Vision-Based Reactive Temporal Logic Motion Planning for Quadruped Robots in Unstructured Dynamic Environments

Zhangli Zhou, Ziyang Chen, Mingyu Cai, Zhijun Li, *Fellow, IEEE*, Zhen Kan, *Senior Member, IEEE*, and Chun-Yi Su, *Senior Member, IEEE*

***Abstract*—Temporal logic-based motion planning has been extensively studied to address complex robotic tasks. However, existing works primarily focus on static environments or assume the robot has full observations of the environment. This limits their practical applications since real-world environments are often dynamic, and robots may suffer from partial observations. To tackle these issues, this study proposes a framework for vision-based reactive temporal logic motion planning (V-RTLMP) for robots integrated with LiDAR sensing. The V-RTLMP is designed to perform high-level linear temporal logic (LTL) tasks in unstructured dynamic environments. The framework comprises two modules: offline preplanning and online reactive planning. Given LTL specifications, the preplanning phase generates a reference trajectory over the continuous workspace via sampling-based methods using prior environmental knowledge. The online reactive module dynamically adjusts the robot trajectory based on real-time visual perception to adapt to environmental changes. Extensive numerical simulations and real-world experiments using a quadruped robot demonstrate the effectiveness of the proposed vision-based reactive motion planning.**

***Index Terms*—Computer vision, formal methods in automation and robotics, linear temporal logic (LTL), online motion planning, quadruped robot.**

I. Introduction

WHEN operating in unstructured dynamic environments, robots face the challenge of unexpected changes that can lead to inefficient behavior or even mission failure. For instance, as shown in Fig. 1, a quadruped robot is instructed to fetch a Coke from the fridge, but a human places a Coke on a nearby table on the robot's path. A reactive motion planning strategy should adjust the plan to fetch the nearby Coke rather than proceeding to the fridge, resulting in a more efficient solution. This problem can be more challenging if a series of tasks with temporal logic constraints are considered where the task operations are coupled with the unstructured dynamic environment. Hence, this work is particularly motivated to investigate reactive temporal logic-based motion planning.

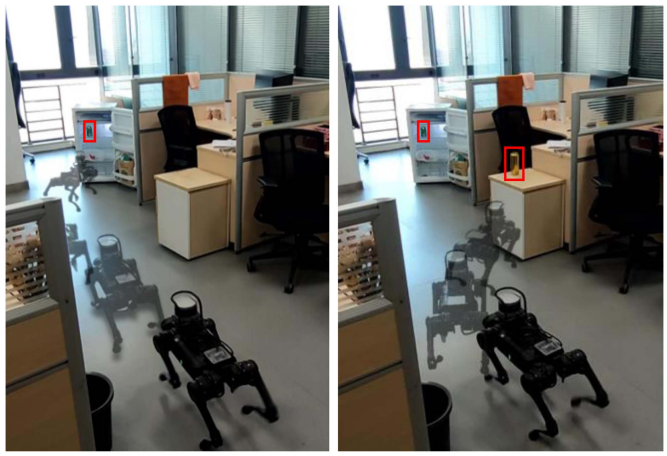

Fig. 1. Left: The quadruped robot originally plans to fetch a Coke in the fridge. Right: It decides to take the Coke nearby instead.

Related work: Temporal logic-based motion planning has received significant attention in recent years [1], [2], [3], [4], [5] due to its capability to express complex robotic tasks beyond traditional point-to-point navigation [6]. Nevertheless, previous studies have either focused on static environments [7], [8] or assumed complete observations of the environment by the robot [9], [10]. These works typically involve generating high-level discrete plans using model-checking theories [11] and following them using low-level hybrid controllers for practical implementation. However, in unstructured dynamic environments, or when the robot has only local perception, the feasibility of these approaches cannot be guaranteed. Reactive and sensor-based motion planning methods have been developed to

Manuscript received 10 March 2023; revised 12 June 2023; accepted 2 July 2023. Date of publication 4 August 2023; date of current version 19 January 2024. This work was supported in part by the National Natural Science Foundation of China under Grant U2013601, Grant 62173314, and Grant 62133013, in part by the Anhui Provincial Natural Science Foundation, Anhui Energy-Internet Joint Program under Grant 2008085UD01, and in part by CAAI-Huawei MindSpore Open Fund. *(Corresponding author: Zhen Kan.)*

Zhangli Zhou, Ziyang Chen, Zhijun Li, and Zhen Kan are with the Department of Automation, University of Science and Technology of China, Hefei 230026, China (e-mail: zzl1215@mail.ustc.edu.cn; yy1220@mail.ustc.edu.cn; zjli@ieee.org; zkan@ustc.edu.cn).

Mingyu Cai is with the Department of Mechanical Engineering, University of California, Riverside, CA 92521 USA (e-mail: mingyu–cai@lehigh.edu).

Chun-Yi Su is with the School of Intelligent Manufacturing, Taizhou University, Taizhou 318000, China, on leave from the Gina Cody School of Engineering and Computer Science, Concordia University, Montreal, QC H3G 1M8, Canada (e-mail: chun-yi.su@concordia.ca).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TIE.2023.3299048.

Digital Object Identifier 10.1109/TIE.2023.3299048

tackle this problem by enabling robots with limited perception capabilities to execute temporal logic-based motion planning in unknown, static environments [12], [13], [14]. These works adjust plans online to complete infeasible tasks as much as possible. By employing model predictive control, these methods have been extended to dynamic environments [15], [16], [17]. Recent works [18], [19] aim to find the optimal solution within a limited time domain based on real-time sensing in scenarios where task conflicts arise in dynamic environments. Despite recent progress, the aforementioned works are built upon a key assumption that the goal is implicitly coupled with a specified location (e.g., given the goal of fetching the Coke, the robot is actually programmed to go to the fridge to get the Coke in practice). This may significantly limit its applicability, as the goal can be time-varying or even infeasible (e.g., the Coke in the fridge may have been taken away), especially in a dynamic environment. A more realistic idea at this point is to formulate the robot's motion planning task based on the target object rather than its position in the environment. For the example of Coke fetching, motion planning focuses on the Coke itself rather than the stored position. Such a design is more practical in terms of mission description, as we can track time-varying targets instead of fixed locations. This is consistent with how humans describe tasks, connecting to vision-based semantic detection and localization for reactive online planning.

Contribution: In this work, we consider a robot integrated with vision and LiDAR performing high-level linear temporal logic (LTL) tasks that encode position constraints to the target object. To achieve this, we developed a vision-based reactive temporal logic motion planning (V-RTLMP) framework consisting of two layers: offline preplanning and vision-based online reactive motion planning. The preplanning phase generates a preliminary trajectory that satisfies the temporal logic constraints based on the robot's prior knowledge through sampling-based methods [20]. The preliminary trajectory in the reactive online module is dynamically adjusted based on the robot's real-time visual perception to react reactively to environmental changes. Specifically, we first build a local scene understanding module using vision and point clouds to update the robot's prior knowledge during mission operation. Reactive planning is initiated if the preplanned trajectory conflicts with the updated knowledge. Two mechanisms, i.e., the greedy and gate mechanisms, are designed to implement reactive planning for the robot. The greedy mechanism is to improve efficiency, and the gate mechanism is to verify task satisfaction.

The main contributions can be summarized as follows.

