2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2025)
October 19-25, 2025. Hangzhou, China

2025 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) | 979-8-3315-4393-8/25/$31.00 ©2025 IEEE | DOI: 10.1109/IROS60139.2025.11245999

Opportunistic Collaborative Planning with Large Vision Model Guided Control and Joint Query-Service Optimization

Jiayi Chen[1], Shuai Wang[2,†], Guoliang Li[3], Wei Xu[4], Guangxu Zhu[1,†],
Derrick Wing Kwan Ng[5], *Fellow, IEEE*, and Chengzhong Xu[3], *Fellow, IEEE*

***Abstract*— Navigating autonomous vehicles in open scenarios is a challenge due to the difficulties in handling unseen objects. Existing solutions either rely on small models that struggle with generalization or large models that are resource-intensive. While collaboration between the two offers a promising solution, the key challenge is deciding when and how to engage the large model. To address this issue, this paper proposes opportunistic collaborative planning (OCP), which seamlessly integrates efficient local models with powerful cloud models through two key innovations. First, we propose large vision model guided model predictive control (LVM-MPC), which leverages the cloud for LVM perception and decision making. The cloud output serves as a global guidance for a local MPC, thereby forming a closed-loop perception-to-control system. Second, to determine the best timing for large model query and service, we propose collaboration timing optimization (CTO), including object detection confidence thresholding (ODCT) and cloud forward simulation (CFS), to decide when to seek cloud assistance and when to offer cloud service. Extensive experiments show that the proposed OCP outperforms existing methods in terms of both navigation time and success rate.**

I. Introduction

While autonomous navigation technologies have seen rapid advancements, achieving high efficiency under safety guarantee remains a challenge, particularly when it comes to recognizing and reacting to unknown objects in open scenarios [1]. The presence of these objects often leads to perception errors propagating to the subsequent control, making ego-vehicle prone to take detour or get stuck [2].

Traditional navigation systems [3]–[5] rely on small local models that struggle to handle unseen objects. Emerging large models [6]–[8] are capable of recognizing a wide variety of objects, but are resource-intensive to operate at a high frequency. While collaboration between the small and large models is a promising direction, determining when and how to engage the cloud large model is non-trivial. Collaboration between large and small models must ensure feasible vehicle dynamics and collision avoidance, especially when using black-box large models. Vehicles should trigger cloud queries based on confidence thresholdsimproper settings may lead to excessive queries or overly conservative behavior. The cloud, in turn, must decide whether to respond by evaluating current and future benefits, as not all detections affect the navigation path. Existing approaches [9]–[15] overlook the coupling between low-level planning and high-level inference, as well as the joint optimization of query and service timing.

To fill the gap, this paper proposes opportunistic collaborative planning (OCP), which seamlessly integrates the efficient planning of local small models and the powerful planning of cloud large models, via two key innovations. The first innovation is to exploit the suitable multi-models edge-cloud collaboration. In particular, the cloud first executes large vision model (LVM) for perception and decision making. Its output is forwarded to the vehicle as guidance of decisions (e.g., lane change), paths (i.e., waypoints), and control styles (e.g., inflation distances), for reconfiguring the model predictive controller (MPC). This forms a closed-loop LVM-MPC collaboration from perception to control, which effectively handles rare objects in open scenario. The second innovation is the collaboration timing optimization (CTO), including both local assistance query and cloud inference service. Object detection confidence thresholding (ODCT) is proposed to decide when to seek cloud assistance on local vehicle. Furthermore, cloud forward simulation (CFS) is proposed to decide when to offer cloud service by considering the trajectory improvements and processing latency.

To evaluate OCP, it is necessary to adopt a high-fidelity simulator with intertwined execution of learning-based perception and optimization-based control. We thus implement OCP in the Car Learning to Act (Carla) platform [16] using robot operation system (ROS). Results in Carla-ROS demonstrate the superiority of the proposed OCP over existing RDA planner [3], and collaborative planner [9], in a multi-lane urban driving scenario. The success rate and navigation time are reduced by over 6% and 26%, respectively.

Our contributions are summarized as follows:

- Propose OCP to handle rare objects in open scenarios by integrating small local and large cloud models.
- Propose LVM-MPC, which is a closed-loop collaboration from perception to control.
- Propose CTO to determine the optimal timing for large model query and inference, respectively.
- Implement OCP and associated methods in Carla, and demonstrate its effectiveness in various scenarios.

This work was supported in part by National Natural Science Foundation of China (Grant No. 62371313, Grant No. 62371444), in part by ShenzhenHong Kong-Macau Technology Research Programme (Type C) (Grant No. SGDX20230821091559018), in part by the Shenzhen Science and Technology Program (Grant No. JCYJ20241202124934046), in part by Guangdong Young Talent Research Project (Grant No. 2023TQ07A708) and the Shenzhen Science and Technology Program (Grant No. RCYX20231211090206005).

[1]Jiayi Chen and Guangxu Zhu are with the Shenzhen Research Institute of Big Data, The Chinese University of Hong Kong (Shenzhen), Shenzhen 518115, China (jiayichen5@link.cuhk.edu.cn, gxzhu@sribd.cn). [2]Shuai Wang is with the Shenzhen Institutes of Advanced Technology (SIAT), Chinese Academy of Sciences, Shenzhen, China (s.wang@siat.ac.cn). [3]Guoliang Li and Chengzhong Xu are with the State Key Laboratory of Internet of Things for Smart City (SKL-IOTSC), University of Macau, Macau, China. [4]Wei Xu is with the Manifold Tech Limited, Hong Kong, China. [5]Derrick Wing Kwan Ng is with the School of Electrical Engineering and Telecommunications, the University of New South Wales, Australia. † denotes corresponding authors.

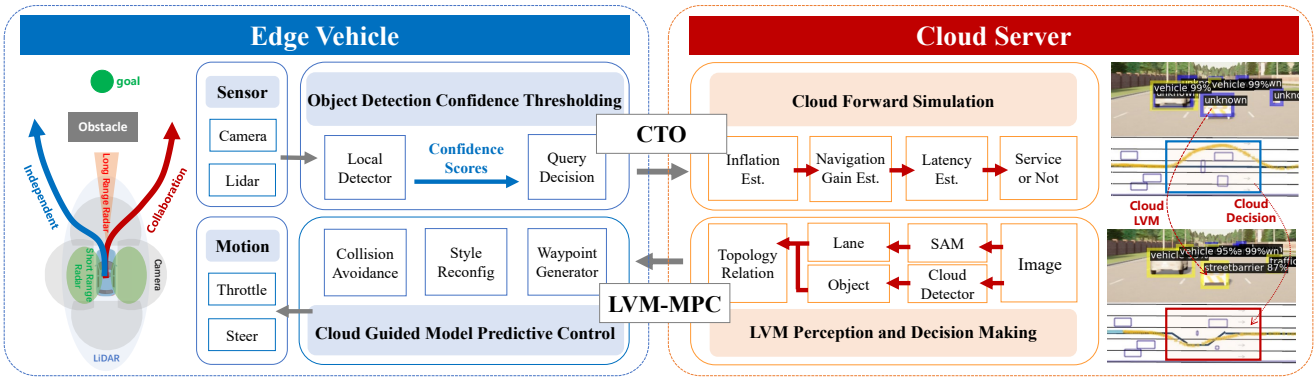


Fig. 1: System architecture of the proposed OCP, which consists of LVM-MPC and CTO blocks.

