RDA: An Accelerated Collision Free Motion Planner for Autonomous Navigation in Cluttered Environments

Ruihua Han, *Graduate Student Member, IEEE*, Shuai Wang, *Member, IEEE*, Shuaijun Wang, *Graduate Student Member, IEEE*, Zeqing Zhang, Qianru Zhang, Yonina C. Eldar, *Fellow, IEEE*, Qi Hao, *Member, IEEE*, and Jia Pan, *Senior Member, IEEE*

***Abstract*—Autonomous motion planning is challenging in multi-obstacle environments due to nonconvex collision avoidance constraints. Directly applying numerical solvers to these nonconvex formulations fails to exploit the constraint structures, resulting in excessive computation time. In this letter, we present an accelerated collision-free motion planner, namely regularized dual alternating direction method of multipliers (RDADMM or RDA for short), for the model predictive control (MPC) based motion planning problem. The proposed RDA addresses nonconvex motion planning via solving a smooth biconvex reformulation via duality and allows the collision avoidance constraints to be computed in parallel for each obstacle to reduce computation time significantly. We validate the performance of the RDA planner through path-tracking experiments with car-like robots in both simulation and real-world settings. Experimental results show that the proposed method generates smooth collision-free trajectories with less computation time compared with other benchmarks and performs robustly in cluttered environments.**

Manuscript received 29 September 2022; accepted 8 January 2023. Date of publication 3 February 2023; date of current version 13 February 2023. This letter was recommended for publication by Associate Editor C. I. Vasile and Editor L. Pallottino upon evaluation of the reviewers' comments. This work was supported in part by Hong Kong SAR Research Grants Council (RGC) General Research Fund (GRF) under Grants 11202119 and 11207818, in part by Innovation and Hong Kong SAR Technology Commission under the InnoHK initiative, in part by the Science and Technology Innovation Committee of Shenzhen City under Grant JCYJ20200109141622964, and in part by the National Natural Science Foundation of China under Grants 62261160654 and 62001203. *(Corresponding authors: Qi Hao; Jia Pan.)*

Ruihua Han is with the Department of Computer Science and Engineering, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China, and also with the Department of Computer Science, The University of Hong Kong, Hong Kong, China (e-mail: hanrh@connect.hku.hk).

Shuai Wang is with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Shenzhen, Guangdong 518055, China (e-mail: s.wang@siat.ac.cn).

Shuaijun Wang is with the Department of Computer Science and Engineering, Harbin Institute of Technology, Shenzhen, Guangdong 518055, China (e-mail: jeanswsj@gmail.com).

Zeqing Zhang, Qianru Zhang, and Jia Pan are with the Department of Computer Science, The University of Hong Kong, Hong Kong, China (e-mail: zzqing@connect.hku.hk; zqrhku@connect.hku.hk; jpan@cs.hku.hk).

Yonina C. Eldar is with the Weizmann Institute of Science, Rehovot 7632706, Israel (e-mail: yonina.eldar@weizmann.ac.il).

Qi Hao is with the Computer Science and Engineering, the Shenzhen Key Laboratory of Robotics and Computer Vision, and the Sifakis Research Institute for Trustworthy Autonomous Systems, Southern University of Science and Technology, Shenzhen, Guangdong 518055, China (e-mail: hao.q@sustech.edu.cn).

The source code is available at https://github.com/hanruihua/RDA_planner.

This letter has supplementary downloadable material available at https://doi.org/10.1109/LRA.2023.3242138, provided by the authors.

Digital Object Identifier 10.1109/LRA.2023.3242138

***Index Terms*—Collision avoidance, constrained motion planning, optimization and optimal control.**

I. Introduction

MOTION planning has drawn rising interest in many practical applications, such as self-driving cars and unmanned logistic vehicles [1], [2]. Its main idea is to compute a trajectory, defined as a sequence of control commands or states, from the current state to the desired state. Motion planners can be classified into four categories [3]: graph search based, sampling based, interpolation based, and optimization based. Graph search based techniques, e.g., state lattice and A star [4], [5], treat the configuration space as a graph of vertices and edges and search the minimum cost path. Sampling based techniques, e.g., the rapidly-exploring random tree (RRT) [6], probe the configuration space with a sampling scheme and have been widely used in real-world applications. Interpolation based techniques construct a new group of data among known reference points through interpolation models, resulting in a feasible, smooth trajectory in structured environments [7]. Compared with other methods, optimization based techniques are advantageous in generating optimized and robust trajectories in complex scenarios through minimizing cost functions under physical constraints, such as dynamics, kinematics, and collision avoidance. For instance, the model predictive control (MPC) approach adopts the vehicle dynamics model to predict future states and produce feasible trajectories dynamically within a set of constraints [8].

Despite its promising features, there are two challenges hindering the development of optimization based techniques in real-world applications. First, the motion planning optimization problem is usually nonconvex due to collision avoidance constraints. Numerous approaches have been proposed to resolve this issue, such as linearizing the constraints by Taylor expansion [9], [10], relaxing the constraints to formulate an unconstrained problem [11], and reformulating the constraints via strong duality [12], [13]. However, most of these approaches consider the point-mass object model, or involve large approximation errors, or fail to exploit the problem structure. The effectiveness of optimization methods under the non-point-mass model requires further investigation (i.e., accounting for the object shape). Second, the computation cost grows significantly with the number of obstacles. While several parallel algorithms [14] have been developed to split multi-agent planning problems into smaller subproblems for each agent, they cannot

be directly extended to the parallelization of obstacles. On the other hand, the primal dual alternating optimization (PDAO) in [15] is able to update different collision avoidance constraints in parallel. However, PDAO may diverge due to the nonlinear coupling among primal and dual variables in collision avoidance constraints. Additionally, PDAO ignores the different importance of upcoming and future states due to the same safety distance for all time slots. Hence, developing a parallel and adaptive computing framework to effectively reduce the computation cost becomes imperative.

This paper proposes an accelerated collision-free motion planner that realizes accurate, real-time, and robust navigation under real-world obstacle shapes, computation constraints, and environmental uncertainties. The main contributions are summarized as follows.

1) We formulate an optimization based navigation problem with the non-point-mass obstacle model. The nonconvex constraints are reformulated into bi-convex counterparts by joint linearization and strong duality. The safety distance in the collision avoidance constraints is automatically tuned via l_1 regularization.
2) We develop a parallel local planner, termed regularized dual alternating direction method of multipliers (RDA), to solve the bi-convex problem in parallel while bringing robustness to the algorithm convergence.
3) We implement the RDA algorithm in a high-fidelity Gazebo simulator and an Ackermann autonomous vehicle, respectively, and compare its performance with several benchmark schemes in terms of computation cost and success rate.

The remainder of this paper is organized as follows. Section II reviews the related work. Section III introduces the distance based collision avoidance constraint and optimization problem statement. Then, Section IV describes the proposed RDA collision-free motion planner. Section V describes the simulation and real-world experiments. Section VI presents the conclusion.