1) We have developed a local scene perception algorithm that enables real-time detection and localization of dynamic elements in the environment, thus allowing the robot to continuously update its knowledge about the surroundings.
2) An online reactive motion planning framework, namely V-RTLMP, is developed to empower the robots to execute tasks subject to temporal logic constraints in unstructured dynamic environments, relying on local perception.
3) Extensive numerical simulations and physical experiments are conducted using the Unitree A1 quadruped robot to validate the efficacy of V-RTLMP.

II. Preliminary and Problem Formulation

A. Preliminaries

Considering a robot operating in an unstructured dynamic environment Env, the interaction between the robot and Env can be captured by a weighted transition system (WTS).

Definition 1: A WTS in Env is a tuple $\mathcal{T} = (X, x_0, \rightarrow_{\mathcal{T}}, \mathrm{AP}, L_X, C_{\mathcal{T}})$, where X is the geometric space of Env; x_0 is the initial state of robot; $\rightarrow_{\mathcal{T}} \subseteq X \times X$ is the geometric transition relation s.t. $(x, x') \in \rightarrow_{\mathcal{T}}$ if $\mathrm{dist}(x, x') \leq \eta$, where η is a predefined maximum step size, and the transition from x to x' is collision-free; AP is the set of atomic propositions indicating the labels of regions; $L_X : X \rightarrow \mathrm{AP}$ is the time-varying labeling function that returns an atomic proposition satisfied at the current location x; and $C_{\mathcal{T}} : (\rightarrow_{\mathcal{T}}) \rightarrow \mathbb{R}^+$ is the geometric Euclidean distance, i.e., $C_{\mathcal{T}}(x, x') = \mathrm{dist}(x, x'), \forall (x, x') \in \rightarrow_{\mathcal{T}}$.

Let $\mathcal{M}$ be the robot's prior knowledge about Env, which contains the labels of regions Env. In practice, $\mathcal{M}$ can be a topological semantic map previously built using semantic simultaneous localization and mapping approaches, e.g., [21]. When considering dynamic environments, the prior information of $\mathcal{M}$ can be used for offline preplanning and $\mathcal{M}$ should be updated online to reflect environment changes and facilitate reactive planning. The dynamic relationship between the objects and the environment is captured by the time-varying labeling function L_X in Definition 1.

Due to the rich expressivity, LTL is used throughout this work to describe the high-level robot missions.

Definition 2 [11]: LTL is a formal language over a set of atomic propositions AP and combinations of Boolean and temporal operators. The syntax of LTL is defined as

$$\phi := \mathrm{true} \,|\, ap \,|\, \phi_1 \wedge \phi_2 \,|\, \neg\phi_1 \,|\, \bigcirc \phi \,|\, \phi_1 \mathcal{U} \phi_2$$

where, $ap \in \mathrm{AP}$ is an atomic proposition, *true*, *negation* $\neg$, and *conjunction* $\wedge$ are propositional logic operators, and *next* $\bigcirc$ and *until* $\mathcal{U}$ are temporal operators. Based on that, other propositional logic operators, such as *false*, *disjunction* $\vee$, *implication* $\rightarrow$, and temporal operators, such as *always* $\Box$ and *eventually* $\Diamond$, can be defined.

The semantics of LTL formulae are expressed over infinite words of 2^{AP}, where 2^{AP} denotes the power set of AP. An infinite word starting from step 0 can be defined as $o = o_0 o_1 \ldots$ s.t. $\forall o(t) \in 2^{\mathrm{AP}}, t \in \mathbb{Z}_{\geq 0}$. The satisfaction of an LTL formula ϕ by the word o is denoted as $o \models \phi$. The semantics of LTL is defined as

$$\begin{aligned}
o \models ap &\Leftrightarrow ap \in o(0) \\
o \models \phi_1 \wedge \phi_2 &\Leftrightarrow o \models \phi_1 \text{ and } o \models \phi_2 \\
o \models \neg\phi &\Leftrightarrow o \not\models \phi \\
o \models \bigcirc\phi &\Leftrightarrow o[1{:}] \models \phi \\
o \models \phi_1 \mathcal{U} \phi_2 &\Leftrightarrow \exists t \text{ s.t. } o[t{:}] \models \phi_2, \forall t' \in [0, t), o[t'{:}] \models \phi_1.
\end{aligned}$$

For more details about LTL syntax, semantics, and model checking, we refer readers to the book [11].

To better describe the robot task under Env, we describe the robot task in the form of a constrained task ϕ_c. Specifically, we describe the task using the semantics of target objects whose superscript indicates its positional constraints. For instance, the constrained task $\phi_c = \Diamond(\mathrm{key}^{\mathrm{drawer}_A} \bigcirc (\Diamond \mathrm{car}))$ requires the

robot to first get the key from drawer A before starting the car. Note that the task ϕ_c can always be expressed by an LTL formula combined with the time-varying labeling function $L_X^B : \text{AP} \to \mathbb{R}^2$. For instance, ϕ_c can be expressed by an LTL formula $\phi = \diamondsuit(\text{key} \bigcirc (\diamondsuit \text{car}))$ (i.e., get the key before starting the car), while $L_X^B(\text{key}) = \text{drawer}_A$ enforces that the key should be obtained from drawer A.

Any LTL formula can be converted to a nondeterministic Büchi automata (NBA).

Definition 3: An NBA over 2^{AP} is a tuple $\mathcal{B} = (Q, Q_0, \Sigma, \to_{\mathcal{B}}, Q_F)$, where Q is the set of states, $Q_0 \subseteq Q$ is the set of initial states, $\Sigma = 2^{\text{AP}}$ is the finite alphabet, $\to_{\mathcal{B}} \subseteq Q \times \Sigma \times Q$ is the transition relation, and $Q_F \subseteq Q$ is the set of accepting states.

Let $\mathcal{B}_\phi$ denote the NBA generated by the LTL formula ϕ. We use $\Delta : Q \times Q \to 2^{\text{AP}}$ to denote the set of atomic propositions that enable transitions of automaton states, i.e., $\forall \sigma \in \Delta(q, q'), q \xrightarrow{\sigma}_{\mathcal{B}\phi} q'$ and $\forall \sigma \notin \Delta(q, q'), q \not\xrightarrow{\sigma}_{\mathcal{B}\phi} q'$. Let $\tau_{\mathcal{B}} = q_0 q_1 q_2 \ldots$ denote a valid infinite run of $\mathcal{B}$, if there exists at least one o_i such that $q_i \xrightarrow{o_i}_{\mathcal{B}\phi} q_{i+1}$. The run $\tau_{\mathcal{B}}$ is called accepting, if it intersects with Q_F infinite often. The word $\tau_o = o_0 o_1 o_2 \ldots, \forall o \in 2^{AP}$ generated from an accepting run satisfies the corresponding LTL formula ϕ. An LTL formula can be translated to an NBA by the tool [22].

At moment t, the observation $\text{obs}(x(t))$ represents the target objects that can be recognized by the robot currently at position $x(t)$ ($x(0) = x_0$) and orientation $\theta(t)$, which is used to update $\mathcal{M}$. The observation $\text{obs}(x(t))$ is stored in a dictionary, where the key represents the object type and the associated key values form a list that stores the possible locations of the object in the environment Env. We use $\text{obj}_i \in \text{AP}$ to denote the type of objects and $N(\text{obj}_i)$ to denote the coordinates of the nearest object obj_i in $\mathcal{M}$ to the current robot position $x(t)$.

