Edge Accelerated Robot Navigation With Collaborative Motion Planning

Guoliang Li, *Graduate Student Member, IEEE*, Ruihua Han, *Member, IEEE*, Shuai Wang, *Senior Member, IEEE*, Fei Gao, *Member, IEEE*, Yonina C. Eldar, *Fellow, IEEE*, and Chengzhong Xu, *Fellow, IEEE*

***Abstract*—Low-cost distributed robots suffer from limited onboard computing power, resulting in excessive computation time when navigating in cluttered environments. This article presents Edge Accelerated Robot Navigation (EARN), to achieve real-time collision avoidance by adopting collaborative motion planning. As such, each robot can dynamically switch between a conservative motion planner executed locally to guarantee safety (e.g., path-following) and an aggressive motion planner executed nonlocally to guarantee efficiency (e.g., overtaking). In contrast to existing motion planning approaches that ignore the interdependency between low-level motion planning and high-level resource allocation, EARN adopts model predictive switching (MPS) that maximizes the expected switching gain with respect to robot states and actions under computation and communication resource constraints. The MPS problem is solved by a tightly coupled decision making and motion planning framework based on bilevel mixed-integer nonlinear programming and penalty dual decomposition. We validate the performance of EARN in indoor simulation, outdoor simulation, and real-world environments. Experiments show that EARN achieves significantly smaller navigation time and higher success rates than state-of-the-art navigation approaches.**

***Index Terms*—Collaborative computing, distributed robotics, model predictive switching (MPS), motion planning.**

Manuscript received 9 November 2023; revised 15 April 2024; accepted 1 June 2024. Date of publication 18 July 2024; date of current version 18 April 2025. Recommended by Technical Editor Y. Gu and Senior Editor M. Indri. This work was supported in part by the Science and Technology Development Fund of Macao S.A.R (FDCT) under Grant 0123/2022/AFJ and Grant 0081/2022/A2, in part by the National Natural Science Foundation of China under Grant 62371444, in part by the Guangdong Basic and Applied Basic Research Project under Grant 2021B1515120067, in part by the Direct Drive Tech Cooperation Project, and in part by the Shenzhen Science and Technology Program under Grant RCYX20231211090206005. *(Guoliang Li and Ruihua Han contributed equally to this work.) (Corresponding authors: Shuai Wang; Chengzhong Xu.)*

Guoliang Li and Chengzhong Xu are with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), Department of Computer and Information Science, University of Macau, Macau 999078, China (e-mail: li.guoliang@connect.um.edu.mo; czxu@um.edu.mo).

Ruihua Han is with the Department of Computer Science, The University of Hong Kong, Hong Kong (e-mail: hanrh@connect.hku.hk).

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

Fei Gao is with the State Key Laboratory of Industrial Control Technology and the Huzhou Institute, Zhejiang University, Zhejiang 310027, China (e-mail: fgaoaa@zju.edu.cn).

Yonina C. Eldar is with the Faculty of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot 7610001, Israel (e-mail: yonina.eldar@weizmann.ac.il).

The code of this paper will be released at https://github.com/GuoliangLI1998/EARN.

This article has supplementary material provided by the authors and color versions of one or more figures available at https://doi.org/10.1109/TMECH.2024.3419436.

Digital Object Identifier 10.1109/TMECH.2024.3419436

I. Introduction

NAVIGATION is a fundamental task for mobile robots, which determines a sequence of control commands to move the robot safely from its current state to a target state [1]. The navigation time and success rate depend on how fast the robot can compute a trajectory. The computation time is proportional to the number of obstacles (spatial) and the length of prediction horizon (temporal) [2]. In cluttered environments with numerous obstacles, the shape of obstacles should also be considered, and the computation time is further multiplied by the number of surfaces of each obstacle [3]. Therefore, navigation in cluttered environments is challenging for low-cost robots.

Currently, most existing approaches reduce the computation time from an algorithm design perspective, e.g., using heuristics [4], approximations [5], parallelizations [6], [7], [8], [9], or learning [10], [11], [12] techniques. This article accelerates the robot navigation from an integrated networking and algorithm design perspective, i.e., the robot can opportunistically execute advanced navigation algorithms by accessing a proximal edge computing server [13], [14], [15]. However, design and implementation of such systems are nontrivial. First, it involves periodic data exchange between robots and servers. Therefore, the server should have low-latency access to the robot; otherwise, communication delays may lead to collisions. Second, different robots may compete for a common computing server, and the computation time increases quickly as the number of robots increases. Third, in case of no proximal server, the robots should be able to navigate individually with only onboard computing resources. These observations imply that the new paradigm needs to maximize the navigation efficiency under practical resource constraints, for which the existing local [6], [7], [8], [9] or edge [13], [14], [15] planning methods become inefficient and unsafe, as they ignore the interdependence between low-level motion planning (e.g., robot states and actions) and high-level decision-making (e.g., resource management).

This article proposes Edge Accelerated Robot Navigation (EARN), which is a collaborative motion planning (CMP)

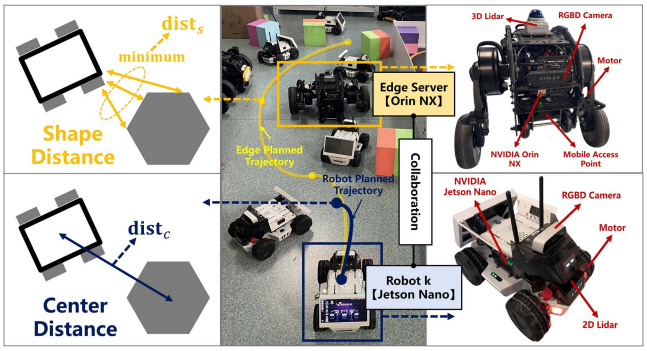


Fig. 1. Low-cost robots execute shape distance collision avoidance assisted by a proximal edge server based on EARN.

framework that enables resource-adaptive switching between a local motion planner at the robot and an edge motion planner at the server. As shown in Fig. 1, with only onboard computer (i.e., Jetson Nano), the robot can only adopt center distance collision avoidance, which generates a short trajectory (marked in blue) blocked by the narrow gap. In contrast, by leveraging EARN, the robot can opportunistically execute shape distance collision avoidance by accessing a proximal edge server (i.e., Orin NX), which generates a long trajectory (marked in yellow) reaching the goal. EARN adopts model predictive switching (MPS) to realize tightly coupled decision making and motion planning (T-DMMP). The MPS maximizes the *expected switching gain* under computation and communication resource constraints, which is in contrast to existing model predictive control (MPC) where robot states and actions are the only considerations. In particular, for high-level resource management, we propose a bilevel mixed-integer nonlinear programming (B-MINLP) algorithm for decision-making, which can automatically identify switching-beneficial robots while orchestrating computation and communication resources. For low-level motion planning, we incorporate the high-level decision variables into a set of conditional collision avoidance constraints, and propose a penalty dual decomposition (PDD) algorithm, which computes collision-free trajectories in parallel with convergence guarantee. We implement our methods by robot operation system (ROS) and validate them in the high-fidelity car-learning-to-act (CARLA) simulation. Experiments confirm the superiority of the proposed scheme compared with various benchmarks in both indoor and outdoor scenarios. We also implement EARN in a multirobot testbed, where real-world experiments are conducted to demonstrate the practical applicability of EARN.

The contribution of this article is summarized as follows.

1) Propose EARN, which enables T-DMMP in dynamic environments[1] under resource constraints based on MPS.
2) Propose B-MINLP and PDD algorithms, which ensure smooth planner switching and real-time motion planning.
3) Implementation of EARN in both simulation and real-world environments.
4) Evaluations of the performance gain brought by EARN compared with extensive benchmarks.

The rest of this article is organized as follows. Section II reviews the related work. Section III presents the architecture and mechanism of the proposed EARN. Section IV presents the core optimization algorithms for EARN. Simulations and experiments are demonstrated and analyzed in Section V. Finally, Section VI concludes this article.

II. Related Work

Motion planning: is a challenge when navigating low-cost robots, due to contradiction between the stringent timeliness constraint and the limited onboard computing capability. Conventional heuristic methods (e.g., path following (PF) [4], spatial cognition [16]) are overconservative, and the robot may get stuck in cluttered environments. Emerging optimization techniques can overcome this issue by explicitly formulating collision avoidance as distance constraints. MPC [2], [3], [4], [6], [7], [8], [9] is the most widely used optimization algorithm, which leverages constrained optimization for generating high-performance collision-free trajectories in complex scenarios. Nonetheless, MPC could be time-consuming when the number of obstacles is large.

Optimization-based collision avoidance: can be accelerated in two ways. First, imitation learning methods [10], [11], [12] can learn from optimization solvers' demonstrations using deep neural networks. As such, iterative optimization is transformed into a feedforward procedure that can generate actions in milliseconds. However, learning-based methods may break down if the target scenario contains examples outside the distribution of the training dataset [12]. Second, the computation time can be reduced by parallel optimizations [6], [7], [8], [9]. For instance, the alternating direction method of multipliers (ADMMs) has been adopted in multirobot navigation systems [9], which decomposes a large centralized problem into small subproblems that are solved in parallel for each robot. By applying parallel computation to obstacle avoidance, ADMM has been shown to accelerate autonomous navigation in cluttered environments [8]. However, *ADMM may diverge when solving nonconvex problems [17]. Unfortunately, motion planning problems are nonconvex due to the nonlinear vehicle dynamics and irregular obstacle shapes [3], [8]. Here, we develop a PDD planner that converges to a Karush–Kuhn–Tucker (KKT) solution and overcomes the occasional failures of ADMM.*

Cloud and edge robotics: are emerging paradigms to accelerate robot navigation [13]. The idea is to allow robots to access proximal computing resources. For example, robot inference and learning applications, such as object recognition and grasp planning, can be offloaded to cloud, edge, and fog as a service [14]. Edge-assisted autonomous driving was investigated in Atik et al. [18]'s work, which offloads the heavy tasks from low-cost vehicular computers to powerful edge servers. However, this type of approach would introduce additional communication latency [19]. To reduce the communication latency, a partial offloading scheme was proposed for vision-based robot navigation [15]. A priority-aware robot scheduler was proposed

[1]States (including poses and velocities) of the obstacles and resources (including communication and computation) of the system are both dynamic.