II. Related Work

Optimization based collision-free motion planners aim to minimize or maximize a utility function w.r.t. a set of constrained state and control variables. For instance, in [16], an MPC scheme is proposed to accomplish the path tracking task by using the varied length time in the prediction horizon, where the collision avoidance constraint is modeled by the environmental envelope [17]. In fact, the time elastic bands (TEB) approach, which is the most popular method in robotics navigation, is also optimization based, which realizes motion planning by solving a multi-objective optimization problem [18]. To guarantee collision avoidance, a formulation of the constraint based on the minimum distance between robots and obstacles has been proposed, which, however, is nonconvex [19], [20]. Duality based approaches have been studied recently to handle nonconvexity, where the distance functions are reformulated to be smooth, differentiable counterparts via strong duality. The "least-intrusive" trajectory method is proposed in [12] for non-point-mass objects. However, its final formulation involves bi-linear terms and needs to be solved via nonconvex numerical solvers. The work [13] linearizes the bi-linear terms through duality and develops a sequential convex programming method. Nevertheless, only point-mass objects are considered in [13], which is inappropriate for practical polygonal robots such as self-driving cars. Furthermore, the complexity of sequential convex programming is exceedingly high when the number of obstacles is large.

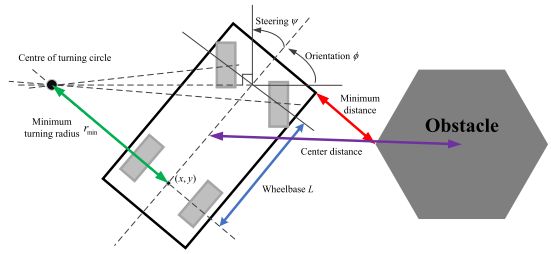


Fig. 1. Illustration of the Ackerman kinetic model and the distance between the robot and the obstacle.

There are many ways to reduce the computation cost for solving the large-scale motion planning problem. Specifically, we could consider only the adjacent agents and obstacles within a pre-defined range of interest such that the number of agents and obstacles is significantly reduced. However, the range of interest increases with the velocity of the vehicle, and collisions may occur if some out-of-range obstacles suddenly appear or accelerate. Distributed optimization has been leveraged to solve the problem in parallel, e.g., collision avoidance in multi-robot systems [15]. However, the algorithm in [15] is not compatible with the real-time robot operation system (ROS). Thus, the performance of parallel computing under practical sensor and actuator uncertainties requires further investigation.

As a promising distributed optimization framework, the alternating direction method of multipliers (ADMM) [21] has been adopted in multi-agent systems [9], [10], [14], which decompose a large centralized problem into several subproblems solved for each agent. However, for the cluttered environment with multiple obstacles, decomposing the problem w.r.t each obstacle via ADMM has not yet been investigated. The method adopted in multi-agent systems cannot be applied directly to multi-obstacle systems, since different agents have their own motion dynamics. The collision avoidance constraints are nonlinearly coupled with each other by a single motion dynamic constraint. Here, we tackle this coupling issue via joint linearization and duality and then leverage regularization and ADMM to enhance the robustness of the algorithm.

III. Problem Statement

A. Non-Point-Mass Collision Avoidance Constraint

The point-mass object is not a good model for some specific robots. For instance, for a robot with Ackermann kinematic model, as shown in Fig. 1. It is vital to take the rectangle shape into account when navigating in the corridor. Thus, a non-point-mass (i.e. arbitrary convex shape) representation is necessary.

1) Obstacle Model: The environment consists of M non-point-mass obstacles. The mth obstacle ($m = 1, \ldots, M$) is modeled as a compact convex set $\mathbb{O}_m$ that is represented by the conic inequality [22]:

$$\mathbb{O}_m = \{\mathbf{o} \in \mathbb{R}^{o_m} | \mathbf{D}_m \mathbf{o} \preceq_{\mathcal{O}_m} \mathbf{b}_m\}, \tag{1}$$

where $\mathbf{D}_m \in \mathbb{R}^{l_m \times o_m}$, $\mathbf{b}_m \in \mathbb{R}^{l_m}$, and $\mathcal{O}_m$ is a proper cone. The partial ordering w.r.t. a cone $\mathcal{K}$ is defined by $\mathbf{x} \preceq_{\mathcal{K}} \mathbf{y} \Leftrightarrow \mathbf{y} - \mathbf{x} \in \mathcal{K}$. Each geometric object can be represented with a proper cone through this formula. For example, if $\mathcal{O}_m = \mathbb{R}_+$, the obstacle is a polyhedron; if $\mathcal{O}_m = \{(\mathbf{x}, t) \in \mathbb{R}^{l_m+1} | \|\mathbf{x}\|_2 \leq t\}$, the obstacle is an ellipsoid. For static obstacles, the set $\mathbb{O}$ is a constant; for dynamic obstacles, the set $\mathbb{O}_m$ can be denoted as $\mathbb{O}_m^t$, which changes over time.

2) Robot Model: The state of each robot at time t is represented by its center point and denoted as $\mathbf{s}_t \in \mathbb{R}^{n_r}$. The value of n_r depends on the working space dimension. When the robot moves in a plane, we have $n_r = 3$ and $\mathbf{s}_t = (x_t, y_t, \theta_t)$, where (x_t, y_t) represents the position and θ_t the orientation. Given a certain state, the robot can be modeled as a compact set $\mathbb{Z}_t$ changing with state variable $\mathbf{s}_t$:

$$\mathbb{Z}_t(\mathbf{s}_t) = \mathbf{R}(\mathbf{s}_t)\mathbf{z} + \mathbf{p}(\mathbf{s}_t),\ \forall \mathbf{z} \in \mathbb{C},$$
$$\mathbb{C} = \{\mathbf{z} \in R^{n_r} | \mathbf{G}\mathbf{z} \preceq_{\mathcal{K}_r} \mathbf{h}\}, \tag{2}$$

where $\mathbf{R}(\mathbf{s}_t) \in \mathbb{R}^{n_r \times n_r}$ is the rotation matrix representing the orientation of the robot and $\mathbf{p}(\mathbf{s}_t) \in \mathbb{R}^{n_r}$ is the transition matrix representing the position of the robot. The vector $\mathbf{z} \in \mathbb{R}^{n_r}$ is the initial pose vector, and $\mathbb{C}$ is a convex set representing the robot shape at the initial position, with $\mathbf{G} \in R^{h \times n_r}$ and $\mathbf{h} \in \mathbb{R}^h$. We assume $\mathbf{R}(\mathbf{s}_t)$ and $\mathbf{p}(\mathbf{s}_t)$ are linear functions of $\mathbf{s}$. Any nonlinear function is represented by a linear approximation. For instance, the rotation matrix can be linearized by its Taylor expansion [23].

3) Collision Avoidance: The minimum distance between a robot and an obstacle $\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O})$ satisfies:

$$\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O}) = \min \{ \|\mathbf{e}\|_2 | (\mathbb{Z}_t(\mathbf{s}_t) + \mathbf{e}) \cap \mathbb{O} \neq \emptyset \}, \tag{3}$$

where $\emptyset$ denotes the empty set. Substituting the expressions of $\mathbb{Z}_t(\mathbf{s}_t)$ and $\mathbb{O}$ into (3), we can find $\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O})$ by solving the following convex problem:

$$\min_{\mathbf{z},\mathbf{o}} \|\mathbf{R}(\mathbf{s}_t)\mathbf{z} + \mathbf{p}(\mathbf{s}_t) - \mathbf{o}\|_2$$
$$\text{s.t. } \mathbf{D}\mathbf{o} \preceq_{\mathcal{K}_o} \mathbf{b},\ \mathbf{G}\mathbf{z} \preceq_{\mathcal{K}_r} \mathbf{h} \tag{4}$$

with variable $\mathbf{z}$ and $\mathbf{o}$. Thus, to guarantee the collision avoidance, we have the following constraint:

$$\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O}) \geq d_{\text{safe}}, \tag{5}$$

where d_{safe} is a positive real number representing the safe distance. Constraint (5) is nonconvex as the convex distance function is at the left hand side of operator $\geq$.