B. Problem Formulation

Consider a quadruped robot that resides in an unstructured dynamic environment Env. The robot is equipped with an RGB-D camera and a LiDAR. Initially, the robot only has a preliminary map $\mathcal{M}$ of the environment. The problem to be solved is formally stated as follows.

Problem 1: Given a constrained task ϕ_c and a quadruped robot with the initial position x_0 in Env, the goal is to design a vision-based reactive motion planning strategy such that ϕ_c is ensured to be completed.

Due to the consideration of unstructured dynamic environments, there are two main challenges in solving Problem 1: the *inefficiency challenges* and the *incompleteness challenges*. The inefficiency challenges represent the emergence of a new planning more efficient in accomplishing the task (e.g., fetch the Coke nearby rather than the further one in the fridge). And the incompleteness challenges indicate infeasible subtasks (e.g., the Coke to be fetched is not in the fridge) and thus a modified planning is needed.

Assumption 1: The constrained task ϕ_c can be accomplished regardless how the environment Env changes.

Assumption 1 is mild and reasonable. Otherwise, the task ϕ_c cannot be completed no matter how the motion planning is designed. For cases on how the task requirements can be relaxed when it cannot be completed, we refer readers to works, such as [19], [23].

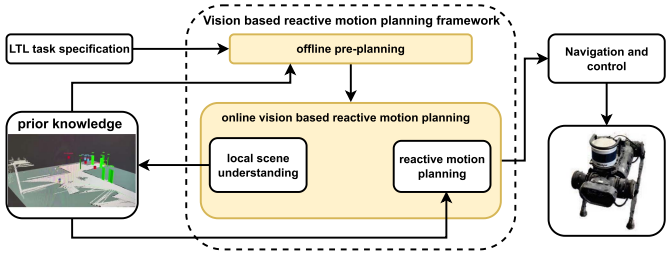

Fig. 2. V-RTLMP.

To solve problem 1, a two-layer planning is developed. The approach overview is shown in Fig. 2. In the following, Section III introduces a preplanning method as a top layer, to generate global trajectories for LTL satisfaction. By taking offline planning as reference trajectories, as the bottom layer, reactive local planning based on sensor fusions is developed in Section IV.

III. Offline Motion Planning

In this section, the robot's prior knowledge of the environment $\mathcal{M}$ and LTL task specification ϕ are used to generate a preplanning. Due to the time-varying labeling functions in the dynamic environment, the target objects are specified using atomic propositions with constraints, such as ϕ_c. Note that ϕ_c can be converted to ϕ by Algorithm 2.

Definition 4: Given the WTS $\mathcal{T}$ and the NBA $\mathcal{B}$, the product Büchi automaton (PBA) is a tuple $P = \mathcal{T} \times \mathcal{B} = (Q_P, Q_P^0, \to_P, Q_P^F, C_P, L_P)$, where $Q_P = X \times Q$ is the set of infinite product states, $Q_P^0 = x_0 \times Q_0$ is the set of initial states; $\to_P \subseteq Q_P \times 2^{\text{AP}} \times Q_P$ is the transition relation defined by the rule: $\frac{x \to_{\mathcal{T}} x' \wedge q \xrightarrow{L_X(x)}_{\mathcal{B}} q'}{q_P = (x,q) \to_P q'_P = (x',q')}$, where $q_P \to_P q'_p$ denotes the transition $(q_P, q'_P) \in \to_P$, $Q_P^F = X \times Q_F$ is the set of accepting states, $C_P : (\to_P) \to \mathbb{R}^+$ is the cost function defined as the cost in the geometric space, e.g., $C_P(q_p = (x,q), q'_p = (x',q')) = C_{\mathcal{T}}(x,x'), \forall (q_P, q'_P) \in \to_P$, and $L_P : Q_P \to \text{AP}$ is the labeling function s.t. $L_P(q_P) = L_X(x), \forall q_P = (x,q)$.

A valid trace $\tau_P = q_P^0 q_P^1 q_P^2 \ldots$ of a PBA is called accepting, if it visits Q_P^F infinitely often. The corresponding accepting word $\tau_o = o_0 o_1 o_2 \ldots, \forall o_i = L_P(q_P^i)$ satisfies the corresponding LTL formula ϕ. Let τ_F denote an accepting trace and $\text{proj}|_X : Q_P \to X$ a function that projects the product state space into the workspace, i.e., $\text{proj}|_X(q_p) = x, \forall q_P = (x,q)$. Using the projection, we can extract a trajectory $\tau_{\mathcal{T}} = \text{proj}|_X(\tau_F)$ that satisfies the LTL formula. More details are presented in [11]. Therefore, the goal of preplanning is to find an accepting path τ_P of PBA, with minimum accumulative geometric cost C_P.

If the the state space of WTS is continuous, it is impossible to explictly construct a PBA. Hence, the sampling-based method TL-RRT* from [24] is employed to track PBA on-the-fly, which can generate the feasible optimal path for LTL satisfaction. The resulting trajectory is a lasso-type sequence in the form of

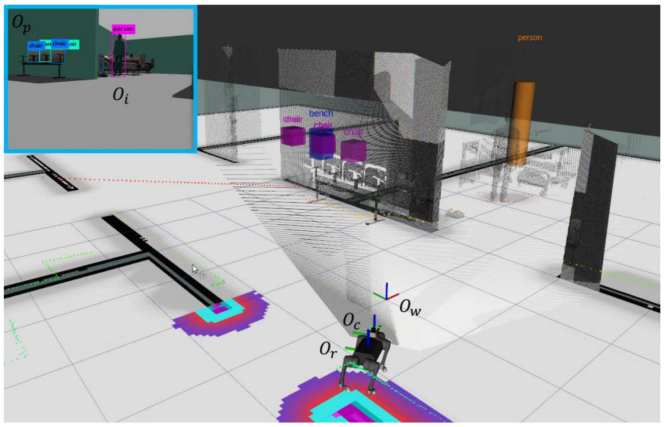

Fig. 3. Robot is visualized under Rviz with the world coordinate system O_w, the robot coordinate system O_r, and the camera coordinate system O_c. The pixel coordinate system O_p and the image coordinate system O_i are zoomed in the upper left corner.

prefix–suffix structure, i.e., $\tau_F = \tau_P^{\text{pre}}[\tau_P^{\text{suf}}]^\omega$, where the prefix part $\tau_P^{\text{pre}} = q_P^0 q_P^1 \dots q_P^K$ is only executed once, and the suffix part $\tau_P^{\text{suf}} = q_P^K q_P^{K+1} \dots q_P^{K+M}$ with $q_P^K = q_P^{K+M}$ is executed infinitely often. Following prior work [3], we can decompose the optimal path $\tau_F = \tau_P^{\text{pre}}[\tau_P^{\text{suf}}]^\omega$ based on automaton components into a sequence of goal-reaching trajectories, i.e., $\tau_F = \tau_0\tau_1 \dots \tau_K[\tau_{K+1} \dots \tau_{K+l}]^\omega$. Each τ_i can be presented as an optimal solution of reachability navigation expressed as a simple LTL formula $\phi_{i,F} = \square\neg\mathcal{O} \wedge \phi_{g_i}$, $\mathcal{O}$ represent obstacles.

In the offline preplanning, the complexity of the process of obtaining the tree is the same as TL-RRT*. The search complexity of the algorithm is $O(V^2)$, where V is the number of vertices of the corresponding automaton. We refer readers for more details about the decomposition procedure in [3].