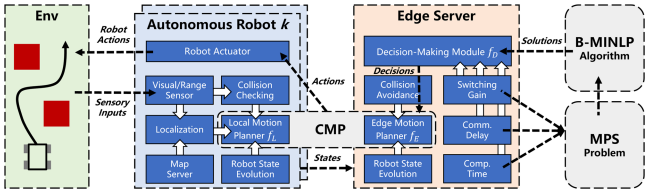


Fig. 2. Architecture of EARN, which consists of a planner-switching decision-making module and collaborative motion planners deployed at the robots and server.

to scale up collaborative visual simultaneous localization and mapping (SLAM) service in edge offloading settings [20]. A cloud robotics platform FogROS2 was proposed to effectively connect robot systems across different physical locations, networks, and data distribution services [21]. In recent DARPA SubT challenge [22], it is found that for multirobot motion planning, it is necessary to develop a higher level "mission planner" that assigns different tasks to different robots according to their states and capabilities. Nonetheless, *current cloud and edge robotics schemes ignore the interdependence between the low-level motion planning (e.g., robot states and actions) and the high-level decision-making (e.g., planner switching and resource management). In contrast, EARN models such interdependence explicitly through MPS and achieves the T-DMMP feature by two optimization algorithms.* Our method belongs to vertical collaboration that split computing loads between the server and the robot, which differs from conventional horizontal collaboration, e.g., cooperative motion planning [6].

III. Tightly Coupled Decision-Making and Motion Planning

The architecture of EARN is shown in Fig. 2, which adopts a decision-making module f_D to execute planner switching between a low-complexity local motion planner $f_L(\cdot)$ deployed at K robots and a high-performance edge motion planner $f_E(\cdot)$ deployed at the edge server. Decision-making operates at a low frequency (e.g., 1 Hz) and motion planning operates at a high frequency (e.g., $10 \sim 100$ Hz). In the following subsections, we first present the CMP formulation that determines f_L and f_E. Then, we present the MPS formulation that determines f_D.

A. Collaborative Motion Planning

At the tth time slot, the kth robot (with $1 \le k \le K$) estimates its current state $\mathbf{s}_{k,t} = (x_{k,t}, y_{k,t}, \theta_{k,t})$ and obstacle states $\{\mathbf{o}_{1,t}, \ldots, \mathbf{o}_{M,t}\}$ with $\mathbf{o}_{m,t} = (a_{m,t}, b_{m,t}, \phi_{m,t})$ (with $1 \le m \le M$), where $(x_{k,t}, y_{k,t})$ and $\theta_{k,t}$ are positions and orientations of the kth robot, and $(a_{m,t}, b_{m,t})$ and $\phi_{m,t}$ are positions and orientations of the mth obstacle. The edge server collects the state information from K robots and executes the decision-making module f_D. The output of f_D is a set of one-zero decision variables $\{\alpha_1, \ldots, \alpha_K\} \in \{0,1\}^K$, where $\alpha_k = 1$ represents the kth robot being selected for edge planning and $\alpha_k = 0$ represents the kth robot being selected for local planning. Given α_k, the local planner or edge planner can generate control vectors by minimizing the distances between the robot's footprints $\{\mathbf{s}_{k,t}\}$ and the target waypoints $\{\mathbf{s}^{\diamond}_{k,t}\}$. Denoting the current time as $t = 0$, the state evolution model $\mathbf{s}_{k,t+1} = E_k(\mathbf{s}_{k,t}, \mathbf{u}_{k,t})$ is adopted to predict the future trajectories $\{\mathbf{s}_{k,t}\}_{t=0}^{H}$, where H is the length of prediction horizon and E_k is determined by Ackermann kinetics

$$E_k(\mathbf{s}_{k,t}, \mathbf{u}_{k,t}) = \mathbf{A}_{k,t}\mathbf{s}_{k,t} + \mathbf{B}_{k,t}\mathbf{u}_{k,t} + \mathbf{c}_{k,t}\,\forall k, t \quad (1)$$

where $(\mathbf{A}_{k,t}, \mathbf{B}_{k,t}, \mathbf{c}_{k,t})$ are coefficient matrices defined in (8)–(10) of [8, Sec. III-B]. Furthermore, based on the state evolution model, we compute the distance $\{\mathbf{dist}(\mathbf{s}_{k,t}, \mathbf{o}_{m,t})\}_{t=0}^{H}$ between the kth robot and the mth obstacle at all time slots within the horizon, where the states $\{\mathbf{o}_{m,t}\}_{t=1}^{H}$ are obtained from motion prediction. Consequently, the CMP problem for the kth robot is formulated as

$$\min_{\{\mathbf{s}_{k,t}, \mathbf{u}_{k,t}\}_{t=0}^{H}} \sum_{t=0}^{H} \left\|\mathbf{s}_{k,t} - \mathbf{s}^{\diamond}_{k,t}\right\|^2 \quad (2a)$$

$$\text{s.t.} \quad \mathbf{s}_{k,t+1} = E_k(\mathbf{s}_{k,t}, \mathbf{u}_{k,t}) \qquad \forall t \quad (2b)$$

$$\mathbf{u}_{\min} \preceq \mathbf{u}_{k,t} \preceq \mathbf{u}_{\max} \qquad \forall t \quad (2c)$$

$$\mathbf{a}_{\min} \preceq \mathbf{u}_{k,t+1} - \mathbf{u}_{k,t} \preceq \mathbf{a}_{\max} \qquad \forall t \quad (2d)$$

$$\Psi(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}, d_{\text{safe}} | \alpha_k) \ge 0 \qquad \forall m \in \mathcal{M}(k), t \quad (2e)$$

where Ψ is a conditional collision avoidance function

$$\Psi(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}, d_{\text{safe}} | \alpha_k) = \alpha_k \left(\mathbf{dist}(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) - d_{\text{safe}}\right) \quad (3)$$

and $\mathcal{M}(k)$ is the set of obstacles within the local map of robot k with its cardinality denoted as $|\mathcal{M}(k)|$. Constants $\mathbf{u}_{\min}$ ($\mathbf{a}_{\min}$) and $\mathbf{u}_{\max}$ ($\mathbf{a}_{\max}$) are the minimum and maximum limits of the control (acceleration) vector, respectively.

Now consider two cases. If $\alpha_k = 1$, the collision avoidance constraints exist, and the associated optimal solution $\{\mathbf{s}^{[1]}_{k,t}, \mathbf{u}^{[1]}_{k,t}\}_{t=0}^{H}$ to problem (2) leads to a *proactive* collision avoidance policy. In this case, problem (2) is solved at the edge server and we define $f_E(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}_{m=1}^{M}) = \mathbf{u}^{[1]}_{k,t}$. On the other hand, if $\alpha_k = 0$, the collision avoidance constraints are discarded, and the optimal $\{\mathbf{s}^{[0]}_{k,t}, \mathbf{u}^{[0]}_{k,t}\}_{t=0}^{H}$ to problem (2) leads to a *reactive* collision avoidance policy. In this case, problem (2) is solved at the robot k and we define $f_L(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}_{m=1}^{M}) = \mathbf{u}^{[0]}_{k,t}$ if the robot sensor detects no obstacles ahead within a certain braking distance d_{B} and $f_L(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}_{m=1}^{M}) = \mathbf{u}_B$ otherwise, where $\mathbf{u}_B$ denotes the braking action.

B. Model Predictive Switching

The decision making module aims to find the optimal decision variables $\{\alpha^*_k\}$ by maximizing the expected switching gain over all robots under the communication and computation resource constraints. First, the communication latency between the kth robot and the edge server should be smaller than a certain threshold D_{th} if $\alpha_k = 1$, i.e., $\alpha_k D_k(\mathbf{s}_{k,t}) \le D_{\text{th}}$, where D_k is a function of the robot state $\mathbf{s}_{k,t}$. Second, the total computation time at the server should not exceed a certain threshold C_{th}, i.e., $\sum_k \alpha_k C_k(H, |\mathcal{M}(k)|) \le C_{\text{th}}$, where C_k is a function of the

prediction time length H and the number of obstacles $|\mathcal{M}(k)|$. Third, the expected switching gain G_k is defined as the difference between the moving distance of planner f_L and that of f_E at time H. Let $\mathbf{dist}_c(\mathbf{s}_{k,H}^{[1]}, \mathbf{s}_{k,0}) = \|[x_{k,H}^{[1]} - x_{k,0}, y_{k,H}^{[1]} - y_{k,0}]\|$ represent the center distance between states $\mathbf{s}_{k,H}^{[1]}$ and $\mathbf{s}_{k,0}$, we have

$$\begin{aligned} &G_k\left(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}, f_E, f_L\right) \\ &= \mathbf{dist}_c\left(\mathbf{s}_{k,H}^{[1]}, \mathbf{s}_{k,0}\right) - \mathbf{dist}_c\left(\mathbf{s}_{k,H}^{[0]}, \mathbf{s}_{k,0}\right) \end{aligned} \tag{4}$$

where $\mathbf{s}_{k,H}^{[1]}$ and $\mathbf{s}_{k,H}^{[0]}$ are the optimal solutions for the cases of $\alpha_k = 1$ and $\alpha_k = 0$, respectively.

With the above-mentioned models, the MPS problem is formulated as

$$\max_{\{\alpha_k\}_{k=1}^K} \quad \sum_{k=1}^{K} \alpha_k G_k\left(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}, f_E, f_L\right) \tag{5a}$$

$$\text{s.t.} \quad \{\mathbf{s}_{k,H}^{[\alpha_k]}\} = \arg \min_{\{\mathbf{s}_{k,t}, \mathbf{u}_{k,t}\}_{t=0}^{H}} \left\{ \sum_{t=0}^{H} \left\|\mathbf{s}_{k,t} - \mathbf{s}_{k,t}^{\diamond}\right\|^2 : (2b) - (2e) \right\} \quad \forall k \tag{5b}$$

$$\alpha_k D_k(\mathbf{s}_{k,t}) \le D_{\text{th}} \quad \forall k \tag{5c}$$

$$\sum_{k=1}^{K} \alpha_k C_k(H, |\mathcal{M}(k)|) \le C_{\text{th}} \tag{5d}$$

$$\alpha_k \in \{0, 1\} \quad \forall k. \tag{5e}$$

Denoting the optimal solution to MPS as $\{\alpha_k^*\}_{k=1}^K$, we set $f_D(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}_{m=1}^M, D_{\text{th}}, C_{\text{th}}) = \{\alpha_k^*\}_{k=1}^K$. It can be seen that MPS requires joint considerations of robot states, obstacle states, communications, and computations. Inappropriate decisions would lead to collisions due to local/edge computation timeout or communication timeout.