B. State Evolution

The robot state evolution can be described as follows:

$$\mathbf{s}_{t+1} = \mathbf{s}_t + f(\mathbf{s}_t, \mathbf{u}_t)\Delta t, \tag{6}$$

where the initial state $\mathbf{s}_0$ is the current state, $\mathbf{u}_t \in \mathbb{R}^{n_u}$ is the control vector, and Δt is the time length of each slot. When the robot moves in a plane, we have $n_u = 2$ and $\mathbf{u}_t = (v_t, \psi_t)$, where (v_t, ψ_t) represents the linear and angular velocities.[1]

[1]The linear and angular velocities (v_t, ψ_t) may also be added to the state space model, where we have $n_r = 5$ and accelerations become the control vector. Note that this would lead to an equivalent optimization problem, and the proposed RDA is still applicable.

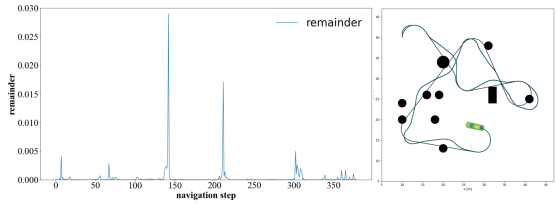


Fig. 2. Accuracy of the adopted linear approximation.

For linear robot dynamics (e.g., omni wheels), we have $f(\mathbf{s}_t, \mathbf{u}_t) = [(\mathbf{A}_t - \mathbf{I})\mathbf{s}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{c}_t)/\Delta t]$, and

$$\mathbf{s}_{t+1} = \mathbf{A}_t\mathbf{s}_t + \mathbf{B}_t\mathbf{u}_t + \mathbf{c}_t,\ t = 0, \ldots, N-1, \tag{7}$$

where $(\mathbf{A}_t, \mathbf{B}_t, \mathbf{c}_t)$ are the coefficient matrices. For nonlinear robot dynamics (e.g., Ackerman wheels [24]), we have

$$f(\mathbf{s}_t, \mathbf{u}_t) = \left[v_t \cos(\theta_t), v_t \sin(\theta_t), \frac{v_t \tan\psi_t}{L} \right]^T.$$

By leveraging the first-order Taylor polynomial, the nonlinear dynamics function can be linearized into the same structure as (7), where the associated coefficients $(\mathbf{A}_t, \mathbf{B}_t, \mathbf{c}_t)$ at time t in (7) are given by:

$$\mathbf{A}_t = \begin{bmatrix} 1 & 0 & -\bar{v}_t \sin(\bar{\theta}_t)\Delta t \\ 0 & 1 & \bar{v}_t \cos(\bar{\theta}_t)\Delta t \\ 0 & 0 & 1 \end{bmatrix}, \tag{8}$$

$$\mathbf{B}_t = \begin{bmatrix} \cos(\bar{\theta}_t)\Delta t & 0 \\ \sin(\bar{\theta}_t)\Delta t & 0 \\ \frac{\tan\bar{\psi}_t \Delta t}{L} & \frac{\bar{v}_t \Delta t}{L\cos^2\bar{\psi}_t} \end{bmatrix}, \tag{9}$$

$$\mathbf{c}_t = \begin{bmatrix} \bar{\theta}_t\bar{v}_t \sin(\bar{\theta}_t)\Delta t \\ -\bar{\theta}_t\bar{v}_t \cos(\bar{\theta}_t)\Delta t \\ -\frac{\bar{\psi}_t\bar{v}_t\Delta t}{L\cos^2\psi_t} \end{bmatrix}, \tag{10}$$

where $\bar{v}_t$ and $\bar{\psi}_t$ are the nominal linear speed and steering angle of the control vector $\bar{\mathbf{u}}_t$ at time t, respectively. The constant L denotes the distance between the front and rear axles. To evaluate the accuracy of the adopted linerization, the trajectories generated by the original dynamics function (green line) and linearized dynamics function (blue line) are shown in Fig. 2. It can be seen that the two trajectories are very close to each other. Moreover, as seen in Fig. 2, the approximation error fluctuates between 0 and 0.03, which is negligible. This is because we can set $\bar{\theta}_t, \bar{v}_t, \bar{\psi}_t$ as the robot states from the last navigation step, and the robot states have a slight change between consecutive steps.

In practice, there are boundary constraints on the control vector $\mathbf{u}_t$:

$$\mathbf{u}_{\min} \preceq \mathbf{u}_t \preceq \mathbf{u}_{\max},\ \mathbf{a}_{\min} \preceq \mathbf{u}_{t+1} - \mathbf{u}_t \preceq \mathbf{a}_{\max}, \tag{11}$$

where $\mathbf{u}_{\min}, \mathbf{u}_{\max} \in \mathbb{R}^{n_u}$ are the minimum and maximum values of the control vector, respectively. The minimum and maximum values of the acceleration vector are denoted by $\mathbf{a}_{\max}$ and $\mathbf{a}_{\min}$, respectively.

C. Problem Formulation

We consider the MPC formulation for autonomous planning, where the future state is predicted by the robot dynamics within a receding horizon consisting of N time steps. The goal

Algorithm 1: RDA Motion Planner.

Initialize the given points of the robot state $\mathbf{s}$ and control vector $\mathbf{u}$;
for *iteration* $k = 1, 2, \cdots$ **do**
  Update the variables $\mathbf{s}$, $\mathbf{u}$, $\mathbf{d}$ by solving (20a) with CVXPY;
  Update the dual collision variables $\boldsymbol{\lambda}$, $\boldsymbol{\mu}$, $\mathbf{z}$ by solving (20b) with accelerated gradient projection in a parallel manner;
  Update the Lagrangian multipliers $\boldsymbol{\zeta}$ and $\boldsymbol{\xi}$ by (20c)-(20d) in a parallel manner;
  if (21) *and* (22) *are satisfied* **then**
    break
  end
end
Apply the first receding step control vector to the robot

of MPC is to minimize a convex and smooth cost function $C_0(\{\mathbf{s}_t, \mathbf{u}_t\}_{\forall t})$ under collision avoidance, state evolution, and boundary constraints. The explicit form of C_0 depends on the type of navigation task. For instance, the cost function of path tracking is given by

$$C_0(\{\mathbf{s}_t, \mathbf{u}_t\}) = \sum_{t=0}^{N} \left[Q_t(\mathbf{s}_t - \mathbf{s}_t^{\diamond})^2 + P_t(v_t - v_t^{\diamond})^2 \right], \quad (12)$$

where $\mathbf{s}_t^{\diamond}$ and $v_t^{\diamond}$ are the reference state and speed. A larger value of the weighting coefficients $\{Q_t, P_t\}$ leads to the robot moving with the reference path.

