STL-Constrained Multi-Robot Trajectory Planning via Constrained Bayesian Optimization and Local Cost Map Learning

***Abstract*— We address multi-robot motion planning under Signal Temporal Logic (STL) specifications with kinodynamic constraints. Exact approaches face scalability bottlenecks and limited adaptability, while conventional sampling-based methods require excessive samples to construct optimal trajectories. We propose a two-stage framework integrating sampling-based online learning with formal STL reasoning. At the single-robot level, our constrained Bayesian Optimization-based Tree search (cBOT) planner uses Gaussian process as surrogate models to learn local cost maps and feasibility constraints, generating shorter collision-free trajectories with fewer samples. At the multi-robot level, our STL-enhanced Kinodynamic Conflict-Based Search (STL-KCBS) algorithm incorporates STL monitoring into conflict detection and resolution, ensuring specification satisfaction while maintaining scalability and probabilistic completeness. Benchmarking demonstrates improved trajectory efficiency and safety over existing methods. Real-world experiments with autonomous surface vehicles validate robustness and practical applicability in uncertain environments.The STL-cBOT Planner will be released as an open-source package and videos of real-world and simulated experiments are available at https://stlbot.github.io/.**

I. Introduction

A central challenge in multi-robot systems lies in generating dynamically feasible trajectories that permit robots to simultaneously reach their target regions without colliding with obstacles or with one another. Conventional methods typically rely on explicitly encoding large sets of low-level constraints, leading to brittle solutions with limited adaptability to real-world conditions. Signal Temporal Logic (STL) [1], by contrast, offers a powerful declarative formalism that enables high-level specification of temporal and logical requirements, supporting sophisticated behaviors such as International Regulations for Preventing Collisions at Sea (COLREGs), coordinated motion through intersections, and adaptive responses to environmental changes.

Early STL-based planning approaches frequently employed model predictive control (MPC)[2, 3, 4], which proved effective but computationally expensive when tackling complex formulas and long horizons. To improve scalability, STL constraints have been incorporated into sampling-based planners such as RRT* and its real-time extensions, with robustness metrics guiding the search toward specification-compliant solutions[5]. Multi-layered frameworks have since integrated kinodynamic constraints and STL through biased exploration and robustness-driven costs, enabling sequential satisfaction of tasks with greater efficiency [6]. More recently, reinforcement learning has been combined with STL semantics to address the reward-shaping problem, producing task-aware and robust policies [7, 8]. These efforts highlight STL's potential in single-agent contexts, but extending such capabilities to multi-robot systems introduces substantial new challenges.

Scaling STL to multi-robot motion planning (MRMP) requires addressing task allocation, inter-agent coordination, and nonlinear or nonholonomic dynamics [9, 10]. Centralized approaches often encode multi-agent STL specifications as mixed-integer linear programs (MILP), allowing implicit task assignment and collision avoidance [9, 11]. While effective, MILP formulations face scalability bottlenecks inherent to their solvers. Decentralized alternatives have thus emerged: barrier-function-based feedback controllers ensure continuous-time STL satisfaction across agents [12, 13], while distributed MPC relaxes infeasible constraints to guarantee recursive feasibility under conflicting tasks [14]. Complementing these methods, differentiable STL robustness has enabled graph neural planners to handle heterogeneous teams and cluttered environments with up to 32 robots [10], offering improved scalability over centralized MPC baselines.

Beyond STL-specific work, broader MRMP research provides valuable insights through multi-agent path finding (MAPF) extensions [15, 16, 17] and sampling-based methods that offer flexibility through collision checking alone. While coupled approaches explore joint state spaces, their synchronized action requirements restrict time optimality, and spatiotemporal variants like Time-Based RRT [18], Temporal PRM [19], and ST-RRT* [20] enhance temporal reasoning but struggle in narrow corridors, limiting scalability. Graphs of Convex Sets (GCS) provide a geometry-driven alternative by constructing collision-free convex regions directly, with applications in UAV navigation, non-Euclidean motion planning, and temporal-logic tasks [21, 22, 23], while multi-robot extensions show efficiency improvements [24, 25] despite dynamic coordination challenges that motivate developments like ST-GCS [26]. Kinodynamic Conflict-Based Search (K-CBS) [27] decomposes planning into high-level coordination and low-level trajectory generation, achieving probabilistic completeness and scalability but struggling with path optimization due to RRT-based planning and cost map neglect. Though not designed for STL, these methods could be augmented with temporal-logic constraints, bridging discrete conflict resolution and decentralized STL planning. Table I summarizes a qualitative comparison of the related methods.