C. Case Study

To demonstrate the practicability of EARN, a case study is conducted for a five-robot system. We consider two embedded computing chips, i.e., NVIDIA Jetson Nano and NVIDIA Orin NX, and two motion planners, i.e., PF (reactive) [4] and RDA (proactive) [8]. The power and price of Jetson Nano are 10 W and 150 $. The power and price of Orin NX are 25 W and 1000 $. Based on our experimental data, a frequency of 50 Hz can be achieved for PF on both chips. RDA can be executed at a frequency of 5 Hz on Jetson Nano, and 20 Hz on Orin NX, when the number of obstacles is 5 [8].

We consider three schemes as follows.

1) Onboard computing, where all robots execute RDA on Jetson Nano.
2) Enhanced onboard computing, where all robots execute RDA on Orin NX.
3) EARN, where four low-cost robots execute PF on Jetson Nano and one edge server execute RDA on Orin NX.

TABLE I
COMPARISON OF DIFFERENT SCHEMES

Metric \ scheme	EARN	Onboard comp.	Enhanced onboard comp.
Latency (ms)	50	200	50
Power (W)	65	50	125
Price ($)	1600	750	5000

The latency, power, and price of different schemes are illustrated in Table I , and we have the following observations.

1) The average computation latency of EARN (50 ms) is significantly smaller than that of the onboard computing (200 ms).
2) EARN (65 W) is more energy efficient than the enhanced onboard computing (125 W).
3) The total system cost of EARN is significantly smaller than that of enhanced onboard computing (1600 $versus 5000 $).

In summary, EARN is faster than onboard computing, and their time difference is the **acceleration gain**. On the other hand, EARN is more energy-and-cost efficient than enhanced onboard computing, and their cost difference is the **computing sharing gain**. The insight is that a robot is not always encountering challenging environments, and it is possible to use a single server to support multiple robots at different times. The acceleration gain and computing sharing gain brought by EARN increase as the complexity of motion planners increases. This makes EARN suitable for large model empowered robot navigation at the edge.

IV. Algorithm Designs for Decision-Making and Motion Planning

A. Decision-Making

To solve the MPS problem, we need mathematical expressions of the communication latency D_k, the computation latency C_k, and the planner switching gain G_k. First, the function D_k is determined by the signal attenuation Q_k between the kth robot and the server. This Q_k is a function of the robot position $\mathbf{s}_{k,t}$, which is known as radio map. It is not only related to the robot-edge distance, but also related to nonline-of-sight signal propagation (shadowing, reflection, diffraction, blockage) [23]. We adopt NVIDIA Sionna, a widely used ray tracing package [24], to obtain the radio map [e.g., Fig. 8(b)], where similar colors represent similar attenuation (and latency). As such, we can segment the radio map into J regions by color and for the jth region, we select representative positions and ping the communication latencies between the robot and the server using onboard WiFi. The measurements are adopted to estimate the latency distribution of each region. For the radio map in Fig. 8(b), according to the measurements, the communication latency D_k is between 30 and 80 ms for the yellow and light green regions; and larger than 100 ms for dark green regions.

Next, we determine the computing latency C_k. Since a common motion planner has a polynomial-time computational complexity, we can write C_k as

$$C_k = \gamma H |\mathcal{M}(k)|^p + \tau \tag{6}$$

where γ and τ are hardware-dependent hyperparameters estimated from historical experimental data. Moreover, p is the order of complexity, which ranges from 1 to 3.5 and depends on the motion planner.

Finally, we determine G_k, which is coupled with the motion planning algorithms f_E and f_L. In particular, if $\mathbf{dist}_c(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) > d_{\mathrm{B}}$, we have $G_k \leq 0$, since the local planning problem is a relaxation of the edge planning problem and the solution of the former is at least no worse than the latter. On the other hand, if $\mathbf{dist}_c(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) \leq d_{\mathrm{B}}$, then $\mathbf{dist}_c(\mathbf{s}_{k,H}^{[0]}, \mathbf{s}_{k,0}) = 0$ as the braking signal is generated and the robot stops. Combining the above-mentioned two cases, G_k is equivalently written as

$$G_k = I_k \, \mathbf{dist}_c \left(\mathbf{s}_{k,H}^{[1]}, \mathbf{s}_{k,0} \right) \tag{7}$$

where

$$I_k = \begin{cases} 1, & \text{if } \mathbf{dist}_c(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) \leq d_{\mathrm{B}}, \mathbf{o}_{m,t} \in \mathcal{W}_k \\ 0, & \text{otherwise} \end{cases} \tag{8}$$

and set $\mathcal{W}_k$ represents the global path formed by waypoints $\{\mathbf{s}_{k,t}^{\diamond}\}$. Note that we have replaced $G_k \leq 0$ with $G_k = 0$, which would not affect the solution of the problem.

Based on the above-mentioned derivations of D_k, C_k, and G_k, the original MPS problem (5) is transformed into

$$\max_{\{\alpha_k\}_{k=1}^K} \quad \sum_{k=1}^{K} \alpha_k I_k \mathbf{dist}_c \left(\mathbf{s}_{k,H}^{[1]}, \mathbf{s}_{k,0} \right) \tag{9a}$$

$$\text{s.t.} \quad \text{constraints (5b), (5e)} \tag{9b}$$

$$\alpha_k D_k(\mathbf{s}_{k,t}) \leq D_{\mathrm{th}} \quad \forall k \tag{9c}$$

$$\sum_{k=1}^{K} \alpha_k \gamma H |\mathcal{M}(k)|^p + \tau \leq C_{\mathrm{th}}. \tag{9d}$$

This is a B-MINLP problem due to the constraint (9b). A naive approach is to solve the inner problem (5b) for all robots and conduct an exhaustive search over $\{\alpha_k\}$. However, the resultant complexity would be $\mathcal{O}(2^K)$, which cannot meet the frequency requirement of the decision-making module. To this end, we propose a low-complexity method summarized in Algorithm 1, which consists of three sequential steps as follows.

1) Prune out impossible solutions for space reduction. Specifically, we conduct the feasibility check of (9c)–(9d) and switching gain check of $\{I_k\}$ for all the robots. This yields sets $\mathcal{A}$ (robots satisfying (9c)–(9d) and having positive values of I_k) and $\mathcal{I}$ (otherwise). Any robot in $\mathcal{I}$ is pruned out, without the need for further motion planning.
2) Leverage parallel motion planning for fast trajectory computations.
3) Put $\{\alpha_k = 0\}_{k \in \mathcal{I}}$ and $\{\mathbf{s}_{k,t}^{[1]}\}_{k \in \mathcal{A}}$ into problem (9), and solve the resultant problem using integer linear programming. The integer programming can be either solved by CVXPY, or accelerated by other penalty continuous relaxation approaches.

Algorithm 1: B-MINLP Decision Making.

Input: robots' states $\{\mathbf{s}_{k,t}\}_{k=1}^K$, obstacles' states $\{\mathbf{o}_{m,t}\}_{m=1}^M$, communication latency threshold D_{th}, computation latency threshold C_{th}.
Initialize: $\mathcal{A} = \mathcal{I} = \emptyset$
for *robot* $k = 1, \cdots, K$ **do**
  if $\alpha_k = 1$ *satisfies* (9c)–(9d) *and* $I_k = 1$ **then**
    Set $\alpha_k = 1$ and update $\mathcal{A} \leftarrow \{k\} \cup \mathcal{A}$
  else
    Set $\alpha_k = 0$ and update $\mathcal{I} \leftarrow \{k\} \cup \mathcal{I}$
  end
end
for *robot* $k \in \mathcal{A}$ **do**
  Compute $\{\mathbf{s}_{k,t}^{[1]}\}$ in (5b) using PDD
end
Solve (9) with integer linear programming.
Output: $f_D(\mathbf{s}_{k,t}, \{\mathbf{o}_{m,t}\}_{m=1}^M, D_{\mathrm{th}}, C_{\mathrm{th}}) = \{\alpha_k^*\}_{k=1}^K$.

B. Motion Planning

When $\alpha_k = 0$, the local motion planner is adopted. To ensure safety, the distances between the robot and other obstacles are first computed using the onboard sensor. If the robot sensor detects any obstacle ahead within a certain distance d_{B}, then the braking action $\mathbf{u}_B$ is directly adopted without further computation (hence this also reduces computation complexity). Otherwise, the following planning problem (which corresponds to (5b) with $\alpha_k = 0$) is considered:

$$\min_{\{\mathbf{s}_{k,t}, \mathbf{u}_{k,t}\}_{t=0}^H} \quad \sum_{t=0}^{H} \left\| \mathbf{s}_{k,t} - \mathbf{s}_{k,t}^{\diamond} \right\|^2 \tag{10a}$$

$$\text{s.t.} \quad \text{constraints (2b)} - \text{(2d)}. \tag{10b}$$

The above-mentioned problem can be independently solved at each robot as there is no coupling among different k. Combining the above-mentioned two cases, the action of robot k can be represented as

$$\mathbf{u}_{k,t}^{[0]} = \begin{cases} \mathbf{u}_{\mathrm{B}}, & \text{if } \mathbf{dist}_c(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) \leq d_{\mathrm{B}}, \mathbf{o}_{m,t} \in \mathcal{W}_k \\ \mathbf{u}_{k,t}^*, & \text{otherwise} \end{cases} \tag{11}$$

where $\{\mathbf{s}_{k,t}^*, \mathbf{u}_{k,t}^*\}$ is the optimal solution to (10) and the set $\mathcal{W}_k$ denotes the global path formed by waypoints $\{\mathbf{s}_{k,t}^{\diamond}\}$, which is given by

$$\mathcal{W}_k = \left\{ \mathbf{s} : \|\mathbf{s} - \mathbf{s}_{k,t}^{\diamond}\| \leq \delta \, \forall t \right\} \tag{12}$$

where δ is the width of the path.