II. Related Work

Autonomous navigation involves perception, which provides necessary information about the environment, and planning, which computes a sequence of feasible actions to reach the goal [2]. Conventional navigation is based on small models [3], [4]. While easy-to-deploy, they suffer from error propagation [5], [10]: the detection results (e.g., voxels, bounding boxes) involves high uncertainties when handling unseen objects in open scenarios, leading to a larger safety distance in the planner [2]. Large models can be adopted to mitigate error propagation. For instance, in [6], large vision model is adopted as a backbone coupled with downstream perception head to understand street scene semantic information. In [7], AgentsCoDriver is proposed, which uses large language models to enhance generalization and realize lifelong learning through cognitive memory. In [8], DriveGPT4 is proposed, which employs multi-modal large models to process videos and captions, thereby effectively explaining vehicle actions. However, large models suffer from a lack of interpretability and the output trajectory may violate vehicle dynamics. Due to high computation requirements, direct deployment on vehicles is not economic and may lead to low execution frequency.

Recently, the collaborative planning paradigm emerges, which deploys small efficient models at the vehicle and large powerful models at the cloud [9], [11]. For instance, object recognition and grasp planning can be offloaded to cloud as a service [12]. In [13], the Language-MPC is proposed to understand natural language instructions and enhance decision-making through large language models while ensuring feasible vehicle dynamics through small MPC models. In DARPA SubT challenge [14], a collaborative planning framework is adopted to deploy the mission planner at the cloud for task assignment and the motion planner at each robot for task execution. To reduce the communication latency introduced by collaboration, a cloud robotics platform FogROS2 is proposed to effectively connect robot systems across different physical locations, networks, and data distribution services [15]. Nonetheless, current collaborative planning schemes ignore the collaboration timing optimization, which may lead to mismatch between large model service and small model requirement. In contrast, OCP optimizes the collaboration timing explicitly through ODCT and CFS.

III. Opportunistic Collaborative Planning

A. Overview of Architecture

The architecture of OCP is shown in Fig. 1. The input of OCP consists of the multi-modal sensor data (e.g., camera images, lidar point clouds) and the navigation goal position (i.e., marked as a green ball). The output consists of collaboration states (i.e., a sequence of binary decisions), collision-free trajectories (i.e., a sequence of vehicle states), and control actions (i.e., throttle and steer). The goal of OCP is to transform unknown objects into known ones by calling large models at the right timing, so as to improve the vehicle trajectory quality to the maximum extent, avoiding detouring or stuck cases as shown at the right hand side of Fig. 1.

To realize the above objective, the OCP system adopts an LVM-MPC block to decide how to collaborate, and a CTO block to decide when to collaborate. The LVM-MPC block executes LVM perception and decision making at the cloud for guiding small MPC model at the local vehicle. The CTO block consists of the ODCT and CFS modules, which decide when to seek cloud assistance and when to offer cloud service, respectively.

B. LVM-MPC and CTO

For LVM-MPC, it is based on the joint processing of cloud and vehicle models. At the cloud, the LVM adopts segment anything model (SAM) [17] for reperception, which recognizes obstacles, generates lanes, and associates obstacles with their corresponding lanes. With these information, the cloud further employs a decision tree for generating behavior decisions (i.e., left, right, keep) and reconfiguring control styles (i.e., inflation distances). These information, together with the calibrated objects and new confidence scores, is forwarded to the ego vehicle. The MPC module generates responsive control commands to ensure safe efficient navigation, by considering both real-time local conditions and strategic guidance from the cloud. Under the non-collaboration mode, the MPC is able to operate independently based on local information.

For CTO, the ODCT module processes vehicle images, assigning confidence scores to detected objects. Based on confidences, it outputs a cloud query decision to determine whether to proceed independently or seek cloud assistance. If ODCT finds low confidence objects, it would upload

the associated detection results and confidence scores to the cloud. Upon receiving the vehicle query, the CFS assesses the necessity for collaboration by jointly considering potential trajectory improvements and processing delays. If CFS accepts the query, LVM-MPC will be initiated, and the vehicle would subsequently upload streaming images and point clouds to the cloud server. Otherwise, the vehicle keeps on local navigation.

IV. LVM Guided MPC

The proposed LVM-MPC is designed to find a collision-free trajectory under the guidance of LVM and constraints of vehicle dynamics. The system operation is divided into T time slots, and the duration between consecutive slots is Δt. At the t-th time slot, $t \in \{1,\dots,T\}$, the input image is concatenated into a vector $\mathbf{x}_t$. The vehicle state is $\mathbf{s}_t = (a_t, b_t, \theta_t)$, where (a_t, b_t) and θ_t denote position and orientation, respectively. The control command is $\mathbf{u}_t = (v_t, \phi_t)$, with v_t and ϕ_t representing linear velocity and steering angle, respectively. The collaboration state is $\beta_t \in \{0,1\}$.

The consecutive states, $\mathbf{s}_t$ and $\mathbf{s}_{t+1}$, must adhere to the discrete-time kinematic model:

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

where $f(\cdot)$ is the state evolution function determined by the vehicle dynamics [3]. The motion operation has its physical limits, which is given by

$$\mathbf{u}_{\min} \le \mathbf{u}_t \le \mathbf{u}_{\max}, \tag{2}$$

where $\mathbf{u}_{\min}$ and $\mathbf{u}_{\max}$ are the minimum and maximum input limits, respectively. To prevent harsh braking or acceleration and improve the vehicle energy consumption, we must also control the change rate of motions:

$$\Delta\mathbf{u}_{\min} \le \mathbf{u}_t - \mathbf{u}_{t-1} \le \Delta\mathbf{u}_{\max}, \tag{3}$$

where $\Delta\mathbf{u}_{\min}$ and $\Delta\mathbf{u}_{\max}$ are the minimum and maximum bounds on change rates of motion vectors.