The MPC optimization problem over a finite future horizon $(t = 1, \cdots N)$ with M obstacles is obtained by combining the constraints in (5), (7), (11):

$$\mathcal{P}_0 : \min_{\{\mathbf{s}_t, \mathbf{u}_t\}} \; C_0(\{\mathbf{s}_t, \mathbf{u}_t\}) \quad (13a)$$

$$\text{s.t.} \;\; \mathbf{s}_{t+1} = \mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{c}_t, \; \forall t, \quad (13b)$$

$$\mathbf{u}_{\min} \preceq \mathbf{u}_t \preceq \mathbf{u}_{\max}, \; \forall t, \quad (13c)$$

$$\mathbf{a}_{\min} \preceq \mathbf{u}_{t+1} - \mathbf{u}_t \preceq \mathbf{a}_{\max}, \; \forall t, \quad (13d)$$

$$\mathbf{dist}\left(\mathbb{Z}_t\left(\mathbf{s}_t\right), \mathbb{O}_m\right) \geq d_{\text{safe}}, \; \forall t, m. \quad (13e)$$

The challenges of solving $\mathcal{P}_0$ in cluttered environments are twofold:

1) (13e) is a sufficient but not necessary condition for collision avoidance, making it difficult to be satisfied;
2) the number of obstacles M in (13e) could be large.

Below, an effective method will be presented to tackle these challenges.

IV. RDA Collision-Free Motion Planner

In this section, the RDA approach for solving $\mathcal{P}_0$ is presented. The overall algorithm is summarized in Algorithm 1.

A. l_1-Regularization

To address the first challenge, we need to reformulate the constraint (13e). Intuitively, an ideal strategy is to generate a larger d_{safe} for the upcoming time steps and a smaller d_{safe} for the future time steps. To this end, this paper proposes a dynamic safety distance approach, where d_{safe} is replaced by a vector variable $\mathbf{d} = [d_1, \ldots, d_N]^T \in \mathbb{R}^N$, and equation (13e) is converted into $\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O}_m) \geq d_t$. Each d_t is upper bounded by $d_{\max}$ and lower bounded by $d_{\min}$. However, such a formulation ignores the different importance of upcoming and future states, which leads to a solution with $d_t = d_{\min}$ for all t. Consequently, the l_1-regularization method is further proposed to generate uneven values of $\{d_t\}$ among different time steps by imposing sparsity upon $\mathbf{d}$. This would automatically allocate different attention to different states. Mathematically, l_1-regularization is realized by adding a penalty function of $\mathbf{d}$ to C_0. The penalty function is set as the negative l_1-norm of $\mathbf{d}$, i.e., $C_1(\mathbf{d}) = -\eta\|\mathbf{d}\|_1 = -\eta \sum_{t=0}^{N} d_t$, where $\eta \in \mathbb{R} \geq 0$ is a weight factor to adjust the collision avoidance ability in different scenarios:

$$\min_{\{\mathbf{s}_t, \mathbf{u}_t\}, \mathbf{d}} \; C_0(\{\mathbf{s}_t, \mathbf{u}_t\}) + C_1(\mathbf{d}) \quad (14a)$$

$$\text{s.t.} \;\; (13b) - (13d), \quad (14b)$$

$$d_t \in [d_{\min}, d_{\max}], \; \forall t, \quad (14c)$$

$$\mathbf{dist}\left(\mathbb{Z}_t\left(\mathbf{s}_t\right), \mathbb{O}_m\right) \geq d_t, \; \forall t, m. \quad (14d)$$

Note that a larger value of η encourages d_t to reach $d_{\max}$ and vice versa.

B. Parallel Computation Via ADMM

To address the second challenge, we first transform the constraint $\mathbf{dist}(\mathbb{Z}_t(\mathbf{s}_t), \mathbb{O}_m) \geq d_t$ into its linearized dual form:

$$\begin{aligned}
&\mathbf{dist}\left(\mathbb{Z}_t\left(\mathbf{s}_t\right), \mathbb{O}_m\right) \geq d_t \Leftrightarrow \\
&\boldsymbol{\lambda}_{t,m} \succeq_{\mathcal{O}_m^*} \mathbf{0}, \; \boldsymbol{\mu}_{t,m} \succeq_{\mathcal{K}_r^*} \mathbf{0}, \\
&\boldsymbol{\lambda}_{t,m}^T \mathbf{D}_m \mathbf{p}_t(\mathbf{s}_t) - \boldsymbol{\lambda}_{t,m}^T \mathbf{b}_m - \boldsymbol{\mu}_{t,m}^T \mathbf{h} \geq d_t, \\
&\boldsymbol{\mu}_{t,m}^T \mathbf{G} + \boldsymbol{\lambda}_{t,m}^T \mathbf{D}_m \mathbf{R}_t(\mathbf{s}_t) = 0, \\
&\left\|\mathbf{D}_m^T \boldsymbol{\lambda}_{t,m}\right\|_* \leq 1,
\end{aligned} \quad (15)$$

where $\boldsymbol{\lambda}_{t,m} \in \mathbb{R}^{l_m}$ and $\boldsymbol{\mu}_{t,m} \in \mathbb{R}^h$ are the dual variables. The symbol $\mathcal{K}^*$ is the dual cone of $\mathcal{K}$ and $\|\cdot\|_*$ is the dual norm. The dual norm of the Euclidean norm is equal to itself. Note that the above transformation is different from that in [12]. Specifically, the method in [12] does not involve linearization of translation and rotation matrices, resulting in a smooth nonconvex dual problem. In contrast, by leveraging joint linearization and duality transformation, (15) has a smooth bi-convex dual form, which facilitates the subsequent parallelization procedure.

Plugging (15) into (14), the problem (14) is converted into be a bi-convex problem:

$$\min_{\substack{\{\mathbf{s}_t, \mathbf{u}_t, d_t\} \\ \{\boldsymbol{\lambda}_{t,m}, \boldsymbol{\mu}_{t,m}, z_{t,m}\}}} \; C_0(\{\mathbf{s}_t, \mathbf{u}_t\}) + C_1(\mathbf{d}) \quad (16a)$$

$$\text{s.t.} \;\; \mathbf{s}_{t+1} = \mathbf{A}_t \mathbf{s}_t + \mathbf{B}_t \mathbf{u}_t + \mathbf{c}_t, \; \forall t, \quad (16b)$$

$$\mathbf{u}_{\min} \preceq \mathbf{u}_t \preceq \mathbf{u}_{\max}, \; \forall t, \quad (16c)$$

$$\mathbf{a}_{\min} \preceq \mathbf{a}_t \preceq \mathbf{a}_{\max}, \; \forall t, \quad (16d)$$

$$d_t \in [d_{\min}, d_{\max}], \; \forall t, \quad (16e)$$

$$z_{t,m} \geq 0, \; \forall t, m, \quad (16f)$$

$$\boldsymbol{\lambda}_{t,m} \succeq_{\mathcal{O}_m^*} \mathbf{0}, \; \boldsymbol{\mu}_{t,m} \succeq_{\mathcal{K}_r^*} \mathbf{0}, \; \forall t, m, \quad (16g)$$

$$\left\|\mathbf{D}_m^T\boldsymbol{\lambda}_{t,m}\right\|_* \le 1,\ \forall t,m, \tag{16h}$$

$$H_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m})=0,\ \forall t,m, \tag{16i}$$

$$I_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},d_t,z_{t,m})=0,\ \forall t,m, \tag{16j}$$

where, the nonlinear functions in (16i) and (16j) are

$$H_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m})=\boldsymbol{\mu}_{t,m}^T\mathbf{G}+\boldsymbol{\lambda}_{t,m}^T\mathbf{D}_m\mathbf{R}_t(\mathbf{s}_t), \tag{17}$$

$$I_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},d_t,z_{t,m})=\boldsymbol{\lambda}_{t,m}^T\mathbf{D}_m\mathbf{p}_t(\mathbf{s}_t)-\boldsymbol{\lambda}_{t,m}^T\mathbf{b}_m-\boldsymbol{\mu}_{t,m}^T\mathbf{h}-d_t-z_{t,m}. \tag{18}$$

Existing methods approximate these bi-linear terms by Taylor polynomial [13] and solve a sequence of convex problems using the interior point method (IPM). The associated computation complexity is $\mathcal{O}((N(n_r+n_u+1)+N(\sum_{m=1}^{M}l_m+Mh))^{3.5})$, which is exceedingly high for large M.

