed red lines); stage 2 lasts for a fixed period t_{stage2}, which is long enough so that all of the CAVs within the batch can traverse the intersection; stage 3 follows stage 2 tightly and continues until $t=t_{\text{f}}$.

Like [26], we require that the CAVs should run stably at the initial and terminal moments of each stage. Herein, the stability denotes that a CAV 1) runs with v_{norm}, 2) has zero acceleration/jerk, 3) has zero steering angle and its derivative, and 4) travels in the direction of the road. These stability conditions ensure that each CAV safely cruises without any collision risk during its stage 3 period. Imposing the stability conditions also makes the first two stages independent. The independence enables us to plan the cooperative trajectories for stage 2 *prior to* stage 1, and ensures that the computational burden at each stage is very light. Section II.C introduces how one batch traverses the intersection at stage 2, and Section

II.D introduces how it runs during stage 1. Throughout stage 3, each batch is requested to *maintain* the aforementioned stable driving status so that collisions are definitely avoided.

C. Planning-based Driving at Stage 2

This subsection introduces how to plan the cooperative trajectories for a single batch during its stage 2, i.e., from the moment that the first CAV within the batch touches the Threshold Line (denoted as t_{init2}) until $t = t_{\text{init2}} + \text{t}_{\text{stage2}}$. For simplicity, the terminal moment of stage 2, i.e., $t_{\text{init2}} + \text{t}_{\text{stage2}}$, is denoted as t_{end2}.

OCP Model Formulation

A small-scale centralized OCP is formulated to describe the cooperative trajectory planning scheme at stage 2 for a single batch consisting of N_V CAVs. The OCP is written as

$$\begin{aligned} &\text{Minimize } \wp\big(\mathbf{x}(t_{\text{end2}})\big), \\ &\text{subject to} \\ &g_{\text{kinematics}}\big(\mathbf{x}(t),\ \mathbf{u}(t)\big) = 0, \\ &g_{\text{collision}}\big(\mathbf{x}(t)\big) \le 0,\ t \in [t_{\text{init2}}, t_{\text{end2}}]; \\ &[\mathbf{x}(t_{\text{init2}}), \mathbf{u}(t_{\text{init2}})] = [\mathbf{x}_{\text{init2}}, \mathbf{u}_{\text{norm}}]; \\ &[\mathbf{x}(t_{\text{end2}}), \mathbf{u}(t_{\text{end2}})] = [\mathbf{x}_{\text{end2}}, \mathbf{u}_{\text{norm}}]. \end{aligned} \tag{2}$$

With a slight abuse of notation, $\mathbf{x}$ and $\mathbf{u}$ refer to the state/control profiles of the currently concerned CAVs within one batch. Recall that stage 2 begins with the moment when the first vehicle in the batch touches the Threshold Line, we can specify the batch's driving status at t_{init2} (how to determine t_{init2} is introduced in the next subsection) and t_{end2}. Fixing the duration of stage 2 to t_{stage2} makes (2) easier to solve. The pursuit for traversing the intersection quickly is reflected by the cost function $\wp$, which will be introduced later.

The state of each CAV at t_{end2} is set as per its own traverse intention. To better explain this point, we enumerate all the possible traverse options as 12 categories by the directions one CAV enters and exits the intersection (see the 12 labels marked A1, A2, …, D3 in Fig. 2). Regarding an A1-type vehicle i, it should run stably along the target road at t_{end2}:

$$\begin{aligned} &[\theta_i(t_{\text{end2}}), v_i(t_{\text{end2}}), \phi_i(t_{\text{end2}}), a_i(t_{\text{end2}}), jerk_i(t_{\text{end2}}), \omega_i(t_{\text{end2}})] = \\ &[\frac{\pi}{2}, \text{v}_{\text{norm}}, 0, 0, 0, 0]. \end{aligned} \tag{3}$$

Herein, θ_i denotes the orientation angle, v_i denotes the velocity, ϕ_i denotes the steering angle, a_i represents the acceleration, $jerk_i$ denotes the derivative of a_i, ω_i stands for the steering angle rate, and v_{norm} denotes the standard velocity mentioned in Section II.B.

$g_{\text{collision}} \le 0$ contains the inter-vehicle and vehicle-to-barrier collision avoidance constraints *only* related to the current batch. Collisions between the current batch and other batches are avoided by the efforts made in stage 1, which are introduced in Section II.D.