A. LVM Perception

If $\beta_t = 0$, the vehicle adopts a local detection model, denoted as a function $g_{\text{local}}(\cdot)$, to map $\mathbf{x}_t$ into a list $\mathscr{B}_t = g_{\text{local}}(\mathbf{x}_t)$, where $\mathscr{B}_t = \{(y_{1,t}, \mathbb{O}_{1,t}), \cdots, (y_{M,t}, \mathbb{O}_{M,t})\}$, with $y_{m,t}$ and $\mathbb{O}_{m,t}$ representing the category and bounding box of the k-th obstacle. Each box is assigned a confidence level $c_{m,t} \in [0,1]$. The design of the local detection model is based on the unsniffer architecture [18].

With unknown objects in $\mathbf{x}_t$, $\mathscr{B}_t$ may mismatch from the ground truth $\mathscr{B}_t^*$, forcing the vehicle to take a detour. To this end, we need to shift the collaboration state from $\beta_t = 0$ to $\beta_t = 1$. Then, the cloud server adopts an LVM-based detection model, denoted as a function $g_{\text{cloud}}(\cdot)$, to calibrate $\mathscr{B}_t$ into $\mathscr{B}_t^\diamond = g_{\text{cloud}}(\mathbf{x}_t)$.

The architecture of our cloud detection model is illustrated in Fig. 2. First, SAM [17], [19] generates a comprehensive set of binary masks $\{\mathbf{v}_i\}$ through zero-shot segmentation, where each mask $\mathbf{v}_i$ corresponds to diverse image elements

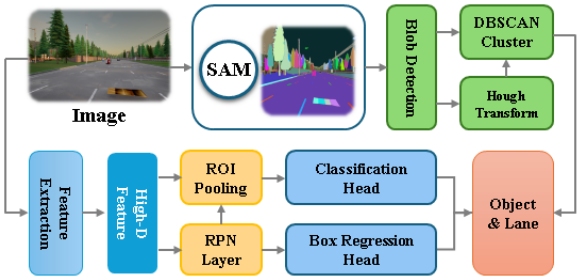


Fig. 2: LVM cloud perception based on SAM.

including objects, lanes, background regions, and edge contours. The mask representation is formally defined as:

$$\mathbf{v}_i(p) = \begin{cases} 1, & \text{if pixel } p \text{belongs to a segmented region} \\ 0, & \text{otherwise} \end{cases}.$$

Second, we localize the road surface by identifying $\mathbf{v}_{\text{road}}$—the mask with maximal pixel continuity along the image's bottom edge, leveraging the prior that roads typically extend toward the camera viewpoint. Third, a proximity-based filter isolates traffic-relevant masks: for each candidate $\mathbf{v}_i$, we retain it only if there exists a pixel $p \in \mathbf{v}_i$ satisfying

$$\min_{p' \in \mathbf{v}_{\text{road}}} \|p - p'\|_2 < \delta_{\text{pixel}}, \tag{4}$$

where δ_{pixel} is set to 0.01% of the image width to tolerate minor localization errors. Finally, density-based clustering merges fragmented SAM regions within unsniffer's bounding boxes $B_t^\diamond$, ensuring spatial consistency between segmentation masks and detection outputs while recalibrating confidence scores $\{c_{m,t}^\diamond\}$ through cross-modal alignment.

To generate the behavior decision $\gamma_t \in \{\text{keep}, \text{left}, \text{right}\}$, the key is to obtain the object-lane topology relation by associating each object with a lane ID. Here we propose to detect lane markings by checking the aspect ratio and orientation of each SAM mask. Specifically, we introduce the concept of slope difference between the left and right edges of the mask. Given the width W of the segment rectangle and the camera intrinsics V (height) and ω (rotation angle), the slope difference between the left and right edges of the mask in the image plane is given by:

$$\Delta\text{slope} = \frac{W}{V\cos\omega}. \tag{5}$$

This value helps us to filter out non-lane marking objects by comparing Δslope for each mask with target Δslope for lane markings. As such, the objects in $\mathscr{B}_t^\diamond$ are associated with the lane IDs. The fused information, including object boxes, object categories, lane positions, and object-lane topology relations, are passed into a decision tree, which generates behavior decision γ_t for subsequent planning and control.

B. Cloud Guided MPC

The MPC module is required to operate in a receding-horizon fashion that allows repeated planning at a high frequency. We generate a local horizon that is divided into H time slots with $\mathscr{H} = \{t, \cdots, t+H\}$, where H is the length of receding horizon. Let $\mathscr{S} = \{\mathbf{s}_t, \cdots, \mathbf{s}_{t+H}\}$ and

$\mathscr{U} = \{\mathbf{u}_t, \cdots, \mathbf{u}_{t+H}\}$ denote the states and actions to be optimized within the local horizon. First, they need to satisfy vehicle dynamic constraints (1)–(3), which are expressed as $\{\mathscr{S}, \mathscr{U}\} \in \mathscr{F}$. Second, they need to satisfy collision avoidance constraints. Given the local perception $\mathscr{B}_t, \{c_{m,t}\}$, cloud perception $\mathscr{B}_t^\diamond, \{c_{m,t}^\diamond\}$, and the collaboration state β_t, the collision avoidance constraints are written as [3], [10]

$$\begin{aligned}&\min_{\mathbf{v},\mathbf{o}}\{\|\mathbf{v}-\mathbf{o}\|_2 | \mathbf{v} \in \mathbb{V}_h(\mathbf{s}_h), \mathbf{o} \in (1-\beta_t)\mathbb{O}_{m,h} + \beta_t \mathbb{O}_{m,h}^\diamond\} \\ &\geq d_0 + (1-\beta_t)(1-c_{m,h})d_{\text{inf}} + \beta_t(1-c_{m,h}^\diamond)d_{\text{inf}},\ \forall m,h, \quad (6)\end{aligned}$$

where $\mathbb{V}(\mathbf{s}_h)$ denote the real-time vehicle moving polytope, d_0 is the minimum safety distance, d_{inf} is the maximum inflation distance. As such, a larger safety distance is guaranteed when the confidence score is smaller [5], [10]. Moreover, collaboration state β_t would also impact the inflation distance, as it switches between the confidence scores $c_{m,h}$ and $c_{m,h}^\diamond$. Lastly, reconfiguring the inflation distance would also *reconfigure the control style, e.g., from conservative to aggressive control*.