In contrast, we solve this bi-convex optimization problem by ADMM, where each iteration includes smaller convex subproblems, and the dual variables associated with different obstacles are updated in parallel to accelerate the computation speed. In particular, the augmented Lagrangian (scaled form) of (16) can be formulated as

$$\begin{aligned}&\mathcal{L}\left(\{\mathbf{s}_t,\mathbf{u}_t,d_t\},\{\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m}\},\{\xi_{t,m},\zeta_{t,m}\}\right)\\&=C_0(\{\mathbf{s}_t,\mathbf{u}_t\})+C_1(\mathbf{d})+J(\{\mathbf{s}_t,\mathbf{u}_t,d_t\})\\&\quad+\sum_{t=0}^{N}\sum_{m=1}^{M}Q_{t,m}(\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m})\\&\quad+\frac{\rho}{2}\sum_{t=0}^{N}\sum_{m=0}^{M}\|I_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},d_t,z_{t,m})+\zeta_{t,m}\|_2^2\\&\quad+\frac{\rho}{2}\sum_{t=0}^{N}\sum_{m=0}^{M}\|H_{t,m}(\mathbf{s}_t,\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m})+\xi_{t,m}\|_2^2,\end{aligned} \tag{19}$$

where $\{\xi_{t,m},\zeta_{t,m}\}$ are dual variables corresponding to the equality constraints (16i) and (16j), ρ is the penalty parameter chosen as a large value. Here $J(\{\mathbf{s}_t,\mathbf{u}_t,d_t\})$ is the indicator function of the constraints (16b)–(16e), i.e., $J(\{\mathbf{s}_t,\mathbf{u}_t,d_t\})=0$ if (16b)–(16e) are satisfied and $J(\{\mathbf{s}_t,\mathbf{u}_t,d_t\})=\infty$ otherwise. Similarly, $Q_{t,m}(\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m})$ is the indicator function of the (t,m)th constraints in (16f)–(16h).

In (19), the first three items are coupled across different t due to the constraint (16b), while the latter three terms are decomposable w.r.t. t and m. Consequently, we split the primal variables into two groups: 1) $\{\mathbf{s}_t,\mathbf{u}_t,d_t\}$; 2) $\{\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m}\}$. Given the fact that the dual variables $\{\xi_{t,m},\zeta_{t,m}\}$ can be updated in parallel, the ADMM method for minimizing the augmented Lagrangian is

$$\begin{aligned}\{\mathbf{s}_t^{k+1},\mathbf{u}_t^{k+1},d_t^{k+1}\}=&\underset{\{\mathbf{s}_t,\mathbf{u}_t,d_t\}}{\arg\min}\,\mathcal{L}\left(\{\mathbf{s}_t,\mathbf{u}_t,d_t\},\right.\\&\left.\{\boldsymbol{\lambda}_{t,m}^k,\boldsymbol{\mu}_{t,m}^k,z_{t,m}^k\},\{\xi_{t,m}^k,\zeta_{t,m}^k\}\right),\end{aligned} \tag{20a}$$

$$\begin{aligned}\boldsymbol{\lambda}_{t,m}^{k+1},\boldsymbol{\mu}_{t,m}^{k+1},z_{t,m}^{k+1}=&\underset{\boldsymbol{\lambda}_{t,m}\boldsymbol{\mu}_{t,m},z_{t,m}}{\arg\min}\quad\mathcal{L}\left(\{\mathbf{s}_t^k,\mathbf{u}_t^k,d_t^k\},\right.\\&\left.\{\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m}\},\{\xi_{t,m}^k,\zeta_{t,m}^k\}\right),\forall t,m,\end{aligned} \tag{20b}$$

$$\xi_{t,m}^{k+1}=\xi_{t,m}^k+(\boldsymbol{\mu}_{t,m}^k)^T\mathbf{G}+(\boldsymbol{\lambda}_{t,m}^k)^T\mathbf{D}_m\mathbf{R}_t(\mathbf{s}_t^k), \tag{20c}$$

$$\zeta_{t,m}^{k+1}=\zeta_{t,m}^k+(\boldsymbol{\lambda}_{t,m}^k)^T\mathbf{D}_m\mathbf{p}_t(\mathbf{s}_t^k)-(\boldsymbol{\lambda}_{t,m}^k)^T\mathbf{b}_m-(\boldsymbol{\mu}_{t,m}^k)^T\mathbf{h}-d_t^k-z_{t,m}^k,\forall t,m, \tag{20d}$$

Problems (20a) and (20b) are convex, and (20c) and (20d) are the subgradients to update the dual variables. Problem (20a) can be efficiently solved via a convex solver, e.g. CVXPY [25], whose computation cost is independent of M. Problems (20b)–(20d) are all separable across (t,m), and $\{\boldsymbol{\lambda}_{t,m},\boldsymbol{\mu}_{t,m},z_{t,m},\xi_{t,m},\zeta_{t,m}\}$ can all be updated in parallel. Furthermore, each problem in (20b) is a conic constrained least squares problem which can be solved with low-cost solvers. The stopping criteria to terminate the iterative procedure is given by

$$\begin{aligned}&\sum_t\sum_m\left[\|H_{t,m}(\mathbf{s}_t^k,\boldsymbol{\lambda}_{t,m}^k,\boldsymbol{\mu}_{t,m}^k)\|_2^2\right.\\&\left.+\|I_{t,m}(\mathbf{s}_t^k,\boldsymbol{\lambda}_{t,m}^k,\boldsymbol{\mu}_{t,m}^k,d_t^k,z_{t,m}^k)\|_2^2\right]\le\epsilon^{\text{pri}},\end{aligned} \tag{21}$$