Let us focus on the vehicle-to-barrier collision avoidance constraints in $g_{\text{collision}} \le 0$. Each CAV is only allowed to driving in partial region of the whole intersection. For example, an A1-type CAV is restricted to traverse in the orange region depicted in Fig. 2. Conventional works [27,28] considered the vehicle-to-barrier constraints as not hitting the green barriers of the orange region, which are non-convex and almost non-differentiable because the orange region contains a sharp corner. This work proposes a unified formulation to describe the vehicle-to-barrier constraints. As illustrated in Fig. 3, for each A1-type CAV i, there exists at least one scalar $\gamma_i \in [0,100]$ such that the vehicle stays in Tunnel 1 during the first γ_i% of $[t_{\text{init2}}, t_{\text{end2}}]$, and stays in Tunnel 2 during the rest $(100 - \gamma_i)$ percentage. Since either Tunnel 1 or 2 is a regularly placed box region, requiring a rectangular vehicle to stay in the tunnels can be modeled more easily than requiring it to avoid hitting the green lines in Fig. 2 [29,30]. The way to specify γ_i will be introduced later. The collision avoidance constraints regarding the other 11 types of vehicles can be formulated in a similar way.

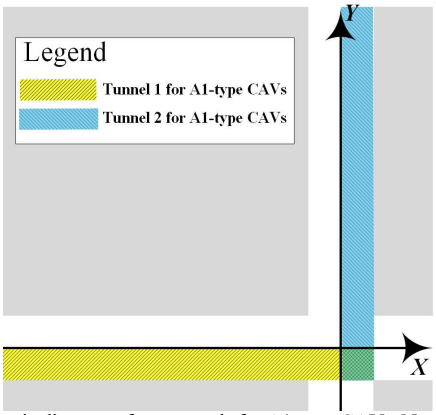


Fig. 3. Schematic diagram of two tunnels for A1-type CAVs. Note that the two tunnels overlap within the intersection.

The cost function $\wp(\mathbf{x}(t_{\text{end}}))$ encourages each CAV to travel farther in its target direction after exiting the intersection. For an A1-type vehicle i, the related term in $\wp$ is $-y_i(t_{\text{end}})$, wherein y_i denotes the Y-axis coordinate in the XOY frame (Fig. 2). This term is also used by a B2- or C3-type CAV. Writing the other 9 types of vehicles' terms in the cost function $\wp$ in a similar way yields

$$\begin{aligned} \wp(\mathbf{x}(t_{\text{end}})) = &\sum_{i \in \text{A1}\cup\text{B2}\cup\text{C3}} -y_i(t_{\text{end2}}) + \sum_{i \in \text{A2}\cup\text{B3}\cup\text{D1}} -x_i(t_{\text{end2}}) + \\ &\sum_{i \in \text{A3}\cup\text{C1}\cup\text{D2}} y_i(t_{\text{end2}}) + \sum_{i \in \text{B1}\cup\text{C2}\cup\text{D3}} x_i(t_{\text{end2}}). \end{aligned} \tag{4}$$

Herein, with abuses of notations, A1, A2, …, D3 denote the sets that contain the IDs of CAV types.

Numerical Solution to OCP

A numerical solution to (2) is derived by discretizing it into an NLP, then solving the NLP via an NLP solver [31]. An incrementally constrained optimization (ICO) framework is proposed to determine γ_i and to facilitate the solution process.

As a preliminary step, we formulate a slack OCP which is the same as (2) except that the vehicle-to-barrier collision avoidance constraints in $g_{\text{collision}} \le 0$ are relaxed. Taking the aforementioned A1-type CAV i as an example, we request it to stay in Tunnel 1 during $t \in [t_{\text{init2}}, t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot \gamma_i\%]$ and to stay in Tunnel 2 during $[t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot \gamma_i\%,\ t_{\text{end2}}]$. But in the slack OCP, CAV i is requested to stay in Tunnel 1 only during $[t_{\text{init2}}, t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot (\gamma_i - \text{gap})\%]$ and to stay in Tunnel 2 during $[t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot (\gamma_i + \text{gap})\%,\ t_{\text{end2}}]$. The parameter gap creates an intermediate interval $[t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot (\gamma_i - \text{gap})\%,\ t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot$

$(\gamma_i + \text{gap})\%]$ such that the vehicle i needs not stay in either tunnel. The absence of tunnel restriction during the intermediate interval makes the slack OCP easier than (2). Although no restrictions are imposed during the intermediate interval, the resultant trajectory would not violate the within-tunnel conditions much because the CAV leaves from Tunnel 1 and still needs to travel in Tunnel 2 right after the intermediate interval. ICO solves the slack OCP iteratively with gap gradually reducing from gap_0 to 0. The overall principle of ICO is presented in Alg. 1.