Having the vehicle dynamics and collision avoidance constraints satisfied, it is then crucial to design a cost function to optimize the navigation performance. In particular, given the vehicle decision $\{\gamma_t\}$, the cloud decision $\{\gamma_t^\diamond\}$, and the collaboration state β_t, the guidance waypoints are generated from the target lane ID as $\mathscr{W}^\diamond = \{\mathbf{w}_t^\diamond, \mathbf{w}_{t+1}^\diamond, \cdots, \mathbf{w}_{t+H}^\diamond\}$ if $\beta_t = 1$ and $\mathscr{W} = \{\mathbf{w}_t, \mathbf{w}_{t+1}, \cdots, \mathbf{w}_{t+H}\}$ if $\beta_t = 0$.[1] As such, the cost function of MPC is given by the distance between the guidance and executable trajectories:

$$C(\mathscr{S}) = \sum_{h=t}^{t+H} \|\mathbf{s}_h - [(1-\beta_t)\mathbf{w}_t + \beta_t \mathbf{w}_t^\diamond]\|^2. \quad (7)$$

It can be seen that the collaboration states would impact the reference waypoints. Based on above discussions, the cloud guided MPC problem is formulated as

$$\mathsf{P}: \min_{\{\mathscr{S},\mathscr{U}\}\in\mathscr{F}} C(\mathscr{S}), \quad \text{s.t. } (6), \quad (8)$$

which is solved via the RDA method in real time [3].

V. Collaboration Timing Optimization

For LVM-MPC in Section IV, the key is to determine the collaboration states $\{\beta_t\}$. Improper collaboration may not improve navigation performance while involving additional computation (e.g., large model inference) and communication (e.g., sensor data sharing) costs. To determine the best collaboration timing, we need joint query-service optimization, where the vehicle adopt ODCT to decide when to seek cloud assistance and the cloud adopts CFS to decide whether to offer cloud service.

A. Large Model Query Optimization via ODCT

As mentioned in Section IV-A, the local detection would output confidence values $\{c_{m,t}\}$ for all detected boxes. The ODCT needs to decide whether to trigger a large model query, denoted as $\alpha_t \in \{0,1\}$, to the cloud for assistance based on these $\{c_{m,t}\}$.

Here, we adopt a simple approach termed threshold based decision, which classify objects as either known or unknown by comparing a confidence threshold, $C_{\text{threshold}}$, with the confidence score $c_{m,t}$ of a bounding box. If $c_{m,t}$ is below $C_{\text{threshold}}$, the object is considered unknown. If there exists any unknown object, a query decision, i.e., $\alpha_t = 1$, is triggered, prompting the vehicle to upload current detection results to the cloud for CFS.

To set a proper $C_{\text{threshold}}$, we propose the ODCT algorithm, which optimizes $C_{\text{threshold}}$ by splitting the data into a training dataset $\mathscr{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N_{\text{train}}}$ and validation dataset $\mathscr{D}_{\text{val}} = \{(\mathbf{x}_j, y_j)\}_{j=1}^{N_{\text{val}}}$. Let $\mathscr{Y}_{\text{train}}$ denote the set of known class labels. The training dataset contains only known objects, i.e., $\forall y_j \in \mathscr{Y}_{\text{train}}$ while the validation dataset contains both known and unknown objects, i.e., $\exists y_j \notin \mathscr{Y}_{\text{train}}$. By training a model on $\mathscr{D}_{\text{train}}$ and validating the model on $\mathscr{D}_{\text{val}}$, we can obtain the predicted class $\hat{y}_j$ of the j-th object and its associated confidence c_j. With $\{\hat{y}_j, c_j\}$, we then optimize $C_{\text{threshold}}$ by maximizing the objective function $G(C_{\text{threshold}})$ over the validation set as

$$C_{\text{threshold}}^* = \arg \max_{C_{\text{threshold}} \in [0,1]} G(C_{\text{threshold}}), \quad (9)$$

where

$$\begin{aligned}G(C_{\text{threshold}}) = &\underbrace{\frac{\sum_{j=1}^{N_{\text{val}}} \mathbb{I}(\hat{y}_j \notin \mathscr{Y}_{\text{train}}) \cdot \mathbb{I}(c_j < C_{\text{threshold}})}{\sum_{j=1}^{N_{\text{val}}} \mathbb{I}(\hat{y}_j \notin \mathscr{Y}_{\text{train}})}}_{\textit{recall of correct unknown detections}} \\ &+ \underbrace{\frac{\sum_{j=1}^{N_{\text{val}}} \mathbb{I}(\hat{y}_j \in \mathscr{Y}_{\text{train}}) \cdot \mathbb{I}(c_j > C_{\text{threshold}})}{\sum_{j=1}^{N_{\text{val}}} \mathbb{I}(\hat{y}_j \in \mathscr{Y}_{\text{train}})}}_{\textit{recall of correct known detections}} \quad (10)\end{aligned}$$

and $\mathbb{I}$ is indicator function. It can be seen that $J(C_{\text{threshold}})$ automatically balances the detection of unknown objects, while minimizing false positives from known objects. This G function is guaranteed to be unimodal, since the number of correct unknown detections is monotonically increasing in $C_{\text{threshold}}$ and the number of correct known detections is monotonically decreasing in $C_{\text{threshold}}$.

B. Large Model Service Optimization via CFS

The CFS aims to balance trajectory improvement against cloud query costs by simulating agent interactions in the environment [20] [21]. It takes local detections $\mathscr{B}_t$, local confidence $\{c_{m,t}\}$, and vehicle kinematics in (1)–(3) as input and generates the service decision β_t as output. At this pre-collaboration point, the cloud has not received the image $\mathbf{x}_t$ yet and the cloud perception results $\mathscr{B}_t^\diamond, \{c_{m,t}^\diamond\}$ are not known. Therefore, we need to predict what will happen after the large model engagement.

Specifically, let function $J(\{c_{m,t}\}, H)$ denote the trajectory length of H time slots given detection confidences $\{c_{m,t}\}$. To compute J, we need to solve problem (8) and obtain the optimal solution $\mathscr{S}^* = \{\mathbf{s}_t^*(\{c_{m,t}\}), \cdots, \mathbf{s}_{t+H}^*(\{c_{m,t}\})\}$, where

[1] In our experiment, the specific values of $\mathscr{W}^\diamond, \mathscr{W}$ are extracted from the high-definition map provided by the Carla simulator.

the optimal state $\{\mathbf{s}_t^*\}$ is a function of $\{c_{m,t}\}$ due to constraint (6). The trajectory length under $\beta_t = 0$ is

$$J(\{c_{m,t}\}, H) = \sum_{i=t}^{t+H-1} \|\mathbf{s}_{i+1}^*(\{c_{m,t}\}) - \mathbf{s}_i^*(\{c_{m,t}\})\|, \quad (11)$$

which completes the MPC forward simulation once.