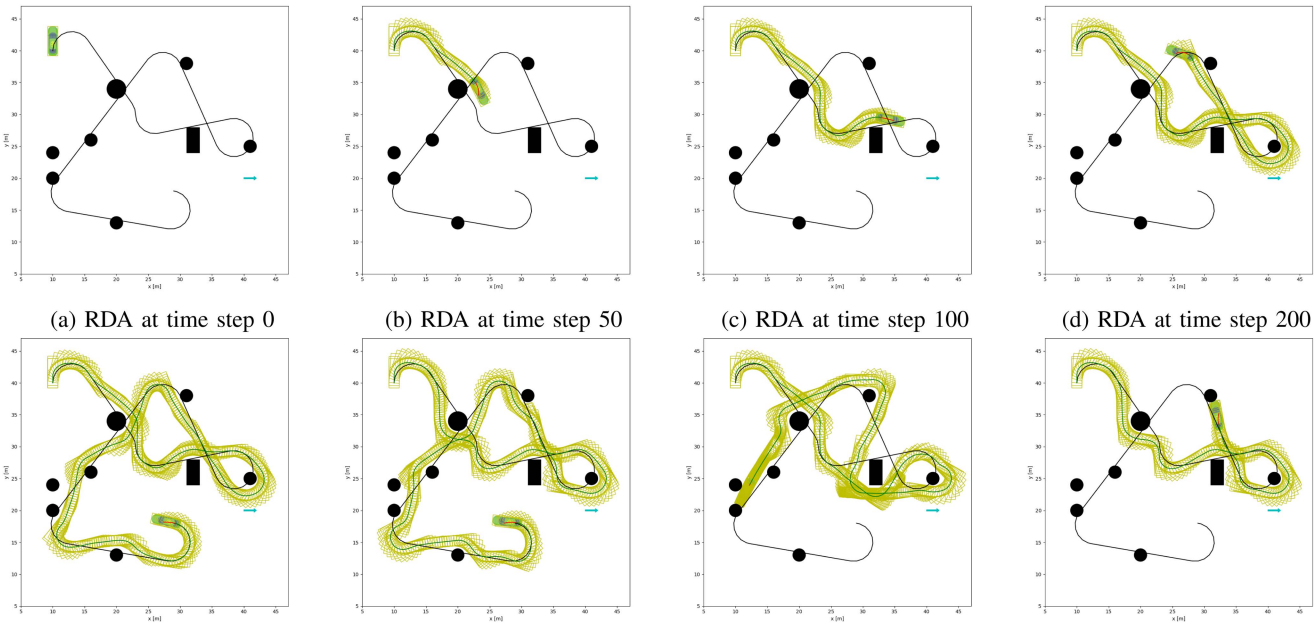
$$\sum_t\sum_m\left[\|\boldsymbol{\lambda}_{t,m}^{k+1}-\boldsymbol{\lambda}_{t,m}^k\|_2^2+\|\boldsymbol{\mu}_{t,m}^{k+1}-\boldsymbol{\mu}_{t,m}^k\|_2^2\right]\le\epsilon^{\text{dual}}, \tag{22}$$

where the first condition guarantees the primal residual being small and the second condition guarantees the dual residual being small. Constants $\epsilon^{\text{pri}}>0$ and $\epsilon^{\text{dual}}>0$ are the stopping criterion values. To achieve the best tradeoff between the solution quality and computational cost, ϵ^{pri} and ϵ^{dual} can be chosen empirically. According to our experimental data [26], the stopping criteria value can be set to $\epsilon^{\text{pri}},\epsilon^{\text{dual}}\in[0.1,1]$.

C. Complexity Analysis and Warm Starting

1) Complexity Analysis: The entire procedure of RDA is summarized in Algorithm 1. In each iteration, the CVXPY solver is first adopted to solve (20a) with a complexity of $\mathcal{O}((N(n_r+n_u+1))^{3.5})$. Then, to optimize dual collision variables, we need to solve (20b), which involves NM subproblems with the mth problem being solved at a computational cost of $\mathcal{O}((l_m+h+1)^{3.5})$. Finally, the Lagrange multipliers are updated via vector-matrix multiplications in (20c)–(20d), which involve a complexity of $\mathcal{O}(N(\sum_{m-1}^{M}l_mn_m+Mhn_r))$. Therefore, the total complexity of RDA is given by $\mathcal{O}(K(N(n_r+n_u+1))^{3.5}+N\sum_{m=1}^{M}(l_m+h+1)^{3.5}+N(\sum_{m-1}^{M}l_mn_m+Mhn_r))$, where K is the number iterations for the RDA to converge.

2) Warm Starting: It can be seen from the above analysis that the complexity of RDA is linear in M. This significantly saves computation time compared with IPM whose complexity is cubic in M. Furthermore, the value of K can be significantly reduced by leveraging warm starting. Specifically, for collision-avoidance autonomous driving, the odometry of the ego-vehicle and the states of surrounding obstacles do not vary significantly between two consecutive time steps, and therefore, the current RDA solutions can be adopted as an initial guess of both the primal and dual solutions to the subsequent RDA iterative procedure. That is, setting $\{\boldsymbol{\lambda}_{t,m}^0,\boldsymbol{\mu}_{t,m}^0,z_{t,m}^0,\xi_{t,m}^0,\zeta_{t,m}^0\}=$



(a) RDA at time step 0 (b) RDA at time step 50 (c) RDA at time step 100 (d) RDA at time step 200

(e) RDA trajectory (f) OBCA trajectory (g) RDA-II trajectory (h) OBCA-II trajectory

Fig. 3. Navigation process and trajectory comparison of RDA and OBCA in a simulated environment.

$\{\boldsymbol{\lambda}^{\diamond}_{t,m}, \boldsymbol{\mu}^{\diamond}_{t,m}, z^{\diamond}_{t,m}, \xi^{\diamond}_{t,m}, \zeta^{\diamond}_{t,m}\}$ for (20a), where the solutions with $\diamond$ denote the cached values obtained from the previous RDA outputs.

V. Experiments and Results

In this section, we adopt numerical simulations, high-fidelity simulators, and hardware experiments to verify the performance and efficiency of RDA. In particular, we consider the collision-free path tracking task for autonomous vehicles. The task is defined as moving as close as possible to a reference path while avoiding collisions. In our experiments, we set $\eta = 10$, $\rho = 10$.

Besides the proposed RDA planner, we also simulate the following schemes for comparison: (1) **Optimization based collision avoidance (OBCA) planner** [12], which directly solves the dual-MPC problem via nonconvex optimization methods such as sequential convex programming; (2) **Time elastic band (TEB) planner** [18], which obtains a sequence of robot poses via general graph optimization and sparsity regularization; (3) **Point-mass ADMM (PMA)** [14], which models each obstacle as a single point and determines the collision condition by computing the distance between this point and the ego-robot position; (4) **OBCA-II**, which adopts the OBCA planner but with no safe-distance regularization; (5) **RDA-II**, which adopts the RDA planner but with no safe-distance regularization.

A. Numerical Simulation

We verify the proposed RDA in our intelligent robot simulator (*ir-sim*)[2], which is a Python-based 2D numerical simulator for robotic localization and navigation. The simulation scenario is given in Fig. 3, where the obstacles are marked in black with various shapes (e.g., rectangle, circle, etc.). The reference path (i.e., the black line) in *ir-sim* is generated from a series of directional waypoints and calculated by Dubins path [27].

[2][Online]. Available: https://github.com/hanruihua/ir_sim.

The resulting navigation process of RDA in a cluttered environment with 8 obstacles is shown in Fig. 3(a)–(e). The actual robot path is represented by a green line and the robot is marked as a yellow box. The predicted trajectory within the receding horizon is represented by a red curve. It can be seen that the vehicle successfully avoids all obstacles while moving close to the reference path in a smooth manner.

We compare the proposed RDA with OBCA, whose trajectories and computation times are shown in Fig. 3(e), (f) and Fig. 5(a), (b), respectively. It can be seen that both approaches have the ability to find a proper collision-free trajectory. However, the required computation time of RDA is significantly shorter than that of OBCA, i.e., 0.1 seconds for RDA versus over 0.3 seconds for OBCA, as illustrated in Fig. 5(a). This implies that the planning frequency of RDA is three times as fast as that of OBCA, thus providing real-time potential in high-speed scenarios. Furthermore, as shown in Fig. 5(b), the computation time of RDA only has a slight change as the number of obstacles increases, while that of OCBA grows dramatically. This corroborates the parallel computation capability of RDA.