Alg. 1: General structure of ICO

Input: Initial/Target configuration of each CAV, and scenario setup;
Output: Cooperative trajectories for the current batch of CAVs at stage 2;

1. $\boldsymbol{\chi} \leftarrow \text{GenerateInitialGuess}()$;
2. $S \leftarrow \text{Linspace}(\text{gap}_0, 0, \text{N}_{\text{step}})$;
3. **for each** $\text{gap} \in S$, **do**
4. $\{\gamma_i\} \leftarrow \text{UpdateGamma}(\boldsymbol{\chi})$;
5. $\text{NLP}_{\text{slack}} \leftarrow \text{UpdateNLP}(\{\gamma_i\}, \text{gap})$;
6. $\boldsymbol{\chi} \leftarrow \text{SolveNLP}(\text{NLP}_{\text{slack}}, \boldsymbol{\chi})$;
7. **end for**
8. **return** $\boldsymbol{\chi}$.

In line 1 of Alg. 1, an initial guess is generated via the function GenerateInitialGuess(). The initial guess is a solution vector that contains the collocation points for **x** and **u** of all the CAVs in the current batch. A quick initial guess is derived by combining N_V single-vehicle trajectories, each of which is simply formed by tracking the reference line with the nominal cruising velocity v_norm. GenerateInitialGuess does not consider the inter-vehicle collisions, which will be dealt with in the iterative computation process of ICO. $\text{Linspace}(\text{gap}_0, 0, \text{N}_{\text{step}})$ in line 2 generates an arithmetic progression consisting of N_{step} scalars from gap_0 to 0. Between lines 3–7, the iterative computation process continues until gap decreases from gap_0 to 0. In each iteration, an estimation of each γ_i is made via the function UpdateGamma(), and the estimated $\{\gamma_i\}$ would be used for formulating the slack NLP problem, which is numerically solved thereafter via SolveNLP(). The derived optimum is recorded in $\boldsymbol{\chi}$ for future usage in the next iteration. Particularly in $\text{UpdateGamma}(\boldsymbol{\chi})$, each γ_i is independently determined. Suppose that by checking $\boldsymbol{\chi}$ one finds that an A1-type CAV i stays in Tunnel 1 during $[t_{\text{init2}}, a]$ and stays in Tunnel 2 during $[b, t_{\text{end2}}]$, then γ_i is specified according to

$$\frac{a+b}{2} = t_{\text{init2}} + \text{t}_{\text{stage2}} \cdot \gamma_i. \tag{5}$$

When $\{\gamma_i\}$ and gap are updated in the next iteration, a new slack NLP problem is formulated using $\text{UpdateNLP}(\{\gamma_i\}, \text{gap})$. Regarding $\text{SolveNLP}(\text{NLP}_{\text{slack}}, \boldsymbol{\chi})$, this work adopts the interior-point method (IPM) to solve each $\text{NLP}_{\text{slack}}$, wherein $\boldsymbol{\chi}$ serves as the initial guess for warm starting. We choose IPM because it is stable and efficient in dealing with large-scale NLP problems [32,33].

Providing a reasonable $\{\gamma_i\}$ *a priori* is important in solving (2), but it is not easy to estimate $\{\gamma_i\}$ accurately. Instead of handling (2) directly, we build an iterative framework in ICO to gradually increase the estimation accuracy of $\{\gamma_i\}$. The introduction of gap makes the iterative solution process free from being misled by an inappropriate decision of $\{\gamma_i\}$. As gap decreases towards 0, the estimation of $\{\gamma_i\}$ gets more accurate gradually. Finally, solving $\text{NLP}_{\text{slack}}$ with $\text{gap} = 0$ is identical to solving (2) numerically.

Recall that the collisions between the current batch and the batches with higher priorities are not considered at stage 2. Such collisions are avoided due to the efforts made at stage 1, which will be introduced in the next subsection.