Contributions. The main contributions of this paper are as follows: (i) We propose a constrained Bayesian optimization–based tree search (cBOT) algorithm for computing collision-free trajectories for individual robots. The cBOT framework incorporates a local cost map modeled with

Method	Multirobot	STL Spec	Learning	Kinodynamic Constraints
MILP [9]	✓	✓	×	×
STL-RRT* [20, 6]	×	✓	×	×
ST-GCS [26]	✓	✓	×	×
MPC [4]	×	✓	✓	×
GCBF [10]	✓	✓	✓	×
K-CBS [27]	✓	×	×	✓
Ours	✓	✓	✓	✓

TABLE I: Qualitative comparison of the related methods.

a Gaussian process (GP) and learns individual constraints using separate GPs. This enables the planner to generate shorter trajectories with fewer samples compared to conventional RRT-based approaches. (ii) We introduce the STL-enhanced Kinodynamic Conflict-Based Search (STL-KCBS) algorithm, which efficiently computes multi-robot trajectories by combining the decoupled, sampling-based cBOT planner with a conflict tree. STL-KCBS inherits the decentralized and scalable properties of the K-CBS framework [27], while also providing probabilistic completeness guarantees. (iii) We present a comprehensive benchmarking study against existing approaches, demonstrating the advantages of the proposed STL-KCBS planner. A qualitative comparison of related methods is summarized in Table I. (iv) We validate the real-world applicability of the STL-KCBS planner through experiments with autonomous surface vehicles operating in a lake environment.

II. Problem Formulation

Consider a team of N robots in a d-dimensional workspace $W \subset \mathbb{R}^d$, with the free space defined as $W_{\text{f}} = X_{\text{f}} = X \backslash X_{\text{obs}}$, where $X_{\text{obs}} \subset X \subseteq \mathbb{R}^d$ represents obstacles. Each robot is initialized at a designated position $\mathbf{x}_i^{\text{init}} \in W$, $\Pi_i(\mathbf{x}_{i,\text{init}}) = p_i^{\text{init}}$; $\Pi_i : X_i \to W$ is the projection function that maps the robot's state into its position. The team's navigation objective is specified by the multi-agent signal temporal logic (MA-STL) formula Ψ (cf. Definition 2), which encodes desired collective behaviors over time. Given a real-valued function $\mu : W \to \mathbb{R}$, an atomic predicate is defined as π^μ, where a point $\mathbf{x} \in W$ satisfies $\mathbf{x} \models \pi^\mu$ if $\mu(\mathbf{x}) \geq 0$.

We present notations used throughout: $\mathbb{R}$, $\mathbb{R}_{\geq 0}$, and $\mathbb{N}$ denote the sets of real, non-negative real, and natural numbers, respectively. We write $\mathbb{R}^n$ for the n-dimensional Euclidean space and $\mathbb{R}^{n\times m}$ for the space of real matrices with n rows and m columns. For $a \leq b$, $[a, b]$ denotes a time interval, and $t + [a, b] := [t + a, t + b]$. The set $\textsc{Fun}(X, Y)$ denotes all functions from X to Y.

Definition 1 (Signal Temporal Logic (STL) Semantics): Let $M \subset \mathbb{R}^d$, $S = \textsc{Fun}(\mathbb{R}_{\geq 0}, M)$ denotes the set of signals, and $F = \textsc{Fun}(\mathbb{R}^d, \mathbb{R})$ the set of real-valued functions from $\mathbb{R}^d$. The syntax of STL is given by [9]:

$$\varphi ::= \top \mid \pi^\mu \mid \neg\varphi \mid \varphi_1 \wedge \varphi_2 \mid \varphi_1 \,\mathcal{U}_{[a,b]}\, \varphi_2,$$

where $\top$ is $\textsc{True}$ and $\mu \in F$.