Now, assume that the confidences after cloud perception follow a certain distribution $c_{m,t}^{\diamond} \in \mathscr{P}$. For example, we can adopt uniform distribution $\mathscr{P} = \mathscr{U}(C_{\text{threshold}} - \Delta c, C_{\text{threshold}} + \Delta c)$, where the mean is set to $C_{\text{threshold}}$ since the LVM is likely to recognize the objects, transforming unknown into known ones, and Δc (e.g., $\Delta c = 0.1$) accounts for the failure detection cases of LVM. Therefore, the expected trajectory improvement achieved by redetecting obstacles via collaboration is given by

$$\begin{aligned}\mathbb{E}[\Delta J] &= \int_{c_{m,t}^{\diamond} \in \mathscr{P}} \left[J(\{c_{m,t}^{\diamond}\}, H) - J(\{c_{m,t}\}, H)\right] dc_{m,t}^{\diamond} \\ &\approx \frac{1}{N} \sum_{\{c_{m,t}^{(n)}\}} \left[J(\{c_{m,t}^{(n)}\}, H) - J(\{c_{m,t}\}, H)\right], \quad (12)\end{aligned}$$

where the second line of approximation is obtained by collecting N samples from $\mathscr{P}$, resulting in N times of MPC forward simulations.

In practice, there may exist communication delay T_{com} and computation delay T_{comp} during collaboration and the trajectory improvement in (12) may be degraded. In such a case, the vehicle would first execute local navigation for a time duration of $T_{\text{com}} + T_{\text{comp}}$, which correspond to $L = \lfloor (T_{\text{com}} + T_{\text{comp}})/\Delta t \rfloor$ time slots, and then execute collaborative navigation for $H - L$ time slots. Thus, the expected trajectory length under latency is

$$\begin{aligned}J'(\{c_{m,t}, c_{m,t}^{\diamond}\}, L, H) &= \sum_{i=t}^{t+L-1} \|\mathbf{s}_{i+1}^*(\{c_{m,t}\}) - \mathbf{s}_i^*(\{c_{m,t}\})\| \\ &+ \sum_{i=L}^{t+H-1} \|\mathbf{s}_{i+1}^*(\{c_{m,t}^{\diamond}\}) - \mathbf{s}_i^*(\{c_{m,t}^{\diamond}\})\|. \quad (13)\end{aligned}$$

Accordingly, the expected trajectory improvement becomes

$$\mathbb{E}[\Delta J'] = \int_{c_{m,t}^{\diamond} \in \mathscr{P}} \left[J'(\{c_{m,t}, c_{m,t}^{\diamond}\}, L, H) - J(\{c_{m,t}\}, H)\right] dc_{m,t}^{\diamond}.$$

To understand how CFS works, an Carla experiment is illustrated in Fig. 3. The ego vehicle detects unknown objects and trigger the cloud query, i.e., $\alpha_t = 1$. However, the cloud finds no significant trajectory change after CFS. Consequently, it will rejects the query and outputs $\beta_t = 0$, making the ego vehicle to navigate on its own.

VI. Experiments

We implemented the OCP system using Python and ROS in the CARLA platform [16]. The ego-vehicle, a Tesla Model3 with longitudinal and lateral wheelbases of 2.87 m and 1.75 m, is equipped with an RGBD camera and a 64-line lidar at 10 Hz. The local detection model processes sensory inputs at up to 100 FPS and local MPC model generates control outputs at 20 Hz. The safety distance of MPC is $d_0 = 0.3$ m and $d_{\text{inf}} = 3.0$ m. The length of prediction horizon

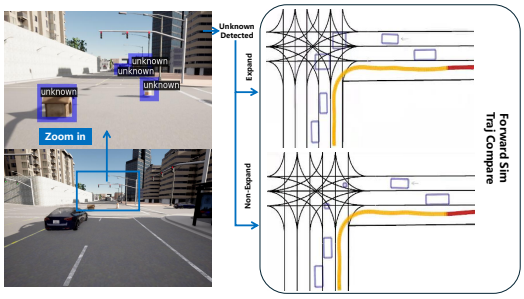


Fig. 3: The case of $\alpha_t = 1$ but $\beta_t = 0$ for CFS.

is $H = 18$, with time step of $\Delta t = 0.3$ s. All experiments are conducted on a Ubuntu workstation with two NVIDIA RTX 3090 GPUs. For comparison, we implement the following schemes: 1) **Local-only strategy (LOS)**, which adopts local perception and a state-of-the-art MPC (i.e., RDA) [3]; 2) **Periodical collaboration strategy (PCS)**: which is a baseline collaboration scheme with fixed interval between consecutive cloud services [12]; 3) **OCP (ours)**, which is the LVM-MPC collaboration scheme based on ODCT and CFS. Note that the collaborative planner in [9] does not involve LVM, and has a similar performance as LOS [3].

A. Experiment 1: Unknown Object Detection

We first conduct experiments to validate the LVM perception and ODCT. We collect a large dataset in Carla Town06 map, generating over 30,000 frames of 1080x720 RGB images synchronized with vehicle odometries. Each frame is annotated with ground truth bounding boxes and undergoes rigorous quality filtering to ensure object visibility, with occluded or partially visible instances excluded during data curation. As shown in Table. I, the dataset contains 49 object categories (including vehicles and 48 other objects in Carla) and partitioned into three subsets for open vocabulary detection. Out of the 49 categories, we set 38 categories as known and the remaining 11 as unknown. The training set of 9,647 samples covers all 38 known classes. The validation set contains 2,060 samples with 21 known classes and 5 unknown classes. The test set contains 3,400 samples with 19 known classes and the other 6 unknown classes.[2] Unknown classes in validation and test sets are strictly non-overlapping and excluded from the training subset.

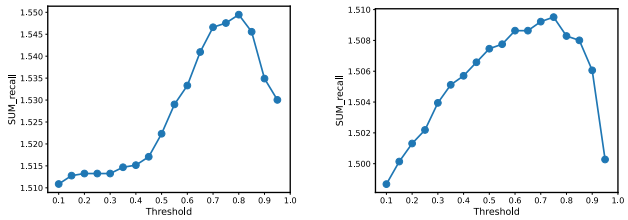
The local and cloud models are implemented based on the unsniffer architecture [18], which provides confidence score for each detection. The local detection model is trained on the standard training subset (9,647 samples) and fine-tuned on the validation set to optimize C_{threhold}. The final evaluation is performed on the test set. A complete training set of 15,107 samples is adopted that spans the 49 classes to train the cloud detection model.

First, the value of G (i.e., sum of two recalls) in (10) versus $C_{\text{threshold}}$ is shown in Fig. 4. It can be seen from Fig. 4a that the recall score reaches its maximum on the validation set when $C_{\text{threshold}} = 0.8$. As shown in Fig. 4b, the highest

[2]The test set involves objects such as trashcan, doghouse and streetbarrier. This aims to demonstrate the robustness of the recognition system in real-world settings. In the subsequent experiments 2–4, all the unknown objects are drawn from the test set.