When $\alpha_k = 1$, the edge motion planner is adopted, which corresponds to (5b) with $\alpha_k = 1$

$$\min_{\{\mathbf{s}_{k,t}, \mathbf{u}_{k,t}\}_{t=0}^H} \quad \sum_{t=0}^{H} \left\| \mathbf{s}_{k,t} - \mathbf{s}_{k,t}^{\diamond} \right\|^2 \tag{13a}$$

$$\text{s.t.} \quad \text{constraints (2b)} - \text{(2e)}. \tag{13b}$$

To solve the above-mentioned problem, we need an explicit form of $\mathbf{dist}(\mathbf{s}_{k,t}, \mathbf{o}_{m,t})$ in constraint (2e). Here, in contrast to

conventional approaches that model the obstacle as a point [25], i.e., $\mathbf{dist}(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) = \mathbf{dist}_c(\mathbf{s}_{k,t}, \mathbf{o}_{m,t})$, we take the obstacles' shapes into account and model them as polyhedrons as in [3] and [8], i.e., $\mathbf{dist}(\mathbf{s}_{k,t}, \mathbf{o}_{m,t}) = \mathbf{dist}_s(\mathbb{G}_{k,t}, \mathbb{O}_{m,t})$. However, if the obstacle shapes are considered, the computation speed would become a challenge, which hinders the practical application of [3] and [8]. This motivates us to develop a PDD motion planner for fast parallel optimization under shape distance model $\mathbf{dist}_s$. Note that PDD is different from RDA [8], as PDD incorporates the online calibration of penalty parameters into RDA, leading to convergence guaranteed robot navigation.

To begin with, we define the shape models $\mathbb{G}_{k,t}$ and $\mathbb{O}_{m,t}$ for robot and obstacle [8]. We omit the subscript of robot index k for simplicity, and set $\mathbb{G}_t$, which represents the geometric region occupied by the robot, is related to its state $\mathbf{s}_t$ and its shape $\mathbb{Z}$

$$\mathbb{G}_t(\mathbf{s}_t, \mathbb{Z}) = \{\mathbf{z} \in \mathbb{Z} | \mathbf{R}(\mathbf{s}_t)\mathbf{z} + \mathbf{p}(\mathbf{s}_t)\} \tag{14}$$

where $\mathbf{R}(\mathbf{s}_t) \in \mathbb{R}^{3\times3}$ is the rotation matrix related to θ_t and $\mathbf{p}(\mathbf{s}_t)$ is the translation vector related to (x_t, y_t). The set $\mathbb{Z} = \{\mathbf{z} \in \mathbb{R}^3 | \mathbf{G}\mathbf{z} \preceq \mathbf{g}\}$ represents the robot shape, where $\mathbf{G} \in \mathbb{R}^{l\times3}$ and $\mathbf{g} \in \mathbb{R}^l$ (l is the number of surfaces for the robot). Similarly, the mth obstacle can be represented by $\mathbb{O}_{m,t} = \{\mathbf{z} \in \mathbb{R}^3 | \mathbf{H}_{m,t}\mathbf{z} \preceq \mathbf{h}_{m,t}\}$. To determine whether the robot collides with the obstacle, the distances between any two points within $\mathbb{G}_t$ and $\mathbb{O}_{m,t}$ are computed. If the distance is smaller than the safe distance d_{safe}, a collision is likely to happen. Consequently, constraint (2e) is equivalently written as

$$\mathbf{dist}_s(\mathbb{G}_t, \mathbb{O}_{m,t}) \geq d_{\text{safe}} \; \forall m \in \mathcal{M}(k), t. \tag{15}$$

Function $\mathbf{dist}_s$ is not analytical, but can be equivalently transformed into its dual form [3], [8]

$$\begin{aligned} &\left\|{\mathbf{H}_{m,t}}^T \boldsymbol{\lambda}_{m,t}\right\| \leq 1 \\ &\boldsymbol{\lambda}_{m,t} \succeq \mathbf{0}, \; \boldsymbol{\mu}_{m,t} \succeq \mathbf{0}, \quad z_{m,t} \geq 0 \\ &\boldsymbol{\mu}_{m,t}^T \mathbf{G} + \boldsymbol{\lambda}_{m,t}^T \mathbf{H}_{m,t} \mathbf{R}(\mathbf{s}_t) = 0 \\ &\boldsymbol{\lambda}_{m,t}^T \mathbf{H}_{m,t} \mathbf{p}(\mathbf{s}_t) - \boldsymbol{\lambda}_{m,t}^T \mathbf{h}_{m,t} \\ &- \boldsymbol{\mu}_{m,t}^T \mathbf{g} - z_{m,t} = d_{\text{safe}} \end{aligned} \tag{16}$$

where $\boldsymbol{\lambda}_{m,t} \in \mathbb{R}^{l_m}$, $\boldsymbol{\mu}_{m,t} \in \mathbb{R}^l$, and $z_{m,t} \in \mathbb{R}$ are the dual variables representing our *attentions* on different surfaces of robots and obstacles. Putting (16) into problem (13), the resultant problem is

$$\min_{\substack{\{\mathbf{s}_t, \mathbf{u}_t\} \\ \{\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}\}}} \sum_{t=0}^{H} \|\mathbf{s}_t - \mathbf{s}_t^{\diamond}\|^2 \tag{17a}$$

$$\text{s.t.} \quad \text{constraints (2b)} - \text{(2d)} \tag{17b}$$

$$\boldsymbol{\lambda}_{m,t} \succeq \mathbf{0}, \boldsymbol{\mu}_{m,t} \succeq \mathbf{0} \qquad \forall m, t \tag{17c}$$

$$\left\|{\mathbf{H}_{m,t}}^T \boldsymbol{\lambda}_{m,t}\right\| \leq 1, z_{m,t} \geq 0 \qquad \forall m, t \tag{17d}$$

$$U_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}\right) = 0 \qquad \forall m, t \tag{17e}$$

$$V_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}, z_{m,t}\right) = 0 \qquad \forall m, t \tag{17f}$$

where we have defined

$$U_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}\right) = \boldsymbol{\mu}_{m,t}^T \mathbf{G} + \boldsymbol{\lambda}_{m,t}^T \mathbf{H}_{m,t} \mathbf{R}(\mathbf{s}_t) \tag{18}$$

$$V_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}, z_{m,t}\right) = \boldsymbol{\lambda}_{m,t}^T \mathbf{H}_{m,t} \mathbf{p}(\mathbf{s}_t) - \boldsymbol{\lambda}_{m,t}^T \mathbf{h}_{m,t} - \boldsymbol{\mu}_{m,t}^T \mathbf{g} - z_{m,t} - d_{\text{safe}}. \tag{19}$$

Now, instead of directly solving the dual problem using optimization software [3] or ADMM [8], we propose a PDD method (summarized in Algorithm 2) that constructs the following augmented Lagrangian:

$$\begin{aligned} \mathcal{L} = &\sum_{t=0}^{H} \|\mathbf{s}_t - \mathbf{s}_t^{\diamond}\|^2 + \frac{\rho}{2} \sum_{t=0}^{H} \sum_{m=1}^{M} \Big\| U_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}\right) \\ &+ \boldsymbol{\zeta}_{m,t} \Big\|^2 + \frac{\rho}{2} \sum_{t=0}^{H} \sum_{m=1}^{M} \left(V_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}, z_{m,t}\right) + \xi_{m,t}\right)^2 \end{aligned}$$

where $\{\boldsymbol{\zeta}_{m,t}, \xi_{m,t}\}$ are slack variables, ρ is the penalty parameter. Then, the augmented Lagrangian is minimized

$$\begin{aligned} \min_{\substack{\{\mathbf{s}_t, \mathbf{u}_t\} \in \mathcal{X}, \\ \{(\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}) \in \mathcal{Y}_{m,t}\}}} \quad & \mathcal{L}\left(\{\mathbf{s}_t, \mathbf{u}_t\}\right. \\ & \left.\{\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}\}; \rho, \{\boldsymbol{\zeta}_{m,t}, \xi_{m,t}\}\right) \end{aligned} \tag{20}$$

where $\mathcal{X}$ is the set for constraints (2b)–(2d), $\mathcal{Y}$ is the set for constraint (17c)–(17d). To solve problem (20), we first optimize the robot states and actions with all other variables fixed via CVXPY, as shown in line 4 of Algorithm 2. Next, we optimize the collision attentions with all other variables fixed via CVXPY as shown in line 5 of Algorithm 2. Finally, from lines 6–10, we either execute dual update or penalty update based on the residual function

$$\begin{aligned} &\Phi\left(\{\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}\}\right) \\ &= \max\left\{\max_{m,t} \left\| U_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}\right) \right\|_{\infty}\right. \\ &\quad \left.\max_{m,t} \left| V_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}, z_{m,t}\right) \right|\right\}. \end{aligned}$$

For dual update, we compute $\{\boldsymbol{\zeta}_{m,t}\}, \{\xi_{m,t}\}$ using first-order method as line 7 in Algorithm 2. For penalty update, we scale the penalty parameter as line 9 in Algorithm 2 with $\beta \geq 1$ being an increasing factor. According to [17, Theorem 3.1], Algorithm 2 is guaranteed to converge to a KKT solution to problem (20). The complexity of PDD is given by $\text{Comp}_k = \mathcal{O}((5H)^{3.5} + H\sum_{m\in\mathcal{M}(k)}(l_m + l_k)^{3.5} + H|\mathcal{M}(k)|(l_m + l_k))$.

Remark: For practical implementation, a static safety distance d_{safe} may not be suitable for all the time slots $t \in [0, H]$ since the state evolution starting from a feasible state may end up at a state that the safety constraint (2e) conflicts with the control bounds (2c) and (2d) [26], [27]. To this end, the fixed d_{safe} is modified to be a variable $d_{\min} \leq d_t \leq d_{\max}$ ($d_{\min}$ and $d_{\max}$ are the lower and upper bounds for d_t, respectively), which is encouraged toward a larger value by adding a norm distance regularizer $P(\{d_t\}) = -\eta\sum_{t=0}^{H}|d_t|$ (η is a weighting factor) to the objective function (2a). This method automatically

Algorithm 2: PDD Motion Planner.

Input: robot's state $\{\mathbf{s}_t\}$, obstacles' states $\{\mathbf{o}_m\}_{m=1}^{M}$
Initialize: robot's action $\{\mathbf{u}_t\}$, attentions $\{\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}\}$, and parameters $\{\beta, \rho, \eta_{iter}\}$
for *iteration* $iter = 1, 2, \cdots$ **do**
  $\{\mathbf{s}_t, \mathbf{u}_t\} \leftarrow \text{argmin}_{\{\mathbf{s}_t, \mathbf{u}_t\} \in \mathcal{X}} \mathcal{L}$
  $(\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}) \leftarrow \text{argmin}_{(\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}) \in \mathcal{Y}_{m,t}} \mathcal{L}, \ \forall m, t$
  if $\Phi\left(\{\boldsymbol{\lambda}_{m,t}, \boldsymbol{\mu}_{m,t}, z_{m,t}\}\right) \leq \eta_{iter}$ **then**
    Update $\{\boldsymbol{\zeta}_{m,t}, \xi_{m,t}\}$ using $\boldsymbol{\zeta}_{m,t} + U_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}\right)$, $\xi_{m,t} + V_{m,t}\left(\mathbf{s}_t, \boldsymbol{\mu}_{m,t}, \boldsymbol{\lambda}_{m,t}, z_{m,t}\right)$
  else
    Update $\rho \leftarrow \beta\rho$
  end
end
Output: $f_E(\mathbf{s}_t, \{\mathbf{o}_{m,t}\}_{m=1}^{M}) = \mathbf{u}_t$.

mitigates of conflicts between collision avoidance and control bound constraints.