In the following section, we take each τ_i as a reference path and synthesize a vision-based reactive motion planning algorithm through RGB-camera and LiDAR to satisfy each $\phi_{i,F}$ in dynamic environments.

IV. Online Reactive Motion Planning

This section presents the developed online vision-based reactive motion planning. Section IV-A introduces the local scene understanding system used by the robot to update its prior knowledge $\mathcal{M}$. Section IV-B then introduces the V-RTLMP framework and proposes both greedy mechanisms and gate mechanisms to improve the planning in dynamic environments.

A. Local Scene Understanding

For hardware settings, the RGB-D camera inside the unite A1 quadruped robot is used to acquire both color image and point cloud as input. Let $x(t)$ and $\theta(t)$ denote the robot's position and orientation in the map, respectively. The obtained local observation obs$(x(t))$ is shown in Fig. 3, which is also regarded as the quadruped robot's local scene understanding. There are a total of five-coordinate systems in obs$(x(t))$, which are the world coordinate system O_w, the robot coordinate system O_r, the camera coordinate system O_c, the image coordinate system O_i, and the pixel coordinate system O_p.

Due to the dynamic nature of the environment, it is crucial for the robot to continuously update its prior knowledge in real time and adjust its plans accordingly. Hence, we propose a four-step approach, i.e., detection, projection, localization, and visualization, to enable the robot to update its prior knowledge via local scene understanding.

1) Detection: To extract the high-level representation of the scene from the color image r of the RGB-D camera, the darknet [25] implemented with MindSpore is used to detect the classes of task-relevant objects in the field of view of camera. All 2-D bounding boxes B related to the task are then obtained (e.g., humans, chairs, and benches in the quadruped robot's conical field of view, as shown in the top left corner of Fig. 3). This component corresponds to line 1 of Algorithm 1.

2) Projection: The feature points (u, v) (i.e., the center point in the bounding boxes) are projected to the 3-D space. Specifically, to find the point (x_w, y_w, z_w) in the world coordinate system O_w that corresponds to the point (u, v), the transformation from the camera coordinate system O_c to the pixel coordinate system O_p is established based on the camera model as

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \frac{1}{z_c}\begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} = \frac{1}{z_c}\mathbf{K}\cdot\begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{1}$$

where, $\mathbf{K}$ is the 3×3 internal reference matrix of the camera obtained by camera calibration [26]. In (1), z_c is the depth of the feature point (u, v), which can be obtained by matching the color map and the depth map of RGB-D camera so that we get the (x_c, y_c, z_c) point corresponding to (u, v) in the camera coordinate system O_c. When the robot is moving in the environment Env, the position and orientation of its fixed camera are recorded according to the odometer. Then we can use the robot coordinate system to find the transformation relation between the camera coordinate system O_c and the world (global) reference coordinate system O_w as

$$\begin{bmatrix} x_c \\ y_c \\ z_c \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R_{rc}} & \mathbf{t}_2 \\ \mathbf{0}^\top & 1 \end{bmatrix}\begin{bmatrix} \mathbf{R_{wr}} & \mathbf{t}_1 \\ \mathbf{0}^\top & 1 \end{bmatrix}\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = \mathbf{L}_{rc}\cdot\mathbf{L}_{wr}\cdot\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2}$$

where, $\mathbf{R}_{rc}$ represents the rotation matrix of the robot coordinate system O_r to the camera coordinate system O_c, $\mathbf{t}_2$ represents the translation, and $\mathbf{L}_{rc}$ represents the affine transformation of the robot coordinate system O_r to the camera coordinate system O_c. $\mathbf{R}_{wr}$, $\mathbf{t}_1$, and $\mathbf{L}_{wr}$ represent the rotation, translation, and affine transformation of the world coordinate system O_w to the robot coordinate system O_r. These transformation matrices can be obtained directly from TF in the robot operating system (ROS). Integrating (1) and (2) to get (3), we can directly obtain the coordinates of feature points (u, v) in the world coordinate

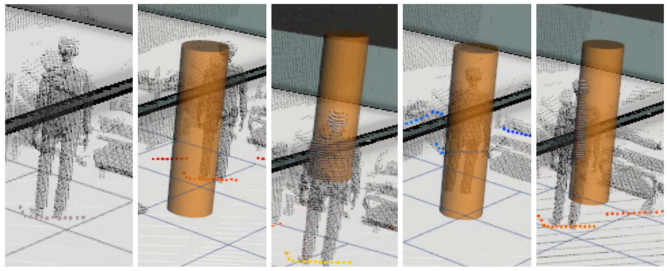

Fig. 4. (a) Human point clouds, and (b)–(d) are the predicted cylinder visualizations based on human's point clouds, where the occupied fractions are (b) 0.98%, (c) 28.59%, (d) 82.24%, and (e) 46.73%, respectively.

Algorithm 1: Function Scene_Understanding().

Input: Color Image r, $\mathbf{L}_{wr}$, $\mathbf{L}_{rc}$, $\mathbf{K}$, Current Moment t_0
Output: Local observation U
1 $B = object\_detection(r)$;
2 **for** *i=0:length(B)* **do**
3 $T = []$;
// T is a temporary list
4 $class = B[i][0], box = B[i][1]$;
5 **for** $(u, v) \in box$ **do**
6 Calculate (x_w, y_w, z_w) in O_w corresponding to (u, v) according to Equ. (3) ;
7 $T.append(x_w, y_w, z_w)$;
8 **end**
9 $p = DBSCAN(T)$;
10 $U[i][0] = class, U[i][1] = p, U[i][2] = t_0$;
11 **end**
12 **return** U;

system (x_w, y_w, z_w) as

$$\begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = z_c \cdot \mathbf{L}_{wr}^{-1} \cdot \mathbf{L}_{rc}^{-1} \cdot \mathbf{K}_1^{\dagger} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}, \mathbf{K}_1 = \begin{bmatrix} \mathbf{K}, \mathbf{0} \end{bmatrix}. \quad (3)$$

3) Localization: The target localization is outlined in lines 2–10 of Algorithm 1. The coordinate transformation (3) is first used to project all points of each bounding box into the world coordinate system to get a point cloud cluster. The point cloud is then segmented and clustered using the density-based spatial clustering of applications with noise [27] so that the centroid p and the respective convex hull dimensions can be obtained. These values are stored in U. The meaning of U is the classes and positions related to the objects within the field of view of the robot in the current scene.

4) Visualization: The previous three steps describe object localization through detection. However, due to the jittering motion of the quadruped robot during movement and the presence of sensor errors, the accuracy of the object localization may not be desired. Thus, several solutions are proposed in this step to improve the prediction accuracy. To optimize memory usage and ensure real-time performance, we created templates for various object classes that can fit the point clouds after clustering. For example, for the human category, we utilized a cylinder with a radius of 0.2 and 2 m high to determine the best position based on the occupancy of the point cloud on the cylinder. We defined the occupancy value in terms of the fraction frac, where $\text{frac} = \frac{n}{100 \times N}\%$, where N refers to the number of point clouds belonging to humans, and n represents the number of point clouds belonging to both humans and cylinders. Generally, the more the point cloud occupies the cylinder's volume after clustering, the more accurate the prediction is. During walking, the quadruped robot can execute Algorithm 1 approximately four times at the same position. Therefore, we select the group with the most significant score t among these four groups as the final prediction. For instance, Fig. 4 illustrates the cylinder's positions corresponding to different occupancy fractions. Typically, the predicted target position is the position of the template with the highest occupancy score. This finalizes the object's positioning within the world coordinate system.