TABLE I: Dataset splits (known vs. unknown classes)

Split	# Samples	Known	Unknown	Total
Local_{Train}	9,647	38	0	38
Local_{Val}	2,060	21	5	26
Local_{Test}	3,400	19	6	25
Cloud_{Train}	15,107	49	0	49



(a) Validation set. (b) Test set.

Fig. 4: Experimental results of ODCT.

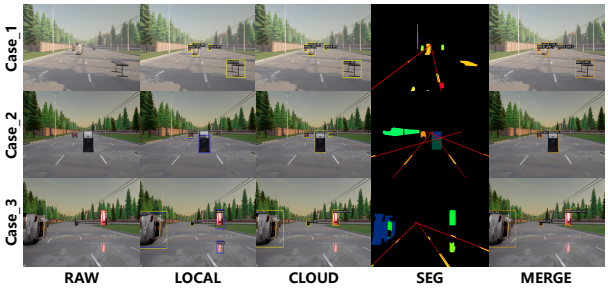


Fig. 5: Comparison of local and cloud perception models.

recall score on the test set is achieved at $C_{\text{threshold}} = 0.75$. This implies that the threshold optimized on the validation set exhibits strong generalization to the test set, although the two sets contain non-overlapping unknown classes. The threshold-optimized local model achieves recalls of 0.83 (known) and 0.71 (unknown) on the validation set, with test performance showing improved known-class recall (0.89) but degraded unknown detection (0.61). The cloud model, trained on all predefined classes (no "unknown" category), attains 0.89 recall on the known-class validation set.

Next, we evaluate the perception performance of the local and cloud models. The associated detection results of the cloud and local models are shown in Fig. 5 (the images are zoomed in for better clarity). We consider 3 cases: case 1 contains only known objects while cases 2–3 contain unknown objects. First, it can be seen that the local model successfully generates bounding boxes for all the known and unknown objects in cases 1–3. However, it does not recognize the categories of unknown objects (i.e., trash can and vending machine) in cases 2 and 3. Fortunately, by leveraging unsniffer, the local model generates low confidence scores below the threshold $C_{\text{threshold}} = 0.8$ for these objects. This prompts the vehicle to generate an "unknown object" warning, triggering a cloud query. Second, the cloud model, with its superior intelligence, correctly recognizes the categories of the unknown objects. The cloud model successfully segments the image into lane (yellow mask) and object masks, and merges the segments with detections for generating the object-lane topology relation (i.e., the lane ID of each object as shown in the MERGE column in Fig. 5).

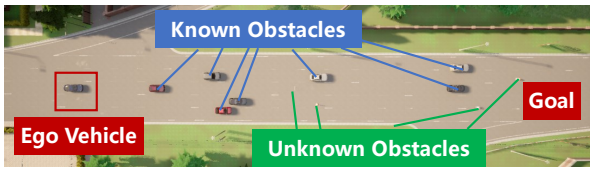


Fig. 6: Scenario configuration for experiment 2.

TABLE II: Quantitative comparison of Experiment 2.

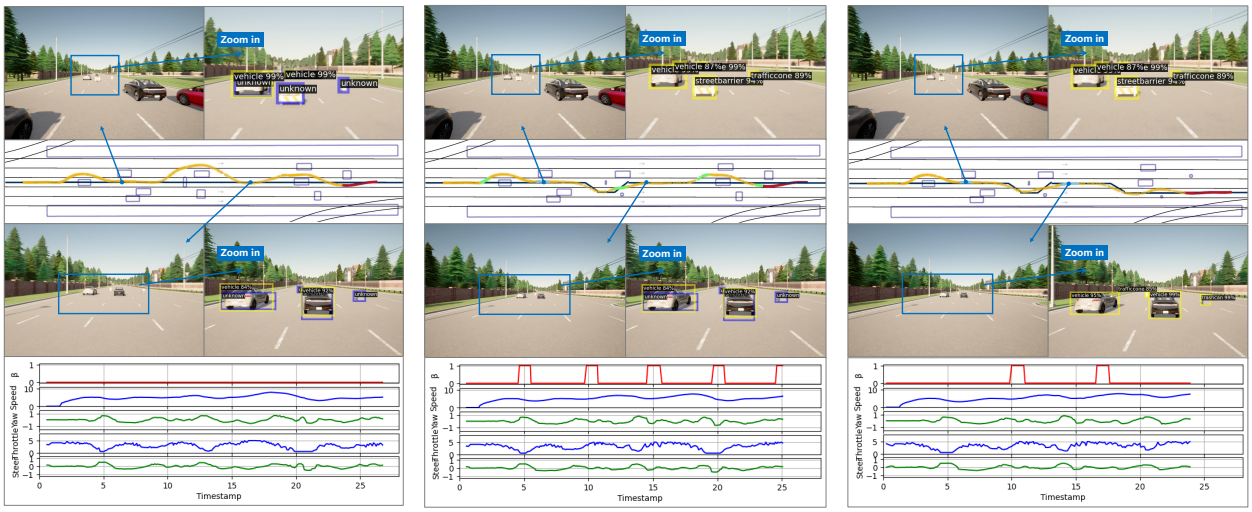
Method (Unit)	FTime (s)	TLenh (m)	AvgLD (m)	SVar (m/s)	MLat (m)
LOS	25.94	124.54	0.53	1.09	1.52
PCS	24.09	122.4	0.59	1.13	1.89
PCS-2	23.53	123.84	0.57	1.39	1.69
OCP	22.35	121.85	0.6	1.3	1.76

B. Experiment 2: Scenarios with Static Obstacles

Next, we evaluate the LOS, PCS, and OCP schemes with a target speed of 6 m/s in the Carla Town06 map. As shown in Fig. 6, the distance between the starting and goal positions is about 110 meters. A total of 7 known objects (i.e., vehicles) and 4 unknown objects (i.e., two traffic cones, one streetbarrier, one trashcan) are located between the ego vehicle and its goal. These 4 unknown objects are grouped into two clusters and placed at two regions, forming two challenges within the track. The trajectories, control profiles, and collaboration states (i.e., $\{\beta_t\}$) of different schemes are shown in Fig. 7a–7c. Yellow trajectories represent local navigation and blue trajectories represent cloud-guided navigation. Green trajectories represent engagement with large model service but no trajectory change. Red trajectories represent future MPC paths.