D. Reservation-based Driving at Stage 1

The preceding subsection has introduced how each batch of CAVs moves at its stage 2. It is still not clear when the batch would reach the Threshold Line. As it is already known how the current batch runs at stage 2, a reservation-based method can be deployed to reserve an arrival time for the current batch to begin stage 2.

Recall that each CAV is assumed to run at the nominal velocity v_norm at the beginning of stage 1. We first check if collisions with the higher-priority batches would happen if the current batch moves with v_norm constantly during stage 1. If this constant-velocity cruising mode does not cause any collisions at stage 2, then it is accepted; otherwise, the time that the current batch reaches the Threshold Line is slightly postponed by ΔT, then the underlying collision occurrence at stage 2 is checked again. This postpone-and-check process continues until the resultant arrival time makes stage 2 free from collisions. In this way, the time for the first vehicle in the batch to reach Threshold Line, i.e. t_{init2}, is determined.

When t_{init2} is specified, the next step is to plan trajectories for the current batch during stage 1, i.e. $t \in [0, t_{\text{init2}}]$. Note that t_{init2} may be rather large if the current batch is initially distant from the intersection center. Throughout stage 1, the CAVs are only allowed to driving along their reference lines with $\phi_i(t) \equiv 0$ $(\forall t \in [0, t_{\text{init2}}], \forall i)$. More importantly, we request that all the CAVs in one batch must accelerate or decelerate at the same pace during stage 1 so that the batch formation *never* alters until t_{init2}. Planning the trajectories at stage 1 is identical to solving a fixed-time OCP (denoted as $\text{OCP}_{\text{stage1}}$). Since all the CAVs in the current batch run at a same pace at stage 1, the dimension of $\text{OCP}_{\text{stage1}}$ is fully independent of N_V. The concerned $\text{OCP}_{\text{stage1}}$ is a simple problem like how a mass point travels along the 1-dim number axis. The cost function of $\text{OCP}_{\text{stage1}}$ may be set quadratically about travel comfort and energy consumption. $\text{OCP}_{\text{stage1}}$ is discretized into a quadratic programming (QP) problem and then solved via a QP solver.

Postponing the arrival time of Threshold Line requires the current batch to decelerate for some time during stage 1, which may cause collisions with the lower-priority batches behind it. If collisions occurrence between the current batch and a lower-priority one during stage 1 is detected, then the involved lower-priority batches keep the same deceleration pace during stage 1 of the current batch; otherwise, the lower-priority batches remain cruising with v_norm. In addition to influencing the lower-priority batches, the current batch may need to yield to the higher-priority batches as well. The technical details are omitted here.

E. Link to Real-world Intersection Situation

The aforementioned subsections have introduced how to

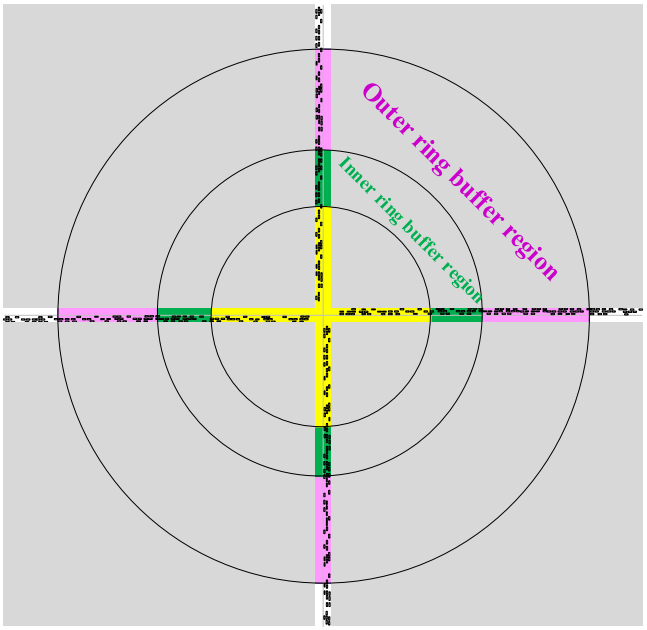


Fig. 4. Schematic diagram of outer and inner ring-like buffer zones established to make the proposed batch-processing method capable of handling a real-world AIM scheme. This figure can be zoomed in to see more clearly.

manage an extremely large flow of CAVs under the all-known assumption. But it is unrealistic to know the information of all the CAV batches simultaneously. Also, the velocity may not be v_{norm} when each CAV's control is taken over by the AIM system. To cater for the real-world cases, two ring-like buffer zones, as depicted in Fig. 4, are established. In the outer ring, each vehicle gets connected with the AIM system, gets its control taken over, and adjusts its driving status to meet the stable entrance condition mentioned in Section II.B. We request that the outer ring is sufficiently wide so that the driving status adjustment can complete. In the inner ring, each CAV continues to move on with constant velocity v_{norm}. Meanwhile, the centralized system divides the batch, plans the cooperative trajectories for stage 2, and then manages the motions for stage 1. We have to request that the inner ring is also sufficiently wide so that the computation completes before the CAV leaves the inner ring.