V. Experiments

A. Implementation

We implemented the proposed EARN system using Python in ROS. The high-fidelity CARLA simulation platform [28] is used for evaluations, which adopts unreal engine for high-performance rendering. Our EARN system is connected to CARLA via ROS bridge [29] and data sharing is achieved via ROS communications, where the nodes publish or subscribe ROS topics that carry the sensory, state, or action information. We simulate two outdoor scenarios and one indoor scenario. All simulations are implemented on a Ubuntu workstation with a 3.7 GHZ AMD Ryzen 9 5900X CPU and an NVIDIA 3090 Ti GPU.

We also implement EARN in a real-world multirobot platform, where each robot has four wheels and can adopt Ackermann or differential steering. The robot named LIMO has a 2-D lidar, an RGBD camera, and an onboard NVIDIA Jetson Nano computing platform for executing the SLAM and local motion planning packages. The edge server is a manually controlled wheel-legged robot, Direct Drive Tech (DDT) Diablo, which has a 3-D livox lidar and an onboard NVIDIA Orin NX computing platform for executing the SLAM and edge motion planning packages. The Diablo is also equipped with a wireless access point.

To obtain the parameters involved in model (6), we execute the PDD motion planning in Algorithm 2 on the AMD Ryzen 9 and NVIDIA Orin NX chips. The experimental data of computation time (ms) versus the number of prediction horizons H and the number of obstacles $|\mathcal{M}(k)|$ is shown in Fig. 3. It can be seen that the computation time of PDD scales linearly with H and $|\mathcal{M}_k|$, which corroborates the complexity analysis. As such, the value of parameter p is set to 1. The computation time ranges from less than 10 to 200 ms, corresponding to a planning frequency

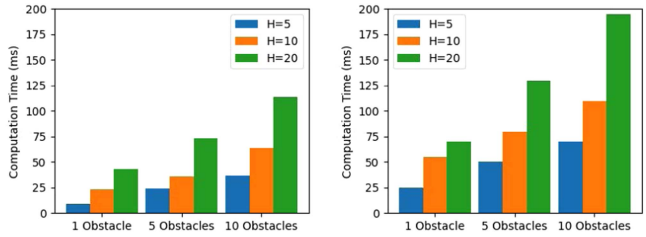


Fig. 3. Computation time (ms) of PDD planner versus the number of prediction horizons H and obstacles $|\mathcal{M}(k)|$. (a) AMD Ryzen 9. (b) NVIDIA Orin NX.

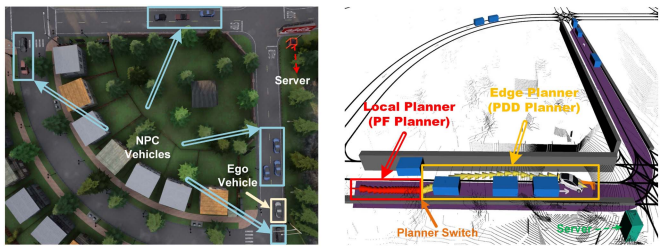


Fig. 4. Crossroad scenario in CARLA Town04 and planner switching from PF to PDD performed by EARN. (a) Crossroad scenario in CARLA Town04 map. (b) EARN switches from PF to PDD for overtaking.

of 5 Hz to over 100 Hz. The parameters γ, τ are obtained by fitting the function $C_k = \gamma H|\mathcal{M}_k| + \tau$ to the experimental data in Fig. 3 using weighted least squares, and is given by $\gamma^* = 0.6$ and $\tau^* = 12$ ms for ADM Ryzen 9 and $\gamma* = 1$ and $\tau^* = 20$ ms for NVIDIA Orin NX.

B. Benchmarks

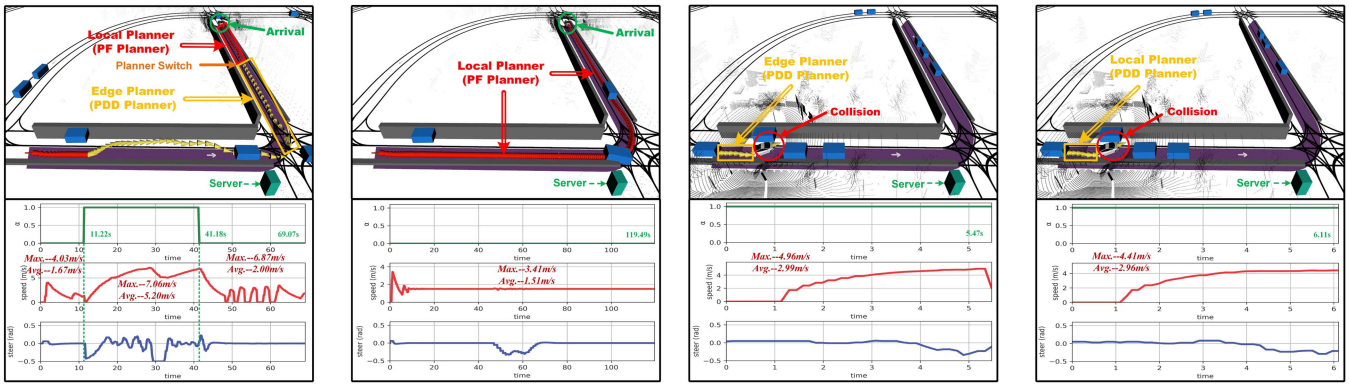
We compare our method to the following baselines.

1) PF planner [4].
2) Collision avoidance MPC (CAMPC) [25], which models each obstacle as a point and determines the collision condition by computing $\mathbf{dist}_c$.
3) RDA planner [8], which solves (17) in parallel via ADMM.
4) PDD-Edge, or PDD-E for short, which executes PDD planner at the edge server following the idea of [30].
5) PDD-Local, or PDD-L for short, which executes PDD planner at the local robot.
6) EDF, which is a CMP scheme where robots are opportunistically selected for edge computing using a nonpreemptive earliest deadline first (EDF) policy [18].[2]

C. Single-Robot Simulation

We first evaluate EARN in single-robot outdoor scenarios. The server is assumed to be equipped with AMD Ryzen 9. The robot is assumed to be a low-cost logistic vehicle with its

[2]Note that Atik et al. [18] do not consider motion planning and we combine our PDD planner with EDF for fair comparison.



(a) (b) (c) (d)

Fig. 5. Trajectories and control parameters of different schemes in crossroad scenario. Trajectories generated by the local and edge motion planners are marked in red and yellow, respectively. (a) EARN. (b) PF. (c) PDD-Edge. (d) PDD-Local.

longitudinal wheelbase and lateral wheelbase being 2.87 and 1.75 m, respectively. Based on these parameters, $(\mathbf{A}_{k,t}, \mathbf{B}_{k,t}, \mathbf{c}_{k,t})$ in (1) can be calculated. The braking distance for planner f_L is set to $d_{\mathrm{B}} = 8.0$ m and the safe distance for planner f_E is set to $d_{\mathrm{safe}} = 1$ m. The length of prediction horizon is set to $H = 5$, with the time step between consecutive motion planning frames being 0.35 s.

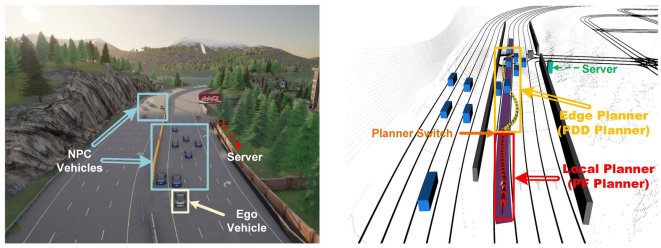
We consider a crossroad scenario in CARLA Town04 map, where the associated location of edge server, the starting/goal position of robot (with a random deviation of 3 m), and the positions of nonplayer character (NPC) vehicles are shown in Fig. 4(a). In this scenario, the maximum number of NPC vehicles in the local map $\mathcal{M}$ is 5. Using $H = 5$ and $|\mathcal{M}| = 5$, we have $C = 27$ ms when PDD is executed at the edge server. The computation time is assumed to be 200 ms when executed locally at the ego vehicle. The computation time of RDA is similar to that of PDD. The computation time of PF is less than 20 ms. The computation time of CAMPC is approximately $1/4$ of that of PDD, as CAMPC ignores the number of edges for each obstacle. The computation latency threshold is set to $C_{\mathrm{th}} = 50$ ms in Algorithm 1 according to the operational speed in the outdoor scenario.

The trajectories and control parameters generated by EARN, PF, PDD-E, and PDD-L are illustrated in Fig. 5(a)–(d). It is observed that both EARN and PF planner can navigate the robot to the destination without any collision, while PDD-E and PDD-L lead to collisions at $t = 5.47$ s and $t = 6.11$ s, respectively. Moreover, EARN executes the planner switching at $t = 11.22$ s as observed in Fig. 5(a). This empowers the robot with the capability to overtake front low-speed obstacles and increase the maximum speed from 4.03 to 7.06 m/s and the average speed from 1.67 to 5.20 m/s, by leveraging edge motion planning through low-latency communication access. In contrast, the maximum speed and average speed achieved by the PF planner are merely 3.41 and 1.51 m/s, respectively.

The quantitative comparisons are presented in Table II, where the performance of each method is obtained by averaging 50 trials. It can be seen that EARN reduces the average navigation time by 46.7% compared to PF, as EARN can switch the planner

TABLE II
COMPARISON OF NAVIGATION TIME AND SUCCESS RATE

Metric \ method	EARN	PF	PDD-E	PDD-L
Avg navigation time (s)	64.23	120.54	49.24	64.75
Success rate (%)	98%	100%	64%	58%



(a) (b)

Fig. 6. Dense traffic scenario in CARLA Town04 and overtaking performed by PDD planner. (a) Dense traffic scenario. (b) Overtaking by PDD planner.

adaptively and utilize the computing resources at the server, as shown in Fig. 4(b). Moreover, both PDD-E and PDD-L suffer from low success rate.[3] This is because PDD-E and PDD-L involve either computation or communication latency, resulting in low end-to-end planning frequency. The proposed EARN achieves a success rate significantly higher than those of PDD-E and PDD-L (98% versus 64% and 58%), which demonstrates the necessity of planner switching performed by EARN.