Without cloud collaboration, the LOS scheme steers away from all the unknown objects, resulting in the longest finish time and trajectory length. The PCS scheme, constrained by fixed time intervals (i.e., 5 s), passes the first challenge but takes a detour in the second challenge, leading to a trajectory and finish time better than LOS but worse than OCP. The proposed OCP scheme, which dynamically interacts with the cloud based on ODCT and CFS, recognizes and reacts to all unknown obstacles efficiently, achieving the shortest finish time and trajectory length. More importantly, as seen from Fig. 7b and Fig. 7c, OCP reduces the number of large model services by 50% compared to PCS (i.e., twice versus 4 times). This is achieved by avoiding those improper services at uncritical points (i.e., green parts on the PCS trajectory). Enhancing perception at these points would not lead to navigation improvements. This demonstrates the **high resource efficiency** brought by the CTO block. Interestingly, at the second challenge, the ego-vehicle steers to the wrong direction before receiving the cloud guidance, but soon corrects its path to the right once cloud guidance provides new detections and waypoints. This demonstrates the **self-correction capability** brought by the LVM-MPC block.

Quantitative comparison of the three schemes is shown in Table II. We repeat the experiment 10 times and evaluate the average metrics. This evaluation consists of finish time (FTime) in s, trajectory length (TLenh) in m, average lateral deviation (AvgLD) in m, speed variability (SVar) in m/s



(a) LOS (b) PCS (c) OCP (Ours)

Fig. 7: Detections, trajectories, control profiles, and collaboration states of different schemes in Experiment 2.

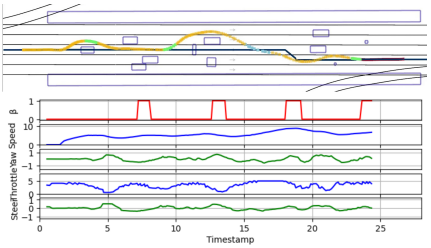


Fig. 8: Trajectory and control profiles of PCS-2.

TABLE III: Quantitative comparison of Experiment 3.

U	Method (Unit)	FTime (s)	TLenh (m)	AvgLD (m)	SVar (m/s)	MLat (m)
0	LOS	42.47	114.27	0.52	2.34	1.59
0	PCS	42.31	114.63	0.53	2.64	1.41
0	OCP	43.00	113.56	0.51	2.41	1.62
1	LOS	48.05	119.95	0.49	2.57	1.54
1	PCS	44.02	114.44	0.53	2.68	1.43
1	OCP	44.86	113.67	0.52	2.53	1.64
2	LOS	55.36	119.32	0.40	2.39	1.55
2	PCS	47.60	114.73	0.49	2.23	1.41
2	OCP	45.44	112.79	0.54	2.48	1.51

and maximum lateral deviation (MLat) in m. To ensure a fair comparison, we also simulate a variant PCS scheme, termed PCS-2, which adopts fixed time intervals of 5.5 s. Its trajectories and control profiles are shown in Fig. 8. It can be seen that the proposed OCP still achieves the the smallest FTime and TLenh, due to the collaboration gain by LVM-MPC and the collaboration timing by CTO.

C. Experiment 3: More Quantitative Evaluations

We further evaluate the LOS, PCS, and OCP schemes at a lower speed (target speed of 3 m/s) with smaller horizon ($H = 10$) and time step ($\Delta t = 0.1s$) in more scenarios. By randomly removing some of the unknown objects in Fig. 6, we can configure difference scenarios with $U \in [0, 1, 2]$. Quantitative results in each scenario are obtained by averaging 30 random simulation runs, with independent realizations in each run. Note that the LOS and PCS schemes may fail due to collision or timeout. Here, we only compute their successful cases for evaluations of trajectory qualities, and the failure cases will be used to compute the success rates.

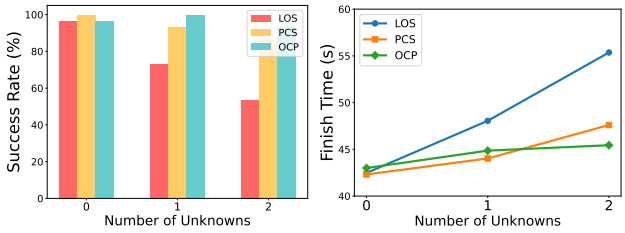
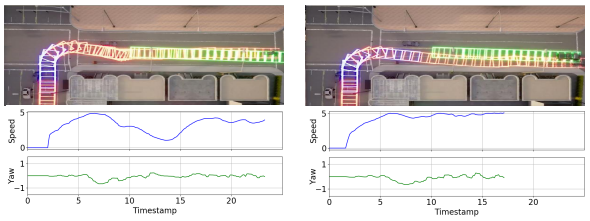


Fig. 9: Comparison of success rate & finish time in Exp. 3.

The evaluation results under different number of unknown objects are shown in Table III. First, as the number of unknown obstacles increases from 0 to 2, the trajectory length and finish time significantly increase for all the simulated schemes, due to the necessity of more collision avoidance operations. Second, the performances of different schemes are similar when the number of unknown objects is zero. However, in presence of a single unknown object, the trajectory length and finish time of LOS increase quickly, as LOS fails in recognizing abnormal objects, leading to either conservative or detouring strategies. The PCS scheme overcomes the above challenge aided by the cloud, but the cloud assistance may be invoked at an inappropriate time due to fixed query intervals. This leads to performance degradation in presence of multiple unknown objects ($U \geq 2$). In contrast, the proposed OCP successfully detects all unknown objects using the cloud LVM, while guaranteeing proper query time using CFS, yielding the shortest trajectory length and finish time among all the simulated schemes.

Success rates are computed based on the following two failure conditions: 1) crash failure: ego-vehicle collides with any obstacle or boundary of the lane; 2) no-progress failure: ego-vehicle gets stuck or becomes off the route, failing to find a feasible path. It can be seen from Fig. 9 the proposed OCP scheme maintains a stable success rate close to 100% in all scenarios. The PCS scheme works well under 0–1 unknown objects, but causes safety issues under 2 unknown objects. The LOS scheme leads to the smallest success rates. Compared to the second-bast scheme, the proposed OCP reduces the success rate by over 6%.



(a) PCS (b) OCP

Fig. 10: Trajectory and control profiles of Experiment 4.

TABLE IV: Quantitative comparison of Experiment 4.

Method (Unit)	**FTime** (s)	**TLenh** (m)	**AvgLD** (m)	**SVar** (m/s)	**MLat** (m)
PCS	23.4	71.47	0.47	2.99	1.68
OCP	17.28	72.90	0.62	2.29	1.44

D. Experiment 4: Scenarios with Dynamic Obstacles

In this experiment, we evaluate our algorithm's performance in a dynamic obstacle scenario at a cross-road in the the Carla Town03 map. The environment includes 2 unknown and 3 known objects. Among the unknowns, one is a static doghouse and the other is a dynamic cybertruck which drives at a constant speed of 4 m/s. The LOS always fails at the turning point due to the uncertain detection of doghouse and cybertruck, which limits the solution space of MPC. Therefore, we only compare the PCS and OCP.