Another issue to be noted is that the position prediction of the target object often drifts when the quadruped robot is walking. For example, a robot with a nonzero pitch angle may affect the prediction in z-direction. This drift is usually caused by a jitter in the robot's gait and is difficult to avoid. When predicting a static target object, the jitter may cause the position prediction to change abruptly in a short period, which will treat the static object as a moving target object. Thus, it is necessary to update the robot's prior knowledge constantly. This erroneous knowledge will likely cause error accumulation and lead to the failure of subsequent subtasks. To solve this problem, we made the following two improvements. First, we set a threshold value according to the movement speed of dynamic objects in the environment. If the drift is higher than this threshold, the object is considered as a dynamic object. Otherwise, it is considered as a drift caused by robot bumps or sensor errors. In this case, the robot treats the target object as a static object, and a reprediction of the target object's position is needed. Second, we improve the tolerance of the gate mechanism and consider the subtask completed when the distance to the target object is less than 0.5 m. The abovementioned solution can significantly improve the accuracy of predicting the target object and distinguish dynamic objects from drift caused by bumps or sensor errors by thresholding, effectively preventing the vision-based reactive planning from falling into a dead loop.

B. Vision-Based Reactive Motion Planning

In this section, we first provide a comprehensive overview of the V-RTLMP framework, and then explain in detail how it tackles the challenges of unstructured dynamic environments. Given an LTL task specification ϕ corresponding to the constrained task ϕ_c, the labeling function L_X^B is used to record the position constraint of the target objects. We can generate offline trajectories π according to Section III that satisfies ϕ. Then, the trajectory π can be projected onto $\mathcal{T}$ to obtain a list of the target object's location locate that needs to be accessed, along with the

Algorithm 2: Task Translate.

Input: constrained task ϕ_c, prior knowledge $\mathcal{M}$
Output: LTL task ϕ
foreach $ap^c \in \phi_c$ **do**
  Record $L_X^B(ap) = c$;
  Replace ap^c in ϕ_c with ap;
end
$\phi = \phi_c$;
return ϕ;

corresponding list of position constraints bind. (lines 1–10 in Algorithm 3). In the following, we show how to adjust offline reference trajectories for dynamic environments using an online vision-based reactive motion planning module.

As shown in lines 11–29 of Algorithm 3, the robot keeps exploring the environment until the task has been completed, while the vision system constantly updates the robot's prior knowledge $\mathcal{M}$ of the environment Env. Two mechanisms, i.e., the greedy and the gate mechanisms, are introduced to deal with the *inefficiency challenges* and *incompleteness challenges* in Section II-B.

The greedy mechanism deals with two main types of situations. In the first situation, when the robot intended to go to locate$[i]$ to execute the subtask x_i, it may find a new location $x^\star$ that satisfies dist(locate$[i], x(t)) \geq$ dist$(x^\star, x(t)) \wedge x^\star \in$ bind$[i] \wedge L_X(x^\star) = x_i$. At this point, rather than continue with the previous goal locate$[i]$, $x^\star$ will be selected instead as the new goal (lines 14–17 in Algorithm 3). Another situation is that the current subtask cannot be completed, and the robot's prior knowledge does not provide valid information. In this case, the robot goes to the vicinity of the next subtask to find the current target object, and if it finds a solution to complete the current subtask on the way, it can execute the subtask (lines 27–29 in Algorithm 3).

The gate mechanism: when the intended location of the subtask locate$[i]$ is reached, the subtask may not be completed due to the change of environment, and it is necessary to find a new target point $N(x_i)$ based on $\mathcal{M}$. This point is the closest point to the robot that can satisfy the requirement bind$[i]$ (lines 21–26 in Algorithm 3). This is when environmental changes make the robot less efficient at completing tasks. Combining the abovementioned steps, we can obtain the optimal trajectory $\tau*$ that satisfies the task ϕ_c on $\mathcal{T}$. It is easy to see that this reaction process is not on the automaton, so it is unnecessary to reconstruct the automaton or research such a time-consuming and labor-intensive operation when a dynamic response occurs in the environment. Thanks to the target object rather than the position in the environment used for the LTL task description, the robot can choose how to carry out reactive planning according to the task autonomously.

The complexity of online planning is $O(n\log n)$, where n is the number of all objects related to the task in the $\mathcal{M}$.

Theorem 1: Under Assumption 1, there exist $\tau^* = \tau_0\tau_1 \ldots \tau_K[\tau_{K+1} \ldots \tau_{K+l}]^\omega$ which satisfies ϕ_c and is the local optimal trajectory for each subtask $\tau_i, \forall i \in \{0, 1, \ldots K + l\}$ in the presence of local observations of the environment.

Proof: At the initial position, the robot's prior knowledge of the environment is $\mathcal{M}$ and the LTL task specification with location constraints is ϕ_c. According to Assumption 1, we know that the task can be completed at this moment. Therefore, we translate ϕ_c into the LTL task specification ϕ through Algorithm 2. Through the sample-based search algorithm TL-RRT* [24] and [3] in Section III, we can search for a path $\tau_F = \tau_0\tau_1 \ldots \tau_K[\tau_{K+1} \ldots \tau_{K+l}]^\omega$ in the PBA for global LTL satisfaction, where each τ_i can be presented as an optimal solution of subreachability task expressed as a simple LTL formula $\phi_{i,F} = \square\neg\mathcal{O} \wedge \phi_{g_i}$, $\mathcal{O}$ represent obstacles.

For any local path τ_i of a subtask $\phi_{i,F}$, the destination is the location of time-varying target. The map is updated based on real-time sensor fusion. If the robot detect the map changing during execution, the replan module is activated to generate local optimal paths using RRT* or A* [28] to satisfy $\phi_{i,F}$. ■

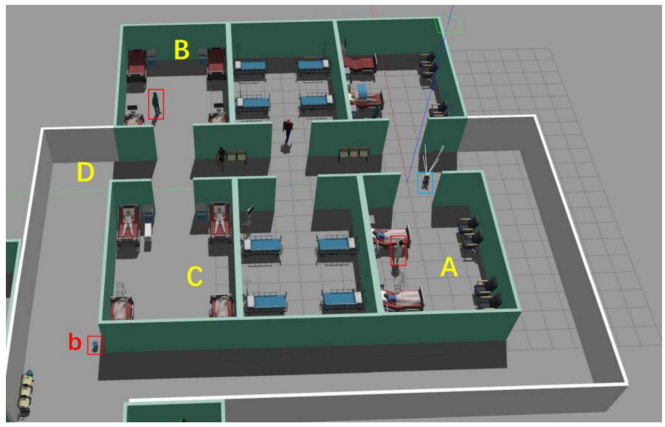

Fig. 5. Quadruped robot is asked to perform the task ϕ_c in a hospital environment. The robot needs to find a nurse to get the prescription in room A and delivers it to the doctor in room B. After taking the medicine, the medicine box should be dropped in the trash bin b in the corridor D and the medicine should be returned to the nurse in room A.

V. Simulation and Experiment

In this section, we experimentally validate the effectiveness of V-RTLMP framework in simulation and real-world environments.