Recall that the planning problem for stage 2 is in a small scale, thus solving it numerically in a high-performance workstation is not slow. The repeated collision checks requested by the reservation-based method can be accelerated by trying multiple values of ΔT in parallel and checking collisions for multiple vehicles in parallel. Therefore, the reservation-based method runs fast in a multicore workstation. Numerically solving OCP_{stage1}, i.e., solving a small-scale QP problem, is also fast. To summarize, the entire proposed architecture is promising to meet the real-time computation demand.

The batch-processing framework holds a reservation-based strategy to address the batch-to-batch conflicts at stage 1, and utilizes a planning-based strategy to generate cooperative trajectories within one batch for stage 2. Decoupling all of the CAVs into batches enables each batch to focus on itself without worrying about others when it is close to the intersection. Via the batch-processing strategy, most of the intricate collision avoidance constraints in the nominal problem (1) are replaced by collision checks conducted in the reservation-based method. This replacement is undoubtedly beneficial because collision checking is easier than handling the constraints during the NLP solution process.

III. Simulations, Experiments, and Discussions

A. Simulation Setups

The proposed batch-processing framework is evaluated via simulations, which were executed on an i9-9900 CPU with 32 GB RAM that runs at 3.10 × 2 GHz. An open-source package of IPM, namely IPOPT [32], is applied in the MATLAB + AMPL environment [34].

An intersection scenario is set up, wherein each of the eight branches connected to the intersection contains 3 lanes. Also, we set $t_{stage2} = 10$ s, $\Delta T = 0.05$ s, $gap_0 = 5$, and $v_{norm} = 20$ m/s. The Threshold Line keeps 40 m from the intersection center. The division of each batch ensures that each batch contains N_V CAVs.

Regarding how to evaluate the travel efficiency, some existing methods provide misleadingly good results because the delays are postponed *outside* the intersection region so that the traffic throughput within or near the intersection appears to be high [10]. To subjectively measure the throughput, 2,000 vehicles are assigned to approach the intersection from the four directions in a medium density. We measure the duration between the moment that the first CAV enters the intersection and the moment that the last one leaves the intersection.

B. On the Efficacy of Batch-processing Framework

We have tested the proposed batch-processing framework under various settings of N_V. The collected comparison results are listed in Table I. Typical simulation results are presented at https://www.bilibili.com/video/bv1ct4y1e7ks.

TABLE I. Simulation Results with Various N_V

N_V	Throughput [s]	Time for stage 1 [s]	Time for stage 2 [s]
2	604.255	0.057	0.113
5	675.882	0.188	0.369
8	599.734	0.369	1.613
10	542.381	0.527	2.348
15	434.857	0.975	5.742

Intuitively we expect the throughput would decrease with the growth of N_V. However, according to Table I, the throughput first increases then declines with the growth of N_V. The rationale behind this interesting phenomenon is presented as follows. When the scale of each batch is small, e.g. $N_V = 2$, the cooperation near the intersection center is not sufficient because each CAV only runs cooperatively with one companion. In spite of this, the two vehicles' trajectories do not occupy much space of the intersection during their stage 2, thus the reserved arrival time is not much delayed. Summarizing these two factors, the throughput is not large. When $N_V = 5$, the cooperation among the vehicles in each batch is improved. At the same time, the five vehicles in each batch occupy the intersection sparsely, thereby causing more delays in reserving an arrival time. The throughput increases because the delayed arrival time is the dominant factor. When N_V becomes even larger, the high-quality cooperation at stage