Here, we compare RDA with RDA-II and OBCA-II with the safe distance d_{safe} being fixed to $0.5\,\text{m}$. It can be seen from Fig. 3(g), (h) that the vehicle gets stuck at position (10, 20) for RDA-II due to the dense obstacles residing within this local region and position (31,32) for OBCA-II due to the 180-degree turn at the right hand side of the reference path. In contrast, the proposed RDA with distance regularization $\eta \in [5, 30]$ passes these challenging points.

Finally, we compare RDA with PMA in another challenging scenario, i.e., the corridor scenario as shown in Fig. 4. It can be seen that the vehicle with RDA (yellow box) successfully crosses the entire tunnel while that with PMA (red circle) gets stuck at the first narrow gap. This is because PMA computes the distance between the centers of the ego-vehicle and the nearby obstacle, which works well only when both objects are circular.

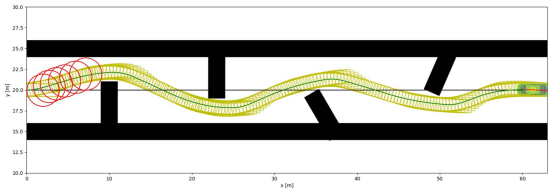
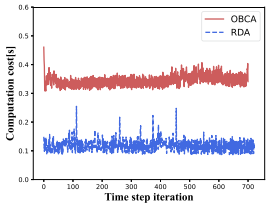
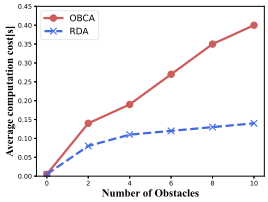


Fig. 4. RDA (yellow box) and PMA (red circle) trajectories in the corridor.





(a) Computation cost in one trial (b) Average computation cost

Fig. 5. Comparison of computation cost of RDA and OBCA.

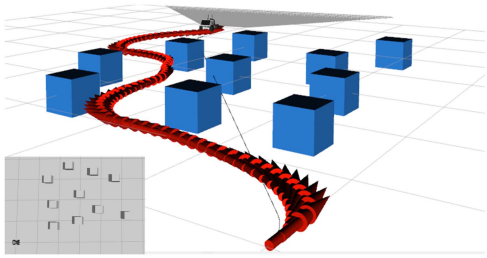


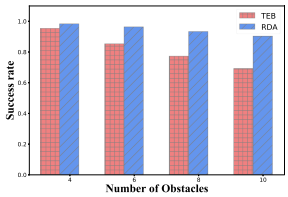
Fig. 6. RDA Simulation in the Gazebo.

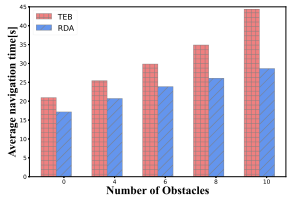
However, in the considered scenario, both the vehicle and the obstacle are rectangles, making the solution to PMA infeasible.

B. Gazebo Simulation

To verify the effectiveness of the proposed RDA under practical sensor and motion constraints, we implement the RDA and TEB methods in Gazebo [28], which is a high-fidelity robotic simulator with close-to-reality sensory data and motion dynamics. The scenario consists of an autonomous vehicle and 10 obstacles (i.e., marked as blue cubes with the size of $0.35\,\mathrm{m} \times 0.35\,\mathrm{m} \times 0.35\,\mathrm{m}$). The receding horizon of RDA is set to 20. Under the above setting, we execute 50 trials with random goal points and positions of obstacles. Fig. 6 illustrates the trajectory of RDA in one particular case when the robot moves from the starting point $(0, 0)$ to the target point $(5, 4)$. The odometry of the vehicle with the RDA planner is illustrated by the red arrows. It can be seen that the proposed RDA generates a smooth and collision-free trajectory in the dense-obstacle scenario.

Performance comparison between RDA and TEB is shown in Fig. 7. The success rates (a successful trail is defined as no collision or stuck during the navigation) of both schemes under the different number of obstacles are shown in Fig. 7(a). The two simulated schemes achieve similar success rates in environments with few obstacles. However, RDA significantly outperforms TEB in environments with dense obstacles. The average navigation time (i.e., the average time required by the planner to move from the starting point to the goal) under





(a) Success rate (b) Average navigation time

Fig. 7. Performance comparison of RDA and TEB.

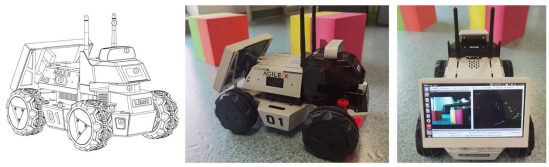


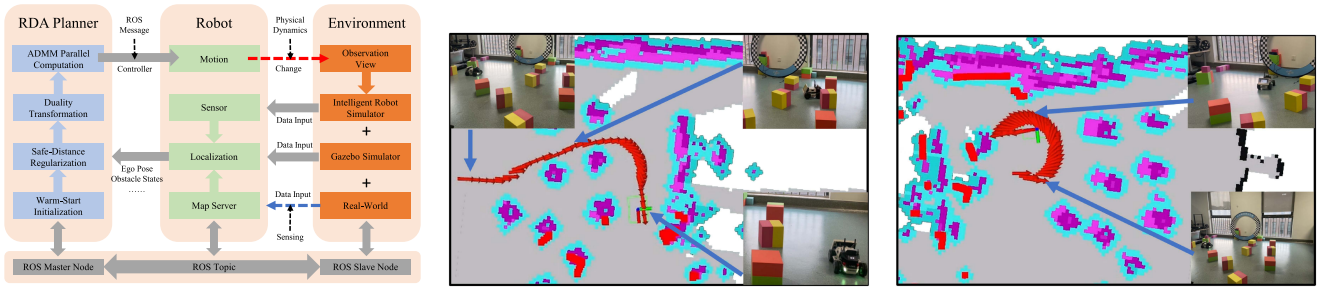
Fig. 8. The hardware of the Ackermann robot.

the different number of obstacles is shown in Fig. 7(b). RDA reduces the navigation time by up to 40%, which demonstrates the necessity of predictive path optimization.

C. Hardware Experiment

To evaluate the performance of RDA in real-time operating systems under limited hardware resources, we implement RDA and TEB planners in an autonomous robot shown in Fig. 8. The robot has four wheels and adopts Ackermann steering, which can be viewed as a size-scaled road vehicle. Its sensor suite consists of a 2D Lidar and multiple cameras (including RGB and depth cameras). NVIDIA Jetson Nano is adopted as the onboard computing platform for processing the sensor data and executing the navigation package. The software architecture of our implementation is shown in Fig. 9(a), which consists of the RDA planner module and the robot module. Data sharing between two modules is realized via ROS communications, where master/slave nodes publish/subscribe topics carrying the controller, map, and odometry information. Note that we convert the obstacles in the costmap into non-point mass representation via DBSCAN [29], and the experimental setting is the same as Gazebo.