The Boolean semantics $(s, t) \models \varphi$ are defined recursively:

$$\begin{aligned}
(s,t) &\models \top \Leftrightarrow \ \textsc{True},\\
(s,t) &\models \pi^\mu \Leftrightarrow \mu(s(t)) \geq 0,\\
(s,t) &\models \neg\varphi \Leftrightarrow \neg\big((s,t) \models \varphi\big),\\
(s,t) &\models \varphi_1 \wedge \varphi_2 \Leftrightarrow (s,t) \models \varphi_1 \ \wedge \ (s,t) \models \varphi_2,\\
(s,t) &\models \Diamond_{[a,b]}\varphi \Leftrightarrow \forall\tau \in t + [a,b], (s,\tau) \models \varphi\\
(s,t) &\models \Box_{[a,b]}\varphi \Leftrightarrow \exists\tau \in t + [a,b], (s,\tau) \models \varphi
\end{aligned}$$

The operators $\Box$ and $\Diamond$ are called "always" and "eventually" respectively. Such operators permit to consider more tangible predicates such as "always remain within bounds" or "eventually reach a target".

Definition 2 (MA-STL Formula): A multi-agent Signal Temporal Logic formula Ψ extends Signal Temporal Logic to multiple robots [9], encapsulating the desired collective behavior. Similarly, it is defined recursively:

$$\Psi := \pi_i^\varphi \mid \Psi_1 \wedge \Psi_2 \mid \Psi_1 \vee \Psi_2, \tag{1}$$

where Ψ_1, Ψ_2 are N-robot STL formulas, and π_i^φ assigns a single-robot STL specification φ to robot i. Formally, the validity of an MA-STL formula with respect to trajectories $(\mathbf{s}_1, \ldots, \mathbf{s}_N)$, i.e. $\mathbf{s}_i \in S$ for $i = 1, \ldots, N$, is given by:

$$\begin{aligned}
(\mathbf{s}_1,\ldots,\mathbf{s}_N) \ &\models \pi_i^\varphi \iff (\mathbf{s}_i, 0) \models \varphi,\ i = 1,\ldots,N,\\
(\mathbf{s}_1,\ldots,\mathbf{s}_N) \ &\models \Psi_1 \wedge \Psi_2 \iff (\mathbf{s}_1,\ldots,\mathbf{s}_N) \models \Psi_1\\
&\qquad \wedge (\mathbf{s}_1,\ldots,\mathbf{s}_N) \models \Psi_2,\\
(\mathbf{s}_1,\ldots,\mathbf{s}_N) \ &\models \Psi_1 \vee \Psi_2 \iff (\mathbf{s}_1,\ldots,\mathbf{s}_N) \models \Psi_1\\
&\qquad \vee (\mathbf{s}_1,\ldots,\mathbf{s}_N) \models \Psi_2.
\end{aligned}$$

MA-STL extends STL predicates with logical operators (and, or, not) and temporal operators (eventually, always) to define properties for the team. Each robot i has motion kinodynamic constraints:

$$\dot{\mathbf{x}}_i(t) = f_i(\mathbf{x}_i(t), \mathbf{u}_i(t)), \tag{2}$$

where $\mathbf{x}_i(t)$ is the state and $\mathbf{u}_i(t)$ is the control for robot i.

Definition 3 (Kinodynamic Constraints): Let $\mathcal{S}_i = (f_i, X_i, U_i, \mathbf{x}_i^{\text{init}})$ be a dynamical system for robot i, where $X_i \subseteq \mathbb{R}^d$ and $U_i \subseteq \mathbb{R}^m$ are bounded state and control spaces. The state space X_i contains obstacles $X_{\text{obs}} \subset X_i$, similarly the free space is defined as $X_{i,\text{f}} = X_i \setminus X_{\text{obs}}$. We asume the function $f_i : X_i \times U_i \to X_i$ to be Lipschitz, and $\mathbf{x}_i^{\text{init}}$ is the initial state satisfying $\Pi_i(\mathbf{x}_i^{\text{init}}) = p_i^{\text{init}}$. Each robot evolves according to (2), with $\mathbf{x}_i(t) \in X_i$ and $\mathbf{u}_i(t) \in U_i$. $\mathbf{x}_i[\mathbf{x}_i^{\text{init}}, u](t)$ is the trajectory that originates at $\mathbf{x}_i^{\text{init}}$ under control policy $\mathbf{u}_i$. Let $V_i = \textsc{Fun}(\mathbb{R}_{\geq 0}, U_i)$ be the set of all control policies for robot i.