We also consider a dense traffic scenario in CARLA Town04 map as shown in Fig. 6(a) with tens of dynamic obstacles. This experiment is used to evaluate the robustness of EARN in dynamic and complex environments. The trajectories and control vectors generated by PDD, PF, RDA, and CAMPC are illustrated

[3]A successful navigation requires the robot to reach the goal without any collision.

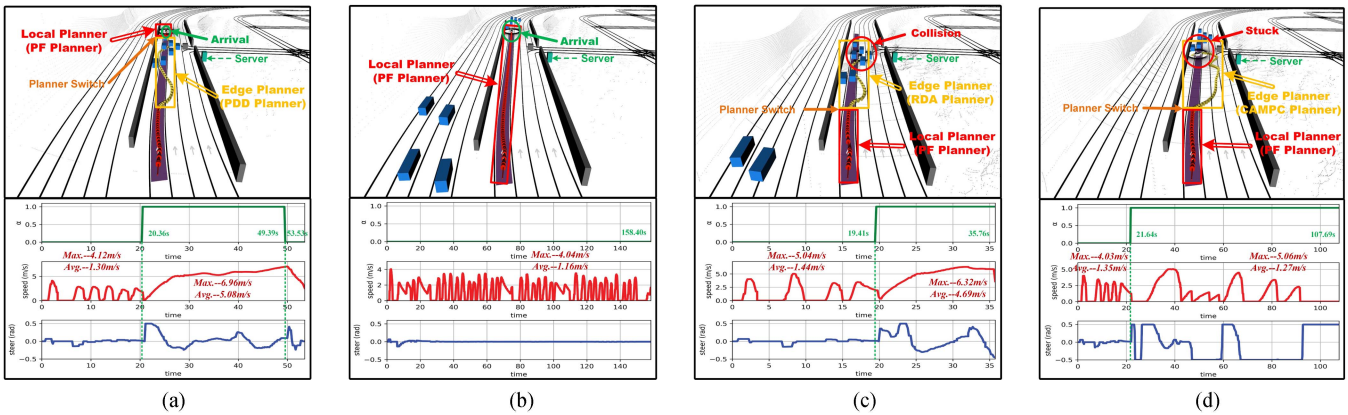


Fig. 7. Trajectories and control parameters of different schemes in dense traffic scenario. (a) PDD. (b) PF. (c) RDA. (d) CAMPC.

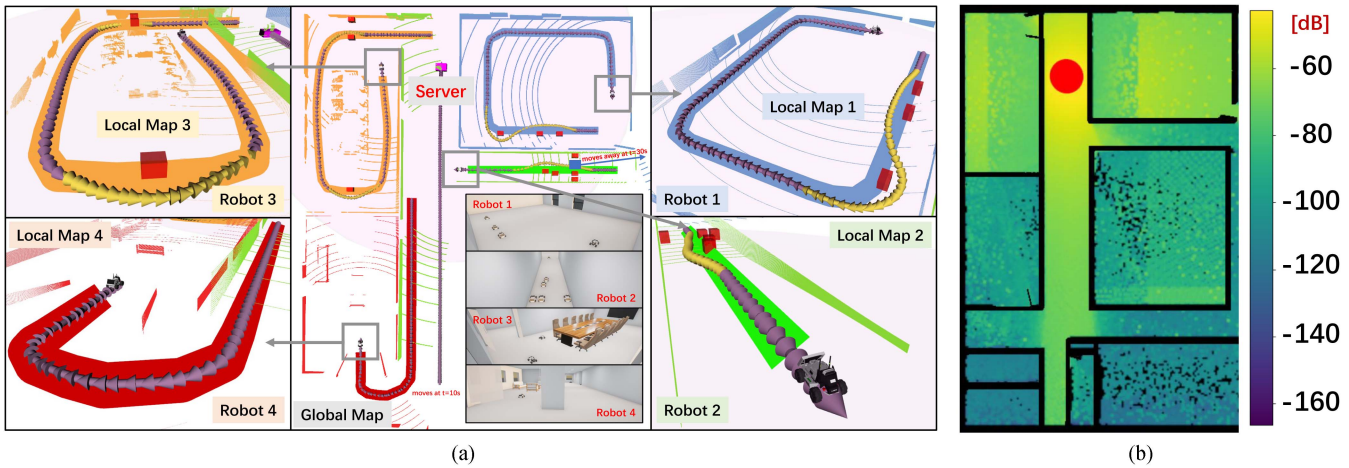
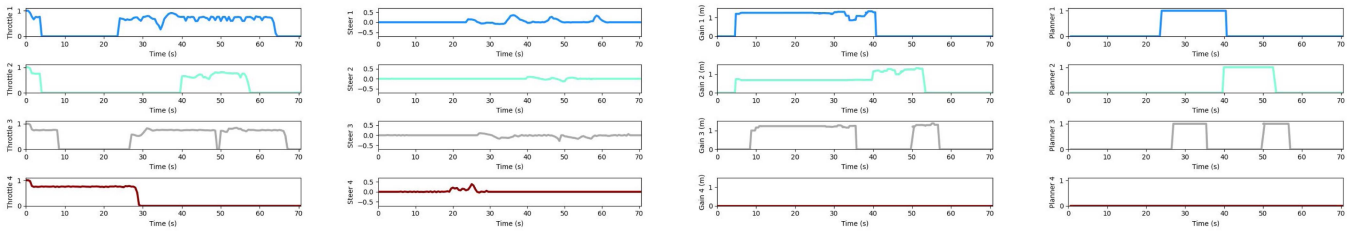


Fig. 8. Simulation results of the multirobot indoor scenario. (a) Trajectories generated by the local (marked in purple) and edge (marked in yellow) motion planners. (b) Radio map of the server.

in Fig. 7(a)–(d). In this scenario, the PDD planner successfully navigates the robot to the destination in merely 53.53 s with average speed 5.08 m/s, as shown in Fig. 7(a). In contrast, the PF planner, while also finishing the task, costs over 158.40 s (with speed from 0 to 4.04 m/s) due to its conservative navigation strategy. Moreover, the robot with RDA planner collides with NPC vehicle as the collision avoidance constraint is not strictly satisfied due to the fixed value of ρ. The CAMPC planner is neither able to navigate the robot to the destination, i.e., the robot gets stuck behind the dense traffic flow due to the center point distance model. Since RDA and CAMPC planners cannot accomplish the navigation task, we only compare the navigation time of PDD and PF. It is found that the PDD planner reduces the navigation time by 66.2% compared to the PF planner, while achieving the same success rate.

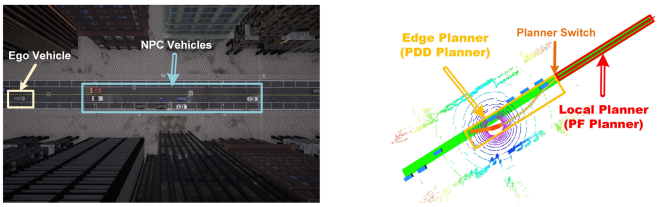
D. Multirobot Simulation

To verify the effectiveness of Algorithm 1, we implement EARN in a multirobot indoor scenario shown in Fig. 8(a), where robot 1 navigates in an office room with its target path marked in blue, robot 2 navigates in a corridor with its target path marked in green, robot 3 navigates in a conference room with its target path marked in orange, and robot 4 navigates from a corridor to a lounge with its target path marked in red. The number of NPC robots, which are marked as red boxes, in local maps 1–4 are are $(|\mathcal{M}_1|, |\mathcal{M}_2|, |\mathcal{M}_3|, |\mathcal{M}_4|) = (3, 6, 3, 0)$, respectively. The server (marked as a pink box) is assumed to be a Diablo wheel-legged robot with NVIDIA Orin NX. Each robot is simulated as an Ackerman steering car-like robot with their length, width, longitudinal wheelbase, and lateral wheelbase being 32.2 , 22.0 , 20.0 , and 17.5 cm, respectively, and the parameters $(\mathbf{A}_{k,t}, \mathbf{B}_{k,t}, \mathbf{c}_{k,t})$ in (1) are computed accordingly. The braking distance for planner f_L is set to $d_{\mathrm{B}} = 1.3$ m and the safe distance for planner f_E is set to $d_{\mathrm{safe}} = 0.1$ m. The length of prediction horizon is set to $H = 20$, with the time step between consecutive motion planning frames being 0.25 s. Using $H = 20$ and the above $\{|\mathcal{M}_k|\}$, we have $(C_1, C_2, C_3, C_4) = (80, 140, 80, 20)$ ms. The total computation latency threshold is set to $C_{\mathrm{th}} = 240$ ms according to the operational speed in indoor environments. The execution time of PDD is assumed to be unacceptable when executed locally at the robot.



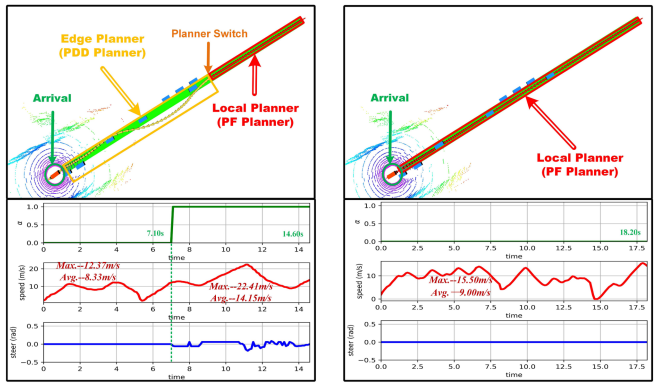
(a) (b) (c) (d)

Fig. 9. The throttle commands, steer commands, switching gains, and planner selections of different robots. (a) Throttle commands. (b) Steer commands. (c) Switching gains. (d) Planner selections.



(a) (b)

Fig. 10. Urban driving scenario in Intel CARLA challenge. (a) Urban driving scenario. (b) Planner switched by EARN.



(a) (b)

Fig. 11. Trajectories and control parameters of different schemes in the official Intel CARLA challenge. (a) EARN. (b) PF.