For PCS, the ego-vehicle turns right while avoiding collision with the doghouse, and follows the lead vehicle cybertruck until reaching the goal. In contrast, for OCP, the ego-vehicle triggers another cloud query and service during car following behind the cybertruck. This is because CFS rejects the vehicle query at improper timing. This reserves the chance for subsequent collaboration and make it possible to overtake the cybertruck by following the cloud generated overtaking waypoints rather than the car-following waypoints, as shown in Fig. 10 (red boxes represent local controller; blue boxes represent LVM-MPC; green boxes represent dynamic obstacle). Consequently, compared to PCS, the proposed OCP finishes the trip in a significantly shorter time (over 26.2% time reduction). This demonstrates the superiority of our OCP strategy in dynamic obstacle scenarios. Details result are shown in Table IV.

VII. Conclusion

This paper presented OCP, a framework that enhances autonomous navigation in open scenarios by integrating small local models with large cloud models. The LVM-MPC collaboration was proposed, which ensures interpretable feasible vehicle actions while enjoying intelligent decisions guided by the cloud. Furthermore, joint query-service optimization was achieved via CTO, which addresses the problems of when to seek and offer collaboration. Experiments in the Carla validated the effectiveness of our OCP, in terms of identifying unknown objects and conducting safe efficient autonomous navigation in various scenarios.

References

[1] S. Feng, H. Sun, X. Yan, H. Zhu, Z. Zou, S. Shen, and H. X. Liu, "Dense reinforcement learning for safety validation of autonomous vehicles," *Nature*, vol. 615, no. 7953, pp. 620–627, 2023.

[2] R. Han, S. Wang, S. Wang, Z. Zhang, J. Chen, S. Lin, C. Li, C. Xu, Y. C. Eldar, Q. Hao *et al.*, "Neupan: Direct point robot navigation with end-to-end model-based learning," *IEEE Transactions on Robotics*, 2025.

[3] R. Han, S. Wang, S. Wang, Z. Zhang, Q. Zhang, Y. C. Eldar, Q. Hao, and J. Pan, "Rda: An accelerated collision free motion planner for autonomous navigation in cluttered environments," *IEEE Robotics and Automation Letters*, vol. 8, no. 3, pp. 1715–1722, 2023.

[4] Z. Han and et al., "An efficient spatial-temporal trajectory planner for autonomous vehicles in unstructured environments," *IEEE Transactions on Intelligent Transportation Systems*, vol. 25, no. 2, pp. 1797–1814, Oct. 2024.

[5] D. Li, B. Liu, Z. Huang, Q. Hao, D. Zhao, and B. Tian, "Safe motion planning for autonomous vehicles by quantifying uncertainties of deep learning-enabled environment perception," *IEEE Transactions on Intelligent Vehicles*, vol. 9, no. 1, pp. 2318–2332, Jan. 2024.

[6] W.-B. Kou, Q. Lin, M. Tang, S. Wang, R. Ye, G. Zhu, and Y.-C. Wu, "Enhancing large vision model in street scene semantic understanding through leveraging posterior optimization trajectory," *arXiv preprint arXiv:2501.01710*, 2025.

[7] S. Hu, Z. Fang, Z. Fang, X. Chen, and Y. Fang, "Agentscodriver: Large language model empowered collaborative driving with lifelong learning," *arXiv preprint arXiv:2404.06345*, 2024.

[8] Z. Xu, Y. Zhang, E. Xie, Z. Zhao, Y. Guo, K. K. Wong, Z. Li, and H. Zhao, "Drivegpt4: Interpretable end-to-end autonomous driving via large language model," *arXiv preprint arXiv:2310.01412*, 2023.

[9] G. Li, R. Han, S. Wang, F. Gao, Y. C. Eldar, and C. Xu, "Edge accelerated robot navigation with collaborative motion planning," *IEEE/ASME Transactions on Mechatronics*, 2024.

[10] S. Zhang, H. Li, S. Zhang, S. Wang, D. W. K. Ng, and C. Xu, "Multi-uncertainty aware autonomous cooperative planning," in *2024 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2024, pp. 1018–1025.

[11] P. Wang, M. Zhu, H. Lu, H. Zhong, X. Chen, S. Shen, X. Wang, and Y. Wang, "Bevgpt: Generative pre-trained large model for autonomous driving prediction, decision-making, and planning," *arXiv preprint arXiv:2310.10357*, 2023.

[12] A. K. Tanwani, R. Anand, J. E. Gonzalez, and K. Goldberg, "Rilaas: Robot inference and learning as a service," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4423–4430, Jul. 2020.

[13] H. Sha, Y. Mu, Y. Jiang, L. Chen, C. Xu, P. Luo, S. E. Li, M. Tomizuka, W. Zhan, and M. Ding, "Languagempc: Large language models as decision makers for autonomous driving," *arXiv preprint arXiv:2310.03026*, 2023.

[14] B. Morrell, R. Thakker, À. Santamaria Navarro, A. Bouman, X. Lei, J. Edlund, T. Pailevanian, T. S. Vaquero, Y. L. Chang, T. Touma *et al.*, "NeBula: TEAM CoSTARs robotic autonomy solution that won phase II of DARPA subterranean challenge," *Field Robotics*, vol. 2, pp. 1432–1506, 2022.

[15] J. Ichnowski and et al., "FogROS2: An adaptive platform for cloud and fog robotics using ROS 2," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2023, pp. 5493–5500.

[16] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[17] A. Kirillov, E. Mintun, N. Ravi, H. Mao, C. Rolland, L. Gustafson, T. Xiao, S. Whitehead, A. C. Berg, W.-Y. Lo, P. Dollar, and R. Girshick, "Segment anything," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, October 2023, pp. 4015–4026.

[18] W. Liang, F. Xue, Y. Liu, G. Zhong, and A. Ming, "Unknown sniffer for object detection: Don't turn a blind eye to unknown objects," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2023, pp. 3230–3239.

[19] N. Ravi, V. Gabeur, Y.-T. Hu, R. Hu, C. Ryali, T. Ma, H. Khedr, R. Rädle, C. Rolland, L. Gustafson *et al.*, "Sam 2: Segment Anything in Images and Videos," *arXiv preprint arXiv:2408.00714*, 2024.

[20] W. Ding, L. Zhang, J. Chen, and S. Shen, "EPSILON: An efficient planning system for automated vehicles in highly interactive environments," *IEEE Transactions on Robotics*, vol. 38, no. 2, pp. 1118–1138, 2021.

[21] M. Lauri and R. Ritala, "Planning for robotic exploration based on forward simulation," *Robotics and Autonomous Systems*, vol. 83, pp. 15–31, 2016.