The system $\mathcal{S}_i$ satisfies an STL specification φ under a control policy $\mathbf{u}_i \in V_i$ if the state trajectory starting at $\mathbf{x}_i^{\text{init}}$ satisfies φ, i.e., $\mathbf{x}_i[\mathbf{x}_i^{\text{init}}, u] \models \varphi$.

These kinodynamic constraints ensure that planned trajectories are physically feasible, adhering to limits on velocity, acceleration, and other dynamic properties.

A finite time horizon T defines the interval $[0, T]$ for planning and evaluation. For each robot i, a positive constant

$s_i > 0$ represents its size, modeled as an axis-aligned box centered at $p_i(t) = \Pi_i(\mathbf{x}_i(t))$ at time t:

$$B_{s_i}(p_i(t)) = \{q \in W : \|q - p_i(t)\|_\infty \leq s_i\}, \tag{3}$$

where $\|\cdot\|_\infty$ is the maximum norm. This square representation can be generalized to other shapes.

Definition 4 (MA-STL Satisfaction with Robustness): A set of position trajectories $(p_1(t), p_2(t), \ldots, p_N(t))$ is ϵ-robust with respect to an MA-STL formula Ψ if, for any corresponding actual state trajectories $\hat{\mathbf{x}}_1(t), \hat{\mathbf{x}}_2(t), \ldots, \hat{\mathbf{x}}_N(t)$ satisfying

$$\sup_{t\in[0,T]} \left\|\Pi_i(\hat{\mathbf{x}}_i(t)) - p_i(t)\right\| \leq \epsilon \quad \text{for all } i, \tag{4}$$

and following their dynamics

$$\dot{\hat{\mathbf{x}}}_i(t) = f_i(\hat{\mathbf{x}}_i(t), \hat{\mathbf{u}}_i(t)) \quad \text{with } \hat{\mathbf{u}}_i(t) \in U_i, \tag{5}$$

the actual position trajectories $(\hat{p}_1(t), \hat{p}_2(t), \ldots, \hat{p}_N(t)) \models \Phi$ and ensure no collisions, i.e., $\|\hat{p}_i(t) - \hat{p}_j(t)\|_\infty \geq s_i + s_j + 2\epsilon$ for all $i \neq j$.

The parameter ϵ ensures collision avoidance by increasing effective robot sizes and provides a robustness margin for STL satisfaction under perturbations.

Problem 1: Given an MA-STL motion planning problem with kinodynamic constraints

$$(W, N, T, \{s_i\}_{i=1}^N, \epsilon, \Psi, \{\mathcal{S}_i\}_{i=1}^N)$$

compute a set of continuous state trajectories $\mathbf{x}_1(t), \mathbf{x}_2(t), \ldots, \mathbf{x}_N(t)$, where each $\mathbf{x}_i(t) : [0, T] \to X_i$, such that they satisfy the MA-STL Ψ, respect kinodynamic constraints, ensure robustness, and guarantee collision avoidance as defined in Definitions 1–4.

III. Decoupled Robot Planning

Here, we propose a constrained Bayesian optimization-based tree search (cBOT) algorithm for computing collision-free trajectories for individual robots. The cBOT planner is an extension of the BOW planner which enjoys probabilistic completeness within a planning horizon. The search tree, denoted by $\mathcal{T}$, is rooted at the initial state $\mathbf{x}_{\text{init}} \in X$ and incrementally expanded toward the goal state $\mathbf{x}_{\text{goal}}$ by sequentially selecting and executing locally optimal controls. At any iteration t, the planner focuses on a local control neighborhood

$$V_d(\mathbf{x}_t) = \{\mathbf{u} \in U : \|\mathbf{u} - \bar{\mathbf{u}}\|_2 \leq d\}, \tag{6}$$

where $d > 0$ is the window radius and $\bar{\mathbf{u}}$ is the nominal control from the previous iteration. This *planning window* restricts search to controls that are reachable within a short horizon, enabling both computational tractability and fine-grained maneuvering. Here, we dropped subindex i from $\mathbf{x}$ for notation simplicity since this procedure applies to each robot i; moreover, $\mathbf{h}_t$ will denote $\mathbf{h}(t)$'s discrete version for any function $\mathbf{h}(t)$.