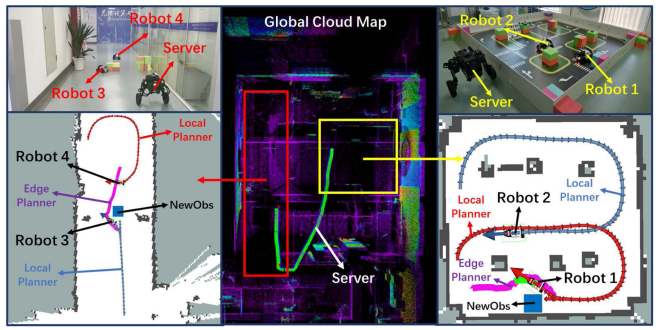


Fig. 12. Real-world experiment 1.

The trajectories of four robots under the proposed EARN are shown in Fig. 8(a), where the purple trajectories are generated by local motion planner and the yellow trajectories are generated by edge motion planner. The associated throttle commands, steer commands, switching gains $\{I_k\}$, and planner selections $\{\alpha_k\}$ are shown in Fig. 9(a)–(d). In particular, starting from $t = 0$ s, all robots adopt local motion planning and move at a speed of 0.55 m/s. Robot 4 encounters no obstacles along its target path; hence it keeps on moving using the local motion planner [as seen from Fig. 9(c)–(d)] until reaching the goal at $t = 29.64$ s [as seen from Fig. 9(a)–(b)]. In contrast, robots 1–3 stop in front of obstacles at $t = 4.08$, 3.88 , and 8.52 s, respectively, as observed from Fig. 9(a)–(b). Consequently, the switching gains of robots 1 and 3 increase from 0 to 1.2. However, they cannot switch their planners immediately, since the edge server is now at the other side of the map and the communication latency between the server and robot 1 or 3 would be large. Note that the switching gain of robot 2 is only 0.67, since the corridor is blocked by 4 obstacles and the robot cannot pass the "traffic jam" even if it switches from local to edge planning.

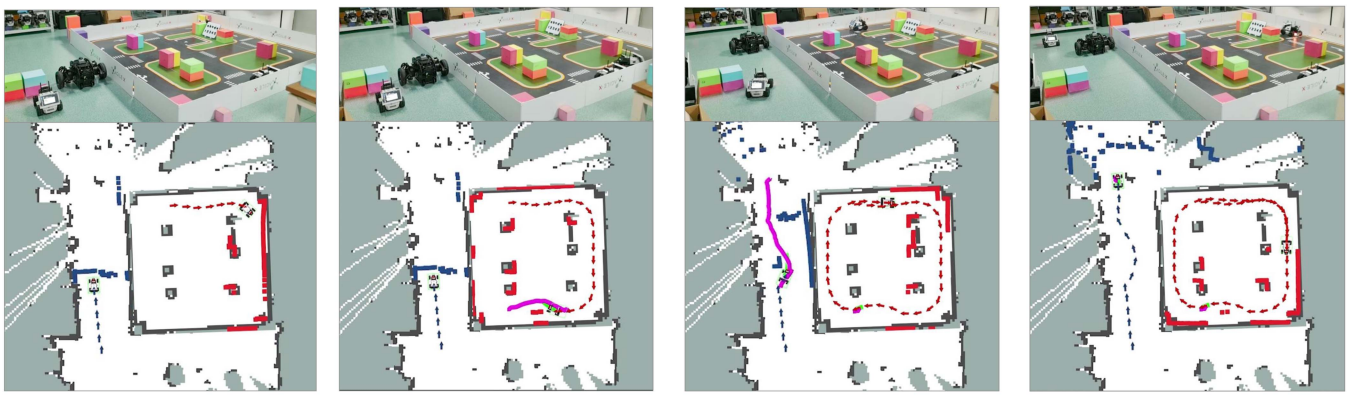
At $t = 10$ s, the server starts to move upwards at a speed of 0.88 m/s, and collaborates with robots 1 and 3 for collision avoidance using edge motion planning with a speed of 0.67 m/s at $t = 24.12$ s and $t = 27.12$ s, respectively. After completing the collision avoidance actions, the switching gains of robots 1 and 3 become 0 and the planners are switched back to local ones to save computation resources at the server. Furthermore, the switching gain of robot 2 increases from 0.67 to 1.2 at $t = 40.08$ s. This is because an NPC robot (marked as a blue box) in the corridor moves away at $t = 30$ s, leaving enough space for robot 2 to pass the traffic jam.

The average navigation time of EARN and EDF is provided in Table III. The navigation time of robots 1–4 with EARN is 64.80 , 57.96 , 67.68 , and 29.64 s, respectively. The navigation time of robots 1–4 with EDF is 65.89 , 56.40 , 95.16 , and 29.29 s, respectively. Compared to EDF, the total navigation time of EARN is reduced by 12.1%. This is because the EDF scheme is based on the expected task execution deadline, not on the low-level motion planning trajectories. As such, EDF fails to recognize the scenarios and traffic in real-time, leading to a potential resource-robot mismatch. For instance, in the considered indoor scenario, robots 2 and 4 are expected to finish their navigation tasks in a shorter time, since their target paths

TABLE III
COMPARISON OF THE ROBOT NAVIGATION TIME UNDER DIFFERENT COMPUTATION TIME CONSTRAINTS

Method	C_{th} (ms)	Num	Navigation time (s) ↓				
			Robot 1	Robot 2	Robot 3	Robot 4	Total
EARN (ours)	240	Four robots	**64.80 (−1.5%)**	57.96	**67.68 (−28.87%)**	29.29	**219.73 (−10.94%)**
EDF [18]	240	Four robots	65.89	**56.40 (-2.69%)**	95.16	29.29	246.74
EARN (ours)	160	Four robots	**64.56 (−32.75%)**	66.72	**80.04 (−40.81%)**	29.29	**240.61 (−23.96%)**
EDF [18]	160	Four robots	96.00	**55.92 (-16.18%)**	135.24	29.29	316.45

The bold value here represents the superior performance between EARN and EDF.



(a) (b) (c) (d)

Fig. 13. Real-world experiment 2. (a) No switching gain. (b) Robot 1 switches. (c) Robot 2 switches. (d) Task accomplished.

are shorter than other robots as seen from Fig. 8(a). Therefore, EDF selects robots 2 and 4 for edge motion planning, despite the fact that robot 2 gets stuck in the traffic jam and robot 4 has a small switching gain from the local to edge planner. The navigation time of EARN and EDF under $C_{\text{th}} = 160$ ms is also provided in Table III. Due to this tighter time constraint, the server cannot navigate multiple robots at the same time, and the waiting time for robots 1 and 3 becomes larger. Consequently, the robot navigation time increases. However, the proposed EARN still outperforms EDF by a large margin.

E. Intel CARLA Challenge

The performance of EARN is also verified in the official Intel CARLA Challenge.[4] As illustrated in Fig. 10(a), the competition is an urban driving scenario of the CARLA Town12 map, with various NPC vehicles on the road. It is observed from Fig. 10(b) that the robot accomplishes the overtaking action and reduces the navigation time by switching the adopted planner to PDD at $t = 7.10$ s. The trajectories and control parameters generated by EARN and PF are also illustrated in Fig. 11(a)–(b). It is observed that both EARN and PF planners are able to navigate the robot to the destination without any collision. However, the navigation time of EARN (14.60 s) is reduced by 19.78% compared to that of the PF planner (18.20 s), which reveals the speed boost and performance enhancement brought by EARN.

[4][Online]. Available: https://leaderboard.carla.org/challenge/

F. Real-World Testbed

Finally, to verify the hardware–software compatibility of EARN and its robustness against sensor and actuator uncertainties, we implement EARN in a real-world testbed consisting of a Diablo wheel-legged robot server and 4 LIMO car-like robots. As shown in Fig. 12, we adopt the livox lidar and the fast-lio algorithm [31] to obtain a cm-level localization of the Diablo robot and the global cloud map of the indoor scenario. The trajectory of Diablo is marked as red–green–blue axes. The cloud map consists of two regions, where robots 1 and 2 navigate inside the sandbox (marked by a yellow box) and robots 3 and 4 navigate in the corridor (marked by a red box). All robots need to avoid collisions with the static environments as well as the cubes therein.

The trajectories of four robots under the proposed EARN are shown at the left and right sides of Fig. 12, where the blue and red arrows are generated by local motion planners and the pink paths are generated by edge motion planners. It can be seen that robots 2 and 4 encounter no obstacles along their target path, and they execute the local motion planner without external assistance until reaching the goal at $t = 25.45$ s and $t = 21.45$ s, respectively. On the other hand, robots 1 and 3 encounter new obstacles (marked in blue boxes) and leverage planner switching in front of the blue boxes. Specifically, robot 1 and Diablo server collaboratively accomplish reverse and overtaking actions, so that the robot reaches the goal at 26.00 s. Robot 3 holds its position at the beginning and waits for connection with the server. At about $t = 30$ s, Diablo moves from the sandbox to

the corridor and assists robot 3 for switching to edge motion planning at $t = 64$ s. With another 13.85 s, robot 3 reaches the goal. This demonstrates the impact of communication on EARN.

To validate the effectiveness of switching gain, real experiment 2 is conducted with two LIMO robots and one Diablo server, as depicted in Fig. 13, where robot 1 navigates inside the sandbox, while robot 2 navigates outside the sandbox. As illustrated in Fig. 13(a)–(b), although the robot 2 is closer to the server, its trajectory is planned by the local planner rather than the edge planner. This is because its path is completely blocked by the server, and its switching gain is zero. Consequently, conducting planner switching for this robot leads to a waste of resources. In contrast, planner switching is adopted for robot 1, as shown in Fig. 13(b), since the local planner gets stuck in front of a box but the edge planner is able to overtake, resulting in enhancement of planning efficiency. Furthermore, as illustrated in Fig. 13(c), with the movement of server, planner switching also occurs for robot 1 due to the increased switching gain. This demonstrates the real-time adaptiveness of EARN. Finally, the task is successfully completed through collaboration between the server and the robots, as depicted in Fig. 13(d). Please refer to our video for more details.

VI. Conclusion

This article proposed EARN to realize CMP and MPS, thereby opportunistically accelerating the trajectory computations while guaranteeing safety. The new feature of EARN is to optimize robot states and actions under communication and computation resource constraints. CARLA simulations and real-world experiments have shown that EARN reduces the navigation time by 12.1% and 46.7% compared with EDF and PF schemes in indoor and outdoor scenarios, respectively. Furthermore, EARN increases the success rate by 53.1% and 69.0% compared with edge or local computing schemes.

References

[1] K. M. Lynch and F. C. Park, *Modern Robotics*. New York, NY, USA: Cambridge Univ. Press, 2017.

[2] 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.