Based on the above implementation, the trajectories (red arrows) of RDA and TEB planners of one trial are shown in Fig. 9(a) and (b), respectively. It can be seen that the robot powered by RDA achieves the target point successfully with a smooth curve. In contrast, the TEB planner fails to find a feasible route to cross those narrow gaps, and the robot eventually gets stuck after a sequence of replanning. This demonstrates the effectiveness and robustness of RDA in real time systems and dense obstacle environments. Note that we adopt the default parameters for TEB. One may fine-tune these parameters to generate a feasible path. However, TEB involves large-scale parameters without interpretability, leading to a large amount of manual effort for calibration whenever the robot enters a new complex scenario. In contrast, RDA only involves several tuning parameters (e.g., η for regularization and ρ for ADMM augmentation), which significantly facilitates practical deployment.



(a) The framework (b) RDA trajectory (c) TEB trajectory

Fig. 9. Framework and the trajectory comparison of RDA and TEB in a real-world experiment.

VI. Conclusion

In this letter, we have presented RDA, an accelerated optimization based collision-free motion planner for autonomous navigation. RDA can support Ackermann dynamics and enable adaptive safety distances, non-point-mass representation of obstacles, and parallel computing of optimal trajectories. Various simulations and real-world experiments have shown that the proposed RDA can achieve a significantly shorter execution time compared to existing OBCA; the failure rate and the navigation time of RDA can be reduced by $10 \sim 40\%$ compared with various benchmark schemes such as TEB and PMA. Future directions include high-speed vehicle and multi-vehicle RDA, as well as real-world verification.

References

[1] Z. Zhang, R. Han, and J. Pan, "An efficient centralized planner for multiple automated guided vehicles at the crossroad of polynomial curves," *IEEE Robot. Automat. Lett.*, vol. 7, no. 1, pp. 398–405, Jan. 2022.

[2] M.-O. Sonneberg, M. Leyerer, A. Kleinschmidt, F. Knigge, and M. H. Breitner, "Autonomous unmanned ground vehicles for urban logistics: Optimization of last mile delivery operations," in *Proc. 52nd Hawaii Int. Conf. System Sci.*, 2019, pp. 1538–1547.

[3] D. González, J. Pérez, V. Milanés, and F. Nashashibi, "A review of motion planning techniques for automated vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 17, no. 4, pp. 1135–1145, Apr. 2016.

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

[5] M. Montemerlo et al., "Junior: The stanford entry in the urban challenge," *J. Field Robot.*, vol. 25, no. 9, pp. 569–597, 2008.

[6] R. R. Radaelli, C. Badue, M. A. Gonçalves, T. Oliveira-Santos, and A. F. D. Souza, "A motion planner for car-like robots based on rapidly-exploring random trees," in *Proc. Ibero- Amer. Conf. Artif. Intell.*, 2014, pp. 469–480.

[7] J. Funke et al., "Up to the limits: Autonomous Audi TTS," in *Proc. IEEE Intell. Veh. Symp.*, 2012, pp. 541–547.

[8] M. Ammour, R. Orjuela, and M. Basset, "Collision avoidance for autonomous vehicle using MPC and time varying sigmoid safety constraints," *IFAC-PapersOnLine*, vol. 54, no. 10, pp. 39–44, 2021.

[9] F. Rey, Z. Pan, A. Hauswirth, and J. Lygeros, "Fully decentralized ADMM for coordination and collision avoidance," in *Proc. IEEE Eur. Control Conf.*, 2018, pp. 825–830.

[10] Z. Wang, Y. Zheng, S. E. Li, K. You, and K. Li, "Parallel optimal control for cooperative automation of large-scale connected vehicles via ADMM," in *Proc. IEEE 21st Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1633–1639.

[11] Z. Wang, X. Zhou, C. Xu, and F. Gao, "Geometrically constrained trajectory optimization for multicopters," *IEEE Trans. Robot.*, vol. 38, no. 5, pp. 3259–3278, Oct. 2022.

[12] X. Zhang, A. Liniger, and F. Borrelli, "Optimization-based collision avoidance," *IEEE Trans. Control Syst. Technol.*, vol. 29, no. 3, pp. 972–983, May 2021.

[13] W. Xia, W. Wang, and C. Gao, "Trajectory optimization with obstacles avoidance via strong duality equivalent and hp-pseudospectral sequential convex programming," *Optimal Control Appl. Methods*, vol. 43, no. 2, pp. 566–587, 2022.

[14] Z. Cheng, J. Ma, X. Zhang, C. W. de Silva, and T. H. Lee, "ADMM-based parallel optimization for multi-agent collision-free model predictive control," 2021, *arXiv:2101.09894*.

[15] R. Firoozi, L. Ferranti, X. Zhang, S. Nejadnik, and F. Borrelli, "A distributed multi-robot coordination algorithm for navigation in tight environments," 2020, *arXiv:2006.11492*.

[16] J. Funke, M. Brown, S. M. Erlien, and J. C. Gerdes, "Collision avoidance and stabilization for autonomous vehicles in emergency scenarios," *IEEE Trans. Control Syst. Technol.*, vol. 25, no. 4, pp. 1204–1216, Jul. 2017.

[17] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Safe driving envelopes for shared control of ground vehicles," *IFAC Proc. Volumes*, vol. 46, no. 21, pp. 831–836, 2013.

[18] C. Rösmann, F. Hoffmann, and T. Bertram, "Kinodynamic trajectory optimization and control for car-like robots," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 5681–5686.

[19] E. J. Rodrıǵuez-Seda, C. Tang, M. W. Spong, and D. M. Stipanović, "Trajectory tracking with collision avoidance for nonholonomic vehicles with acceleration constraints and limited sensing," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1569–1592, 2014.

[20] Z. Zhang, Y. Zhang, R. Han, L. Zhang, and J. Pan, "A generalized continuous collision detection framework of polynomial trajectory for mobile robots in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 7, no. 4, pp. 9810–9817, Oct. 2022.

[21] S. Boyd et al., "Distributed optimization and statistical learning via the alternating direction method of multipliers," *Found. Trends Mach. Learn.*, vol. 3, no. 1, pp. 1–122, 2011.

[22] S. Boyd, S. P. Boyd, and L. Vandenberghe, *Convex Optimization*. Cambridge, MA, USA: Cambridge Univ. Press, 2004.

[23] T. Barfoot, J. R. Forbes, and P. T. Furgale, "Pose estimation using linearized rotations and quaternion algebra," *Acta Astronautica*, vol. 68, no. 1/2, pp. 101–112, 2011.

[24] K. M. Lynch and F. C. Park, *Modern Robotics*. Cambridge, MA, USA: Cambridge Univ. Press, 2017.

[25] S. Diamond and S. Boyd, "CVXPY: A Python-embedded modeling language for convex optimization," *J. Mach. Learn. Res.*, vol. 17, no. 83, pp. 1–5, 2016.

[26] R. Han et al., "RDA: An accelerated collision-free motion planner for autonomous navigation in cluttered environments," 2022, *arXiv:2210.00192*.

[27] A. M. Shkel and V. Lumelsky, "Classification of the Dubins set," *Robot. Auton. Syst.*, vol. 34, no. 4, pp. 179–202, 2001.

[28] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2004, vol. 3, pp. 2149–2154.

[29] M. Ester et al., "A density-based algorithm for discovering clusters in large spatial databases with noise," *kdd*, vol. 96, no. 34, 1996, pp. 226–231.