Within this local window, a set of p candidate controls $\{\mathbf{u}_t^{(i)}\}_{i=1}^p \subset V_d(\mathbf{x}_t)$ is sampled and evaluated using constrained Bayesian optimization (CBO). The objective function $J(\mathbf{u})$, which may encode criteria such as travel time, energy, or smoothness, is modeled as a Gaussian process (GP)

$$J(\mathbf{u}) \sim \mathcal{GP}(\mu_J(\mathbf{u}), \sigma_J^2(\mathbf{u})). \tag{7}$$

Each motion constraint $c_k(\mathbf{u}) \leq 0$, typically arising from obstacle avoidance or kinematic limits, is modeled independently as

$$c_k(\mathbf{u}) \sim \mathcal{GP}(\mu_k(\mathbf{u}), \sigma_k^2(\mathbf{u})), \tag{8}$$

allowing the probability of constraint satisfaction to be computed as

$$P_{\text{feas}}(\mathbf{u}) = \prod_{k=1}^{K} \Phi\left(-\frac{\mu_k(\mathbf{u})}{\sigma_k(\mathbf{u})}\right), \tag{9}$$

where $\Phi(\cdot)$ is the Gaussian cumulative distribution function.

The search is guided by the constrained expected improvement (CEI) acquisition function

$$\text{CEI}(\mathbf{u}) = \text{EI}(\mathbf{u}) \cdot P_{\text{feas}}(\mathbf{u}), \tag{10}$$

where $\text{EI}(\mathbf{u})$ is the standard Bayesian optimization improvement metric over the best observed cost J_{best}:

$$\text{EI}(\mathbf{u}) = \begin{cases} z(\mathbf{u})\,\Phi(z(\mathbf{u})) + \sigma_J(\mathbf{u})\,\phi(z(\mathbf{u})), & \sigma_J(\mathbf{u}) > 0, \\ 0, & \text{otherwise}, \end{cases} \tag{11}$$

with

$$z(\mathbf{u}) = \frac{J_{\text{best}} - \mu_J(\mathbf{u})}{\sigma_J(\mathbf{u})}, \tag{12}$$

and $\phi(\cdot)$ the Gaussian probability density function. The CEI formulation naturally balances performance improvement and safety: candidates with high EI but low feasibility are penalized, whereas safer but slightly less promising candidates may be favored if their overall CEI is higher.

Once the control

$$\mathbf{u}_t^* = \underset{\mathbf{u}\in V_d(\mathbf{x}_t)}{\arg\max}\ \text{CEI}(\mathbf{u}) \tag{13}$$

is selected, the state is propagated forward over a short horizon Δt using the motion model $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$, integrated via the fourth-order Runge–Kutta method:

$$\mathbf{x}_{t:t+\Delta t} = \text{RK4}\left(f, \mathbf{x}_t, \mathbf{u}_t^*, \Delta t\right). \tag{14}$$

The resulting short-horizon trajectory is checked for constraint satisfaction using the collision checker. If all constraints are met, the new states added to $\mathcal{T}$ as a child of $\mathbf{x}_t$ with an edge labeled by $\mathbf{u}_t^*$; otherwise, the planner selects a random valid parent from $\mathcal{T}$ and resumes from that state.

Algorithm 1 Constrained BO-based Tree Search

Input: Initial state $\mathbf{x}_0$, goal state $\mathbf{x}_{\text{goal}}$, parameters, goal radius r_{goal}, STL constraints φ
Output: Motion tree, $\mathcal{T}$
Initialize tree with current state, $\mathcal{T} = \{x_0\}$
while $\|\mathbf{x}_t - \mathbf{x}_{\text{goal}}\|_\infty \leq r_{\text{goal}}$ **do**
    Define planning window in control space, V_d
    Initiate a dataset, $\mathcal{D} \leftarrow \emptyset$
    for Sample control $\mathbf{u} \in V_d$ **do**
        Evaluate cost $J(\mathbf{u})$ and constraints $c_k(\mathbf{u})$ for a short planning horizon
        Update dataset, $\mathcal{D} \leftarrow D \cup \{(\mathbf{u}, J(\mathbf{u}), c_k(\mathbf{u}))\}$
    end for
    Train $\mathcal{GP}$ models with the dataset $\mathcal{D}$ online
    Compute CEI values using equation (10)
    Select control $\mathbf{u}_t^*$ with the highest CEI value using equation (13)
    Predict states $\mathbf{x}_{t:t+\Delta t}$ by applying the same control $\mathbf{u}_t^*$ using equation (14)
    if constraints satisfy, $\mathbf{x}_{t:t+\Delta t} \models \varphi$ **then**
        Extend the motion tree $\mathcal{T}$ with $\mathbf{x}_{t:\Delta t}$ and $\mathbf{u}_t^*$
    else
        Update current state $\mathbf{x}_t$ by randomly sampling a valid parent state $\mathbf{x}$ from the motion tree $\mathcal{T}$
    end if
end while