A. Simulation

A simulated hospital environment is established in Gazebo [31], as shown in Fig. 5. To accommodate the dynamic and unstructured nature of the hospital environment, we consider mobile target objects. For instance, doctors may traverse from room A to room B during patient rounds, and pedestrians may be present in the corridors. In such dynamic and unstructured environments, two scenarios can arise. First, there might exist alternative approaches that can outperform the offline preplanning methods, enabling robots to accomplish tasks with greater

Algorithm 3: V-RTLMP.

Input: prior knowledge $\mathcal{M}$, constrained task ϕ_c, robot initial position l_0, WTS $\mathcal{T}$.
Output: trace τ^*

Convert ϕ_c to ϕ according to Alg. 2;
Convert ϕ to NBA $\mathcal{B}$;
Product $\mathcal{T}$ and $\mathcal{B}$ to get P;
Get $\tau_F = \tau_P^{pre}[\tau_P^{suf}]^{\omega}$ according to TL-RRT* [24];
Reformulate the trajectory into the modular form $\tau_P^{pre} = q_P^0 q_P^1 \dots q_P^K$;
for i = *0: K* **do**
  $x_i = proj|_X(q_p^i)$;
  $locate.append(x_i)$; // location list
  $bind.append(L_X^B(x_i))$; // constrain list
end
for i = *0: length(object)* **do**
  $subtask = x_i$;
  while $x(t) \neq locate[i]$ **do**
    Navigate towards $locate[i]$ by [29];
    Update $\mathcal{M}$ by U = scene_understanding() in Alg. 1;
    if $subtask \in U[:,0] \wedge N(subtask) \in bind[i] \wedge N(subtask) \leq dist(locate[i], x(t))$ **then**
      $locate[i] = N(subtask)$; // greedy mechanism
    end
    acquire $x(t)$ according to Lio-sam [30];
  end
  if $subtask \neq L_X(locate[i])$ // gate mechanism **then**
    Delete $locate[i]$ in $\mathcal{M}[subtask][:]$;
    if $\exists j.st.\mathcal{M}[subtask][j] \in bind[i]$ **then**
      find $j^\star = \arg\min_{j \in \mathbb{Z}^\star} \mathcal{M}[subtask][j] \in bind[i]$;
      $locate[i] = \mathcal{M}[subtask][j^\star]$;
    else
      $locate[i] = N(x_{i+1})$; // greedy mechanism
    end
  end
  i -= 1;
end
$\tau^* = x_0 x_1 ... x_K$;
Return τ^*;

efficiency. Hence, it is crucial for the robot to possess a keen understanding of how the dynamic environment influences the task while executing subtasks. This awareness empowers the robot to select appropriate actions in every subtask, thereby enhancing its ability to effectively achieve the desired goal. Second, there are situations where tasks cannot be completed by following the preplan due to the dynamic environment. In this case, the robot needs to evaluate the impact of the dynamic environment on the task in real-time and replans when the subtask cannot be completed. The simulation was performed on a computer equipped with an Intel Core i9 20×3.7 GHz, 64 GB RAM, and NVIDIA GeForce RTX 3090. The Unitree A1 quadruped robot is introduced to the simulated environment, and the MIT Mini Cheetah's convex model predictive controller [32], [33] is used for the control of the quadruped robot. The quadruped robot is equipped with an RGB-D camera and a single-line LiDAR on the back. We rely on the ROS for the system information transfer.

The constrained task is specified as $\phi_c = \square\lozenge \text{nurse}^{\text{Room}_A} \wedge \square\lozenge \text{doctor}^{\text{Room}_B \vee \text{Room}_C} \wedge \square\lozenge \text{can} \wedge \square\lozenge \text{nurse}^{\text{Room}_A}$, where nurse represent nurses, doctor represents doctors, and can represents trash cans. The superscript represents position constraints, and empty means no constraints. The task ϕ_c requires the quadruped robot to go to a room A to find the nurse to consult the patient's condition and then go to room B or room C to find a doctor and get medicine. Before bringing the medicine to the nurse in room A, it needs to throw the box of medicine into a random trash can (the order is not reflected in the formula because we used the default order of the LTL2BA tool).

In Fig. 5, the locations of target objects where the robot performs its task are marked with red boxes. Given the current environment, a feasible solution to ϕ_c is to go from room A to a room B, then goes to the trash can b, in the corridor D to throw the trash, and finally return to room A. If Env is dynamic unstructured, as shown in Fig. 5, the doctor may walk from room B to room C, and there might exist other trash cans a in the corridor D. Hence, the robot may not find the doctor when arriving at room B. This will trigger the gate mechanism. Specifically, the current subtask is considered as not completed yet, even if the robot arrives at room B as preplanned. The gate mechanism declines the state transition in NBA $\mathcal{B}$, as shown in Fig. 6(b). Since the subtask of finding a doctor to take medicine could not be completed as desired, the robot needs to replan its motion by going to room C to look for the doctor. When reaching room C and finding the doctor, the gate mechanism confirms the completion of the current subtask and the task, then ϕ_c proceeds, as shown in the state transition in Fig. 6(c). In Fig. 6(d), the robot finds another trash can a on its way to the trash can b. Since can a is closer than can b to its current location, the greedy mechanism kicks in, allowing the robot to choose the trash can a instead of can b throwing the medicine package. After the medicine package has been trashed at can a, the gate mechanism confirms the completion of task can and proceeds to the last subtask. More details are referred to in the experiment video.[1]

B. Experiment

The Unitree A1 quadruped robot experiment was conducted, as shown in Fig. 7(a). In a dynamic and unstructured environment, the locations of entities, such as people, mugs, and bags may change. For example, the person might walk from location A to B or relocate a package from location C to A. These variations make the robot's preplanned trajectories based on prior knowledge inefficient or ineffective. Therefore, robots need to plan reactively by constantly updating their real-time understanding of the environment to ensure the task's successful completion. The quadruped robot is equipped with an RGB-D camera and a VLP-16 3 d LiDAR scanner on the back [see Fig. 7(b)]. Unlike the simulation that relies on the odometer for the localization in the environment, here Lio-sam [30] was employed to determine the robot's position by LiDAR point cloud matching. The experiments were deployed on a TX2 on