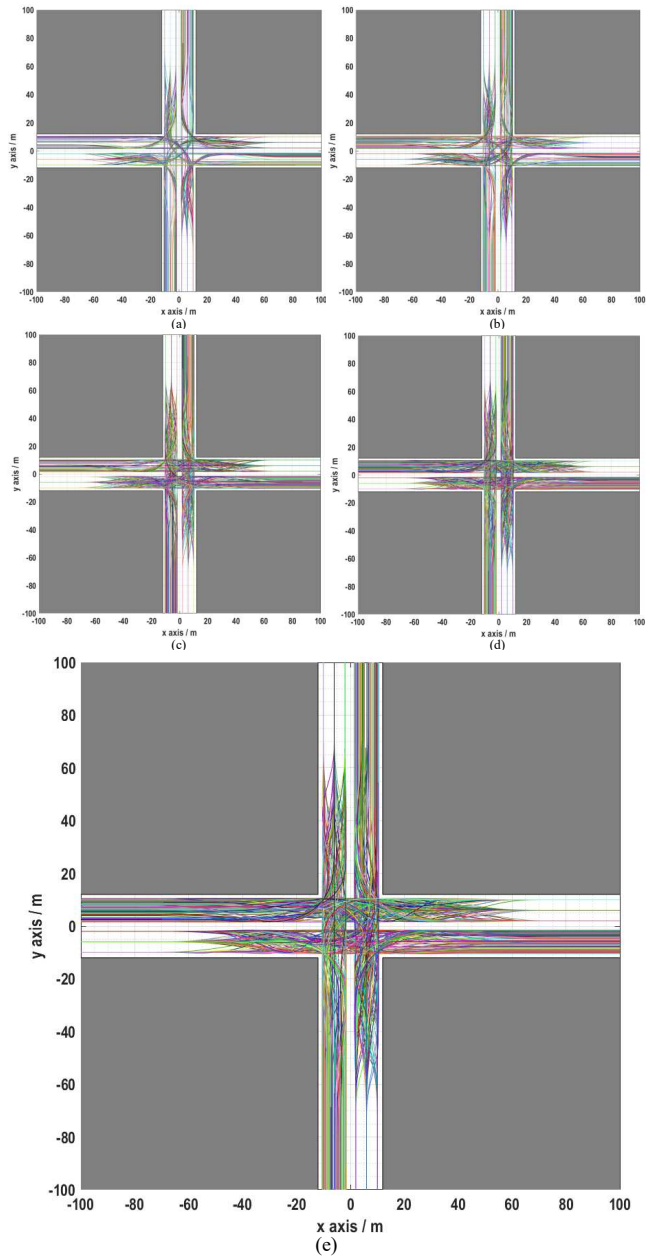
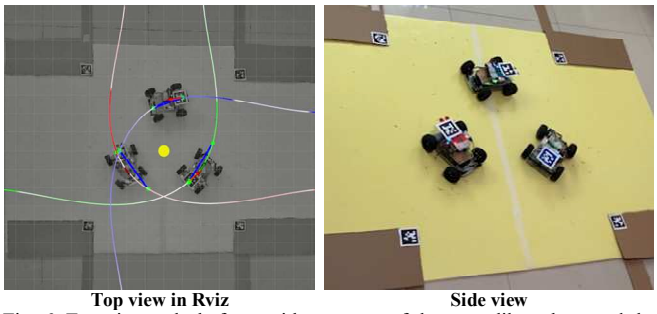


Fig. 5. Footprints of CAVs' trajectories computed by the batch-processing framework: (a) $N_V = 2$; (b) $N_V = 5$; (c) $N_V = 10$; (d) $N_V = 15$; (e) an enlarged view of (d).



Top view in Rviz Side view

Fig. 6. Experimental platform with a swarm of three car-like robots and the test performance.

2 dominates to reduce the delays at stage 2, thus the gross delay, i.e. the throughput declines with N_V. The footprints of the traffic flow are depicted in Fig. 5 for typical selections of N_V, which supports the aforementioned analysis.

Table I also reports the average CPU time spent on computing the motions for either stage of each batch. Even under large N_V conditions, the CPU time is not overly long and will be further reduced if the codes are optimized or if a better processer is used. With the development of better processers, setting N_V larger and larger is possible. This indicates that our proposed batch-processing framework is a promising method that always has chances to do better in the future.

C. On the Efficiency of ICO

ICO is the core innovation that makes the entire batch-processing strategy work. When $N_V = 15$, the average CPU time for each batch would be 479.256 seconds with the optimization method in [25].