This iterative process, as summarized in Algorithm 1, continues until the current state satisfies the goal condition $\|\mathbf{x}_t - \mathbf{x}_{\text{goal}}\|_\infty \leq r_{\text{goal}}$, where r_{goal} represents the goal tolerance radius. To guarantee collision-free trajectories, the algorithm validates each intermediate trajectory segment $\mathbf{x}_{t:t+\Delta t}$ against the set of obstacles $\mathcal{O}$ through STL specification verification, formulated as $\varphi := \bigwedge_{o \in \mathcal{O}} \square_{[0,T]}(dist(p_i(t), o) > d_{\min})$, ensuring minimum separation distances are maintained throughout the planned path. Here, $dist(p, S) = \inf_{s \in S} \|p - s\|_\infty$. Notably, this decoupled planning approach intentionally excludes inter-robot collision checking at the individual trajectory level, delegating this responsibility to the high-level STL-KCBS planner which handles multi-robot coordination and conflict resolution.

IV. Signal Temporal Logic enhanced KCBS

The Signal Temporal Logic enhanced Kinodynamic Conflict-Based Search (STL-KCBS) algorithm replaces conventional geometric intersection tests with temporal logic specifications, providing a more expressive and verifiable framework for multi-robot coordination than standard K-CBS approaches. In terms of conflict detection, conventional K-CBS identifies conflicts through direct geometric intersection, defined as $K = (i, j, [t_s, t_e])$ where $B_{s_i}(p_i(t)) \cap B_{s_j}(p_j(t)) \neq \emptyset$. In contrast, STL-KCBS employs robustness-based conflict detection using STL monitors, expressed as $\mathcal{K} = \{(i, j, t) \in \{1, \ldots, N\}^2 \times [t_s, t_e] : \mu(t) < 0\}$, where $\mu(t)$ represents the STL robustness metric, and φ is the safety specification. For constraint representation, conventional K-CBS generates spatial-temporal constraints such as $C_i = \{(\mathbf{x}, t) \in X_i \times [t_s, t_e] : \mathbf{x} \in B_{s_i}(p_i(t))\}$. However, STL-KCBS expresses constraints as temporal logic formulas that can encode complex behavioral requirements beyond simple collision avoidance.

The STL-KCBS algorithm enhances the conventional K-CBS framework by integrating STL monitors into the conflict detection mechanism, offering several theoretical and practical advantages for multi-robot motion planning. At its core, the algorithm initializes an STL safety monitor φ for each robot pair (i, j) where $i \neq j$. This monitor encodes safety specifications that efficiently enforce collision avoidance, including spatio-temporal safety constraints such as $\varphi = \square_{[0,T]}(\|p_i(t) - p_j(t)\|_\infty > d_{\text{min}})$.

Robustness-based conflict detection in STL-KCBS utilizes the STL robustness metric $\mu(t)$, which provides a quantitative measure of specification satisfaction: greater than zero if φ is satisfied with margin, equal to zero if marginally satisfied, and less than zero if violated. This continuous metric enables gradient-based optimization for trajectory refinement, early conflict prediction through robustness degradation monitoring, and quantitative safety margins for robust planning. At each timestep t, the algorithm constructs a comprehensive state vector $\bar{\mathbf{x}} = [t, p_i(t), p_j(t)]^T$, incorporating time, and positions to support STL specifications.