[1][Online]. Available: https://youtu.be/BKQ6SyD0oDw

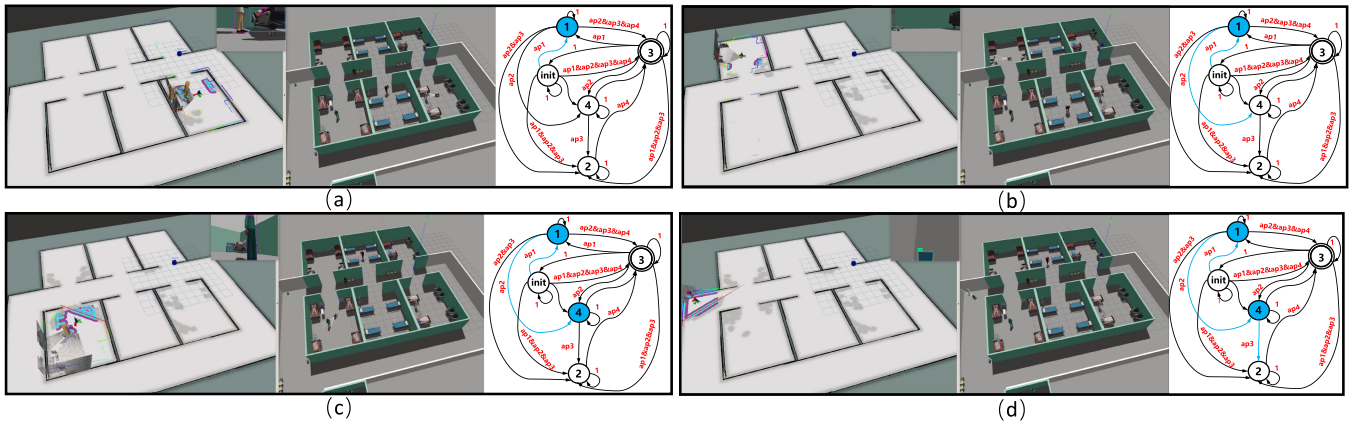

Fig. 6. Visualization of the task ϕ_c, where the $api\,(i = 1, 2, 3, 4)$ in the transition graph represents different subtasks of ϕ_c. (a) Successful completion of nurse$^{\text{room}_A}(ap1)$. (b) Gate mechanism declines the state transition in NBA $\mathcal{B}$. (c) Successful completion of doctor$^{\text{Room}_B \vee \text{Room}_C}(ap2)$. (d) Successful completion of can$(ap3)$.

Fig. 7. Experiment setup. (a) Initial environment. (b) Unitree A1 quadruped robot. (c) Visualized robot trajectory.

the quadruped robot and a laptop with an Intel i7-11800H and RTX3050Ti NVIDIA card. The TX2 is mainly responsible for robot motion control, navigation, and obstacle avoidance, while the laptop is used for the V-RTLMP algorithm. Communication between devices is also via ROS.

In the experiment, the constrained task is $\phi_c = \diamondsuit(\text{water} \wedge \bigcirc(\diamondsuit(\text{bag} \wedge \bigcirc(\diamondsuit \text{human}^{B \vee A})))) \vee \diamondsuit(\text{bag} \wedge \bigcirc(\diamondsuit(\text{water} \wedge \bigcirc(\diamondsuit \text{human}^{B \vee A}))))$, where water represents the water bottle, bag represents the school bag, and human represents the person. The task ϕ_c requires the quadruped robot first to find the water bottle and school bag and then hand them to the human, who might be in either area A or area B. Similarly, ϕ_c is first translated to an LTL task specification $\phi = \diamondsuit(\text{water} \wedge \bigcirc(\diamondsuit(\text{bag} \wedge \bigcirc(\diamondsuit \text{human})))) \vee \diamondsuit(\text{bag} \wedge \bigcirc(\diamondsuit(\text{water} \wedge \bigcirc(\diamondsuit \text{human}))))$ according to Algorithm 2 for offline preplanning. As shown in Fig. 7(c), since the target objects are out of the robot's field of view, neither the school bag nor the water bottle can be observed by the robot at the initial position 1. The robot goes to area C and area D sequentially according to the preplanned trajectory. Unfortunately, none of the target objects can be observed, and thus the gate mechanism determines that the subtask cannot be completed. Hence, the robot needs to explore the environment to find another way to complete the subtask. Since no position constraints are associated with the water bottle and the school bag and no relevant information is available in its prior knowledge, the greedy mechanism is activated to guide the robot toward the next preplanned position. Later, at the corner, the robot finds the school bag and water bottle in the area of A. Then the robot goes to area A to complete the task of packing the school bag and water bottle. After the gate mechanism confirms that two subtasks have been finished, the robot goes to area B to hand the water and the school bag to the human. More details are referred to in the experiment video.[2]

VI. Conclusion

This work presented a V-RTLMP framework in unstructured dynamic environments. The effectiveness of our approach was demonstrated via numerical simulation and physical experiments. Future work will investigate heterogeneous multirobot systems for more challenging collaborative tasks.

[2][Online]. Available: https://youtu.be/He1-lEn3ot4

References

[1] D. Tian et al., "Two-phase motion planning under signal temporal logic specifications in partially unknown environments," *IEEE Trans. Ind. Electron.*, vol. 70, no. 7, pp. 7113–7121, Jul. 2023.

[2] M. Cai, M. Hasanbeig, S. Xiao, A. Abate, and Z. Kan, "Modular deep reinforcement learning for continuous motion planning with temporal logic," *IEEE Robot. Autom. Lett.*, vol. 6, no. 4, pp. 7973–7980, Oct. 2021, doi: 10.1109/LRA.2021.3101544.

[3] M. Cai, E. Aasi, C. Belta, and C.-I. Vasile, "Overcoming exploration: Deep reinforcement learning for continuous control in cluttered environments from temporal logic specifications," *IEEE Robot. Automat. Lett.*, vol. 8, no. 4, pp. 2158–2165, Apr. 2023.

[4] G. Chen, M. Liu, and Z. Kong, "Temporal-logic-based semantic fault diagnosis with time-series data from industrial Internet of Things," *IEEE Trans. Ind. Electron.*, vol. 68, no. 5, pp. 4393–4403, May 2021.

[5] X. Zhou, T. Yang, Y. Zou, S. Li, and H. Fang, "Multiple sub-formulae cooperative control for multi-agent systems under conflicting signal temporal logic tasks," *IEEE Trans. Ind. Electron.*, vol. 70, no. 9, pp. 9357–9367, Sep. 2023.

[6] Y. Li, R. Cui, Z. Li, and D. Xu, "Neural network approximation based near-optimal motion planning with kinodynamic constraints using RRT," *IEEE Trans. Ind. Electron.*, vol. 65, no. 11, pp. 8718–8729, Nov. 2018.

[7] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, "Formal approach to the deployment of distributed robotic teams," *IEEE Trans Rob.*, vol. 28, no. 1, pp. 158–171, Feb. 2012.

[8] H. Kress-Gazit, G. E. Fainekos, and G. J. Pappas, "Temporal-logic-based reactive mission and motion planning," *IEEE Trans Rob.*, vol. 25, no. 6, pp. 1370–1381, Dec. 2009.

[9] S. Li, D. Park, Y. Sung, J. A. Shah, and N. Roy, "Reactive task and motion planning under temporal logic specifications," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2021, pp. 12618–12624.

[10] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible LTL constraints," *IEEE Trans. Autom. Control*, vol. 68, no. 1, pp. 301–316, Jan. 2023.

[11] C. Baier and J.-P. Katoen, *Principles of Model Checking*. Cambridge, MA, USA: MIT Press, 2008.

[12] V. Vasilopoulos, W. Vega-Brown, O. Arslan, N. Roy, and D. E. Koditschek, "Sensor-based reactive symbolic planning in partially known environments," in *Proc. IEEE Int. Conf. Robot. Autom.*, 2018, pp. 5683–5690.

[13] M. Guo and D. V. Dimarogonas, "Multi-agent plan reconfiguration under local LTL specifications," *Int. J. Robot. Res.*, vol. 34, no. 2, pp. 218–235, 2015.

[14] M. Lahijanian, M. R. Maly, D. Fried, L. E. Kavraki, H. Kress-Gazit, and M. Y. Vardi, "Iterative temporal planning in uncertain environments with partial satisfaction guarantees," *IEEE Trans. Rob.*, vol. 32, no. 3, pp. 583–599, Jun. 2016.

[15] C. I. Vasile, X. Li, and C. Belta, "Reactive sampling-based path planning with temporal logic specifications," *Int. J. Robot. Res.*, 2020, Art. no. 0278364920918919.