D. Experimental Setups and Discussions

Real experiments, as shown in Fig. 6, were conducted with a swarm of three car-like robots in a 2 m×2 m indoor scenario. AprilTags were placed on the top of the robots and the surrounding street blocks for visual localization via a global camera facing the scenario. The perception and localization information was sent to a desktop PC, wherein the proposed method was executed to generate the cooperative trajectories before the robots began to move. The optimized trajectories were sent to the robots through the ZigBee communication technology for closed-loop tracking. Regarding the trajectory tracking in each robot, a PID controller is used for longitudinal tracking, a Pure Pursuit controller is used for lateral tracking, and the control frequency is set to 10 Hz [35]. Typical experimental results are presented in www.bilibili.com/video/BV1ct4y1e7ks?p=2.

IV. Conclusions

This paper has introduced a batch-processing framework for the lane-free AIM scheme, which aims to grasp the strengths of both planning-based and reservation-based methods. Simulation cases with long-lasting traffic flows indicate that the proposed method is efficient. Small-scale experiments validate the proposal preliminarily.

Our future works include 1) extending the framework so that it can handle mixed traffic flows with both cooperative and non-cooperative vehicles, 2) enlarging the scale of robots involved in the real-world tests, and 3) allowing the number of vehicles in each batch to change as per the density of the road.

References

[1] J. Guanetti, Y. Kim, and F. Borrelli, "Control of connected and automated vehicles: State of the art and future challenges," *Annual Reviews in Control*, vol. 45, pp. 18–40, 2018.

[2] B. Li, N. Jia, P. Li, and Y. Li, "Incrementally constrained dynamic optimization: A computational framework for lane change motion planning of connected and automated vehicles," *Journal of Intelligent Transportation Systems*, vol. 23, no. 6, pp. 557–568, 2019.

[3] D. Yoon, G. Ali and B. Ayalew, "Cooperative perception in connected vehicle traffic under field-of-view and participation variations," In *Proc. 2019 IEEE 2nd Connected and Automated Vehicles Symposium*, pp. 1–6, 2019.

[4] B. Li, Y. M. Zhang, Y. Feng, et al., "Balancing computation speed and quality: A decentralized motion planning method for cooperative lane changes of connected and automated vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 340–350, 2018.

[5] R. Jia, P. Jiang, L. Liu, et al., "Data-driven congestion trends prediction of urban transportation," *IEEE Internet of Things Journal*, vol. 5, no. 2, pp. 581–591, 2018.

[6] L. Chen, and C. Englund, "Cooperative intersection management: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2015.

[7] J. Rios-Torres, and A. A. Malikopoulos, "A survey on the coordination of connected and automated vehicles at intersections and merging at highway on-ramps," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 5, pp. 1066–1077, 2016.

[8] B. Li, Y. M. Zhang, N. Jia, et al., "Autonomous intersection management over continuous space: A microscopic and precise solution via computational optimal control," *IFAC-PapersOnLine*, 17312–17317, 2021.

[9] M. Khayatian, M. Mehrabian, E. Andert, et al., "A survey on intersection management of connected autonomous vehicles," *ACM Transactions on Cyber-Physical Systems*, vol. 4, no. 4, pp. 1–27, 2020.

[10] Y. Meng, L. Li, F. Wang, et al., "Analysis of cooperative driving strategies for nonsignalized intersections," *IEEE Transactions on Vehicular Technology*, vol. 67, no. 4, pp. 2900–2911, 2018.

[11] J. Ding, L. Li, H. Peng, and Y. Zhang, "A rule-based cooperative merging strategy for connected and automated vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol, 21, no. 8, pp. 3436–3446, 2019.

[12] H. Xu, Y. Zhang, L. Li, and W. Li, "Cooperative driving at unsignalized intersections using tree search," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 11, pp. 4563–4571, 2019.

[13] K. Dresner, and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research*, vol. 31, pp. 591–656, 2008.

[14] L. Li, and F. Wang, "Cooperative driving at blind crossings using intervehicle communication," *IEEE Transactions on Vehicular Technology*, vol. 55, no. 6, pp. 1712–1724, 2006.

[15] A. Mirheli, L. Hajibabai, and A. Hajbabaie, "Development of a signal-head-free intersection control logic in a fully connected and autonomous vehicle environment," *Transportation Research Part C: Emerging Technologies*, vol. 92, pp. 412–425, 2018.

[16] A. Mirheli, M. Tajalli, L. Hajibabai, and A. Hajbabaie, "A consensus-based distributed trajectory control in a signal-free intersection," *Transportation Research Part C: Emerging Technologies*, vol. 100, pp. 161–176, 2019.