[3] 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.

[4] Z. Wang, X. Zhou, and J. Wang, "Extremum-seeking-based adaptive model-free control and its application to automated vehicle path tracking," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 5, pp. 3874–3884, Oct. 2022.

[5] Z. Han et al., "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Trans. Intell. Transp. Syst.*, vol. 25, no. 2, pp. 1797–1814, Feb. 2024.

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

[7] 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. Int. Conf. Intell. Transp. Syst.*, 2018, pp. 1633–1639.

[8] R. Han et al., "RDA: An accelerated collision-free motion planner for autonomous navigation in cluttered environments," *IEEE Robot. Automat. Lett.*, vol. 8, no. 3, pp. 1715–1722, Mar. 2023.

[9] Z. Huang, S. Shen, and J. Ma, "Decentralized iLQR for cooperative trajectory planning of connected autonomous vehicles via dual consensus ADMM," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 11, pp. 12754–12766, Nov. 2023.

[10] Y. Xiao, F. Codevilla, A. Gurram, O. Urfalioglu, and A. M. López, "Multimodal end-to-end autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 1, pp. 537–547, Jan. 2022.

[11] Y. Pan et al., "Imitation learning for agile autonomous driving," *Int. J. Robot. Res.*, vol. 39, no. 2/3, pp. 286–302, Mar. 2020.

[12] W.-B. Kou et al., "Communication resources constrained hierarchical federated learning for end-to-end autonomous driving," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2023, pp. 9383–9390.

[13] K. Goldberg, "Robots and the return to collaborative intelligence," *Nature Mach. Intell.*, vol. 1, pp. 2–4, Jan. 2019.

[14] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg, "RILaaS: Robot inference and learning as a service," *IEEE Robot. Automat. Lett.*, vol. 5, no. 3, pp. 4423–4430, Jul. 2020.

[15] S. Hayat, R. Jung, H. Hellwagner, C. Bettstetter, D. Emini, and D. Schnieders, "Edge computing in 5G for drone navigation: What to offload?," *IEEE Robot. Automat. Lett.*, vol. 6, no. 2, pp. 2571–2578, Apr. 2021.

[16] D. Liu, Z. Lyu, Q. Zou, X. Bian, M. Cong, and Y. Du, "Robotic navigation based on experiences and predictive map inspired by spatial cognition," *IEEE/ASME Trans. Mechatron.*, vol. 27, no. 6, pp. 4316–4326, Dec. 2022.

[17] Q. Shi and M. Hong, "Penalty dual decomposition method for nonsmooth nonconvex optimization—Part I: Algorithms and convergence analysis," *IEEE Trans. Signal Process.*, vol. 68, pp. 4108–4122, 2020.

[18] S. T. Atik, M. Brocanelli, and D. Grosu, "Are turn-by-turn navigation systems of regular vehicles ready for edge-assisted autonomous vehicles?," *IEEE Trans. Intell. Transp. Syst.*, vol. 24, no. 10, pp. 11146–11156, Oct. 2023.

[19] P. Huang, L. Zeng, X. Chen, K. Luo, Z. Zhou, and S. Yu, "Edge robotics: Edge-computing-accelerated multi-robot simultaneous localization and mapping," *IEEE Internet Things J.*, vol. 9, no. 15, pp. 14087–14102, Aug. 2022.

[20] J. Xu et al., "SwarmMap: Scaling up real-time collaborative visual SLAM at the edge," in *Proc. USENIX Symp. Networked Syst. Des. Implementation*, 2022, pp. 977–993.

[21] J. Ichnowski et al., "FogROS2: An adaptive platform for cloud and fog robotics using ROS 2," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2023, pp. 5493–5500.

[22] B. Morrell et al., "NeBula: TEAM CoSTAR's robotic autonomy solution that won phase II of DARPA subterranean challenge," *Field Robot.*, vol. 2, pp. 1432–1506, 2022.

[23] K. Sui et al., "Characterizing and improving WiFi latency in large-scale operational networks," in *Proc. ACM Int. Conf. Mobile Syst., Appl., Serv.*, 2016, pp. 347–360.

[24] J. Hoydis et al., "Sionna: An open-source library for next-generation physical layer research," 2022, *arXiv:2203.11854*.

[25] S. Zhang, S. Wang, S. Yu, J. Yu, and M. Wen, "Collision avoidance predictive motion planning based on integrated perception and V2V communication," *IEEE Trans. Intell. Transp. Syst.*, vol. 23, no. 7, pp. 9640–9653, Jul. 2022.

[26] W. Xiao and C. Belta, "High-order control barrier functions," *IEEE Trans. Autom. Control*, vol. 67, no. 7, pp. 3655–3662, Jul. 2022.

[27] W. Xiao, C. A. Belta, and C. G. Cassandras, "Sufficient conditions for feasibility of optimal control problems using control barrier functions," *Automatica*, vol. 135, 2022, Art. no. 109960.

[28] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator," in *Proc. Conf. Robot Learn.*, 2017, pp. 1–16.

[29] Carla-simulator, 2022. [Online]. Available: https://github.com/carla-simulator/ros-bridge

[30] J. Wehbeh, S. Rahman, and I. Sharf, "Distributed model predictive control for UAVs collaborative payload transport," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 11666–11672.

[31] W. Xu, Y. Cai, D. He, J. Lin, and F. Zhang, "FAST-LIO2: Fast direct lidar-inertial odometry," *IEEE Trans. Robot.*, vol. 38, no. 4, pp. 2053–2073, Aug. 2022.

Guoliang Li (Graduate Student Member, IEEE) received the B.Eng. and M.Phil. (*summa cum laude*) degrees in electrical and electronic engineering from the Southern University of Science and Technology, Shenzhen, China, in 2020 and 2023, respectively. He is currently working toward the Ph.D. degree in computer science with the University of Macau, Macau, China.

His research interests include optimization, motion planning, and autonomous systems.

Dr. Li was the recipient of the National Scholarship in 2022 and the IEEE ICDCS Workshop on SocialMeta Best Student Paper Award in 2023.

Ruihua Han (Member, IEEE) received the B.Eng. degree in industrial equipment and control engineering from the Wuhan University of Technology, Wuhan, China, and the M.S. degree in microelectronics and solid state electronics from Xiamen University, Xiamen, China. He is currently working toward the Ph.D. degree in computer science with The University of Hong Kong, Hong Kong.

His research interests include optimization, motion planning, reinforcement learning, and autonomous robots.

Shuai Wang (Senior Member, IEEE) received the Ph.D. degree in electrical and electronic engineering from the University of Hong Kong, Hong Kong, in 2018.

He is currently an Associate Professor with the Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, Beijing, China, where he leads the Intelligent Networked Vehicle Systems Laboratory. He has authored or coauthored more than 90 journal and conference papers. His research interests include autonomous systems and wireless communications.

Dr. Wang was the recipient of three Best Paper Awards from IEEE.

Fei Gao (Member, IEEE) received the Ph.D. degree in electronic and computer engineering from the Hong Kong University of Science and Technology, Hong Kong, in 2019.

He is currently a Tenured Associate Professor with the Department of Control Science and Engineering, Zhejiang University, Hangzhou, China, where he leads the Flying Autonomous Robotics (FAR) group affiliated with the Field Autonomous System and Computing Laboratory. His research interests include aerial robots, autonomous navigation, motion planning, optimization, and localization and mapping.

Yonina C. Eldar (Fellow, IEEE) received the B.Sc. degree in physics and the B.Sc. degree in electrical engineering from Tel-Aviv University, Tel-Aviv, Israel, and the Ph.D. degree in electrical engineering and computer science from the Massachusetts Institute of Technology (MIT), Cambridge, MA, USA, in 2002.

She is currently a Professor with the Department of Mathematics and Computer Science, Weizmann Institute of Science, Rehovot, Israel, where she heads the center for Biomedical Engineering and Signal Processing and holds the Dorothy and Patrick Gorman Professorial Chair. She is also a Visiting Professor with MIT, a Visiting Scientist with Broad Institute, and an Adjunct Professor with Duke University, Durham, NC, USA, and was a Visiting Professor at Stanford.

Dr. Eldar was the recipient of many awards for excellence in research and teaching, including the IEEE Signal Processing Society Technical Achievement Award (2013), the IEEE/AESS Fred Nathanson Memorial Radar Award (2014), the IEEE Kiyo Tomiyasu Award (2016), the Michael Bruno Memorial Award from the Rothschild Foundation, the Weizmann Prize for Exact Sciences, the Wolf Foundation Krill Prize for Excellence in Scientific Research, the Henry Taub Prize for Excellence in Research (twice), the Hershel Rich Innovation Award (three times), and the Award for Women with Distinguished Contributions. She received several best paper awards and best demo awards together with her research students and colleagues, was selected as one of the 50 most influential women in Israel, and was a Member of the Israel Committee for Higher Education. She is the Editor in Chief of Foundations and Trends in Signal Processing, a Member of several IEEE Technical Committees and Award Committees, and heads the Committee for Promoting Gender Fairness in Higher Education Institutions in Israel. She is a Member of the Israel Academy of Sciences and Humanities, and the EURASIP Fellow. She was a Horev Fellow of the Leaders in Science and Technology program at the Technion and an Alon Fellow.

Chengzhong Xu (Fellow, IEEE) received the Ph.D. degree in computer science and engineering from the University of Hong Kong, Hong Kong, in 1993.

He is currently a Chair Professor of computer science with the University of Macau, Macau, China. Prior to that, he was in the faculty of Wayne State University, Detroit, MI, USA, and the Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, Beijing, China. He has authored or coauthored two research monographs and more than 500 journal and conference papers and received more than 17 000 citations. He was also a coinventor of more than 150 patents and a co-founder of Shenzhen Institute of Baidou Applied Technology. His research interests are in cloud and distributed computing, intelligent transportation and autonomous driving.

Dr. Xu was the Chair of IEEE Technical Committee on Distributed Processing from 2015 to 2020. He was a recipient of the Faculty Research Award, Career Development Chair Award, and the President's Award for Excellence in Teaching of WSU, and the "Outstanding Oversea Young Scholar" award of NSFC in 2010. He serves or served on a number of journal editorial boards, including IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON CLOUD COMPUTING, IEEE TRANSACTIONS ON PARALLEL AND DISTRIBUTED SYSTEMS, *Journal of Parallel and Distributed Computing*, *Science China*, and *ZTE Communication*. He was a best paper awardee or nominee of conferences, including HPCA'2013, HPDC'2013, ICPP'2015, and SoCC'2021.