[16] Y. Li, E. M. Shahrivar, and J. Liu, "Safe linear temporal logic motion planning in dynamic environments," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, 2021, pp. 9818–9825.

[17] A. Ulusoy and C. Belta, "Receding horizon temporal logic control in dynamic environments," *Int. J. Robot. Res.*, vol. 33, no. 12, pp. 1593–1607, 2014.

[18] M. Cai, H. Peng, Z. Li, H. Gao, and Z. Kan, "Receding horizon control based motion planning with partially infeasible LTL constrains," *IEEE Control Syst. Lett.*, vol. 5, no. 4, pp. 1279–1284, Oct. 2021.

[19] Z. Li, M. Cai, S. Xiao, and Z. Kan, "Online motion planning with soft metric interval temporal logic in unknown dynamic environment," *IEEE Control Syst. Lett.*, vol. 6, pp. 2293–2298, 2022, doi: 10.1109/LCSYS.2022.3145058.

[20] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *Int. J. Robot. Res.*, vol. 30, no. 7, pp. 846–894, 2011.

[21] A. Rosinol, M. Abate, Y. Chang, and L. Carlone, "Kimera: An open-source library for real-time metric-semantic localization and mapping," in, *Proc. IEEE Int. Conf. Robot. Autom.*, 2020, pp. 1689–1696.

[22] P. Gastin and D. Oddoux, "Fast LTL to Büchi automata translation," in *Proc. Int. Conf. Comp. Aid. Ver.* 2001, pp. 53–65.

[23] Z. Zhou, D. J. Lee, Y. Yoshinaga, S. Balakirsky, D. Guo, and Y. Zhao, "Reactive task allocation and planning for quadrupedal and wheeled robot teaming," in *Proc. IEEE 18th Int. Conf. Automat. Sci. Eng. (CASE)*, 2022, pp. 2110–2117.

[24] X. Luo, Y. Kantaros, and M. M. Zavlanos, "An abstraction-free method for multirobot temporal logic optimal control synthesis," *IEEE Trans. Rob.*, vol. 37, no. 5, pp. 1487–1507, Oct. 2021.

[25] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020, *arXiv:2004.10934*.

[26] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 22, no. 11, pp. 1330–1334, Nov. 2000.

[27] M. Ester, H.-P. Kriegel, J. Sander, and X. Xu, "A density-based algorithm for discovering clusters in large spatial databases with noise," *Proc. Int. Conf. Knowl. Dis. Data Mining*, vol. 96, no. 34, pp. 226–231, 1996.

[28] S. M. Persson and I. Sharf, "Sampling-based A* algorithm for robot path-planning," *Int. J. Robot. Res.*, vol. 33, no. 13, pp. 1683–1708, 2014.

[29] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robot. Autom. Mag.*, vol. 4, no. 1, pp. 23–33, Mar. 1997.

[30] T. Shan, B. Englot, D. Meyers, W. Wang, C. Ratti, and D. Rus, "LIO-SAM: Tightly-coupled LiDAR inertial odometry via smoothing and mapping," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2020, pp. 5135–5142.

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

[32] D. Kim, J. Di Carlo, B. Katz, G. Bledt, and S. Kim, "Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control," 2019, *arXiv:1909.06586*.

[33] J. Di Carlo, P. M. Wensing, B. Katz, G. Bledt, and S. Kim, "Dynamic locomotion in the MIT Cheetah 3 through convex model-predictive control," in *Proc. IEEE/RSJ Int. Conf. Intel. Robot. Syst.*, 2018, pp. 1–9.

Zhangli Zhou received the B.E. degree in mechanical engineering from the Department of Mechanical Engineering, Wuhan University of Technology, Wuhan, China, in 2016. He is currently working toward the Ph.D. degree in control engineering with the University of Science and Technology of China, Hefei, China.

His research interests include robotics and reactive task and motion planning.

Ziyang Chen received the B.E. degree in control engineering from the Department of Automation, Beihang University, Beijing, China, in 2020. He is currently working toward the Ph.D. degree in control science and engineering with the University of Science and Technology of China, Hefei, China.

His research interests include robotics and multiagent systems.

Mingyu Cai received the Ph.D. degree in mechanical engineering from the University of Iowa, Iowa City, IA, USA, in 2021.

He is an Assistant Professor with the University of California, Riverside, CA, USA. He is a Research Scientist with Honda Research Institute, San Jose, CA, USA. From 2021, he has been a Postdoctoral Associate with the Department of Mechanical Engineering, Lehigh University, Bethlehem, PA, USA. His research interests encompass robotics, machine learning, control theory, and formal methods, with a focus on applications in motion planning, decision-making, nonlinear control, and autonomous driving.

Zhijun Li (Fellow, IEEE) received the Ph.D. degree in mechatronics from Shanghai Jiao Tong University, Shanghai, China, in 2002.

Since 2017, he has been a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China, where he has been the Vice Dean of the School of Information Science and Technology, since 2019. From 2003 to 2005, he was a Postdoctoral Fellow with the Department of Mechanical Engineering and Intelligent Systems, The University of Electro-Communications, Chofu, Japan. From 2005 to 2006, he was a Research Fellow with the Department of Electrical and Computer Engineering, National University of Singapore, and Nanyang Technological University, Singapore. His research interests include wearable robotics, biomechatronics systems, nonlinear control, and computational optimization.

Prof. Li has been the Co-Chair of IEEE SMC Technical Committee on Bio-Mechatronics and Bio-Robotics Systems, and IEEE RAS Technical Committee on Neuro-Robotics Systems. He is a Fellow of the AAIA. He is a Member of Board of Governors, IEEE Systems, Man and Cybernetics Society(2023-2025), and Associate Editor for several IEEE transactions.

Zhen Kan (Senior Member, IEEE) received the Ph.D. degree in mechanical engineering from the Department of Mechanical and Aerospace Engineering, University of Florida, Gainesville, FL, USA, in 2011.

He is currently a Professor with the Department of Automation, University of Science and Technology of China, Hefei, China. From 2012 to 2016, he was a Postdoctoral Research Fellow with the Air Force Research Laboratory, Eglin AFB, and the University of Florida Research and Engineering Education Facility, and was an Assistant Professor with the Department of Mechanical Engineering, University of Iowa from 2016 to 2019. His research interests include networked control systems, nonlinear control, formal methods, and robotics.

Dr. Kan currently serves on program committees of several internationally recognized scientific and engineering conferences and is an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL.

Chun-Yi Su (Senior Member, IEEE) received the Ph.D. degree in control engineering from the South China University of Technology, Guangzhou, China, in 1990.

He is currently with the School of Intelligent Manufacturing, Taizhou University, Taizhou, China, on leave from Concordia University. In 1998, he joined Concordia University, Montreal, QC, Canada, after a seven-year stint with the University of Victoria, Victoria, BC, Canada. He has authored or coauthored more than 500 publications in journals, book chapters, and conference proceedings. His research interests include control theory and its applications to various mechanical systems.

Dr. Su was an Associate Editor for the IEEE TRANSACTIONS ON AUTOMATIC CONTROL and the IEEE TRANSACTIONS ON CONTROL SYSTEMS TECHNOLOGY. He has been on the editorial board of several journals, including *Mechatronics (IFAC)*. He is a Distinguished Lecturer of the IEEE RA Society.