[17] M. Levin, and D. Rey, "Conflict-point formulation of intersection control for autonomous vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 85, pp. 528–547, 2017.

[18] B. Xu, S. E. Li, Y. Bian, S. Li, X. J. Ban, J. Wang, and K. Li, "Distributed conflict-free cooperation for multiple connected vehicles at unsignalized intersections," *Transportation Research Part C: Emerging Technologies*, vol. 93, pp. 322–334, 2018.

[19] A. Malikopoulos, C. G. Cassandras, and Y. Zhang, "A decentralized energy-optimal control framework for connected automated vehicles at signal-free intersections," *Automatica*, vol. 93, pp. 244–256, 2018.

[20] I. H. Zohdy, R. K. Kamalanathsharma, and H. Rakha, "Intersection management for autonomous vehicles using iCACC," In *Proc. 15th International IEEE Conference on Intelligent Transportation Systems*, pp. 1109–1114, 2012.

[21] M. Kamal, J. Imura, A. Ohata, et al., "Coordination of automated vehicles at a traffic-lightless intersection," In *Proc. 16th International IEEE Conference on Intelligent Transportation Systems*, pp. 922–927, 2013.

[22] P. Lin, J. Liu, P. J. Jin, and B. Ran, "Autonomous vehicle-intersection coordination method in a connected vehicle environment," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 4, pp. 37–47, 2017.

[23] B. Liu, Q. Shi, Z. Song, and A. Kamel, "Trajectory planning for autonomous intersection management of connected vehicles," *Simulation Modelling Practice and Theory*, vol. 90, pp. 16–30, 2019.

[24] Z. Li, Q. Wu, H. Yu, et al., "Temporal-spatial dimension extension-based intersection control formulation for connected and autonomous vehicle systems," *Transportation Research Part C: Emerging Technologies*, vol. 104, pp. 234–248, 2019.

[25] B. Li, and Y. M. Zhang, "Fault-tolerant cooperative motion planning of connected and automated vehicles at a signal-free and lane-free intersection," *IFAC-Papers Online*, vol. 51, no. 24, pp. 60–67, 2018.

[26] B. Li, Y. M. Zhang, Y. Zhang, et al., "Near-optimal online motion planning of connected and automated vehicles at a signal-free and lane-free intersection," In *Proc. 2018 IEEE Intelligent Vehicles Symposium*, pp. 1432–1437, 2018.

[27] B. Li, Y. M. Zhang, Z. Shao, and N. Jia, "Simultaneous versus joint computing: A case study of multi-vehicle parking motion planning," *Journal of Computational Science*, vol. 20, pp. 30–40, 2017.

[28] B. Li, Y. M. Zhang, N. Jia, et al., "Paving green passage for emergency vehicle in heavy traffic: Real-time motion planning under the connected and automated vehicles environment," In *15th IEEE International Symposium on Safety, Security, and Rescue Robotics*, 153–158, 2017.

[29] B. Li, Y. M. Zhang, and Z. Shao, "Spatio-temporal decomposition: A knowledge-based initialization strategy for parallel parking motion optimization," *Knowledge-Based Systems*, vol. 107, pp. 179–196, 2016.

[30] B. Li, T. Acarman, X. Peng, et al., "Maneuver planning for automatic parking with safe travel corridors: A numerical optimal control approach", In *Proc. 2020 European Control Conference*, pp. 1993–1998, 2020.

[31] B. Li, and Z. Shao, "A unified motion planning method for parking an autonomous vehicle in the presence of irregularly placed obstacles," *Knowledge-Based Systems*, vol. 86, pp. 11–20, 2015.

[32] A. Wächter, and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, pp. 25–57, 2006.

[33] L. T. Biegler, *Nonlinear Programming: Concepts, Algorithms, and Applications to Chemical Processes*. vol. 10. Philadelphia, PA, USA: SIAM, 2010.

[34] R. Fourer, D. M. Gay and B. W. Kernighan. *AMPL: A Modeling Language for Mathematical Programming*. Brooks/Cole-Thomson Learning, Pacific Grove, 2003.

[35] B. Li, Y. Ouyang, Y. M. Zhang, et al., "Optimal cooperative maneuver planning for multiple nonholonomic robots in a tiny environment via adaptive-scaling constrained optimization," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 1511–1518, 2021.