Algorithm 2 Kinodynamic Conflict-Based Search with STL Monitor

procedure STLConflictSearch($plan$)
    $T \leftarrow \max$(trajectory lengths in plan)
    conflicts $\leftarrow \emptyset$
    for each robot pair (i, j) **do**
        Initialize STL safety monitor μ for p_i and p_j
        for $t = 0$ to $T - 1$ **do**
            $\bar{\mathbf{x}} \leftarrow$ create state vector at timestep t
            μ.add_sample($\bar{\mathbf{x}}$)
        end for
        for $t = 0$ to $T - 1$ **do**
            if $\mu(t) < 0$ **then**
                conflicts $\leftarrow$ conflicts $\cup \{(i, j, t)\}$
            ▷ Add conflict between i and j at timestep t
            end if
        end for
    end for
    return conflicts
end procedure

Algorithm 2 shows the workflow of STL-KCBS conflict detection which begins with an initialization phase, where the maximum trajectory length T across all robots is determined, and STL monitors are initialized for each robot pair with appropriate safety specifications. In the monitor population step, for each timestep $t \in [0, T-1]$, the state vector $\bar{\mathbf{x}}$ is constructed from a pairwise robot trajectories, and samples are added to the STL monitor via μ.add_sample($\bar{\mathbf{x}}$). During robustness evaluation, for each timestep t, the robustness $\mu(t)$ is computed; if $\mu(t) < 0$, a conflict (i, j, t) is recorded. Finally, conflict resolution involves generating

STL-constrained trajectory refinements and propagating temporal constraints through the search tree.

Implementation considerations for STL-KCBS include hierarchical decomposition with careful monitor design, where the choice of STL specifications impacts performance; longer horizons increase expressiveness but raise computational costs, atomic predicates must balance granularity with evaluation efficiency, and robustness computation can leverage efficient algorithms. Integration with low-level planning requires the planner to be STL-aware, minimizing the cost-to-go function, i.e., $\min_{u_i \in U_i} \|\mathbf{x}_i^{\text{init}} + \int_0^T f_i(\mathbf{x}_i(t), \mathbf{u}_i(t))dt - \mathbf{x}_i^{\text{goal}}\|_2$ subject to robust satisfaction, $\mu(t) \geq 0$.

V. Experiments and Results

All experiments were performed on a desktop computer running Ubuntu 22.04 LTS with an Intel® Core™ i7-10700 CPU operating at 2.90 GHz and 32 GB of RAM. For robot localization, we utilized a VICON motion capture system running at 120 Hz for indoor environments and standard GPS sensors for outdoor scenarios. The decoupled planner operated under the following kinodynamic constraints: maximum velocity of 1.0 $\mathrm{m\,s^{-1}}$, minimum velocity of 0.0 $\mathrm{m\,s^{-1}}$, maximum acceleration of 0.5 $\mathrm{m/s^2}$, maximum yaw rate of 0.6981 $\mathrm{rad\,s^{-1}}$, and maximum yaw acceleration of 2.0472 $\mathrm{rad/s^2}$. We configured the prediction horizon between 3.0 s and 50.0 s with a time discretization step of 0.1 s. The KCBS algorithm employed merge bound parameters ranging from 25 to 60. For GP kernel within the decoupled planner, we utilized a Squared Exponential ARD kernel (Radial Basis Function) to effectively learn the local cost map.

A. STL Specification and Hierarchical Decomposition

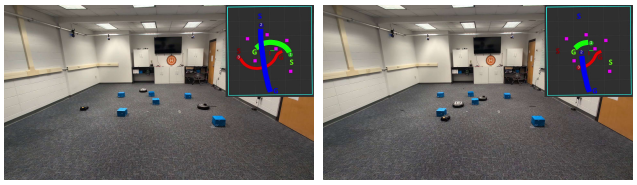
STLcBOT combines the decoupled cBOT planner with the high-level STL-KCBS planner to handle multi-robot coordination under STL constraints. Safety constraints ensure minimum separation distances throughout the mission:

$$\varphi_{\text{safety}} = \bigwedge_{i \neq j} \left(\square_{[0,T]} \left(||p_i(t) - p_j(t)||_\infty > d_{\min} \right) \right) \wedge \bigwedge_{o \in \mathcal{O}} \left(dist(p_i(t), o) > d_{\min} \right),$$

while reachability constraints guarantee goal achievement: $\varphi_{\text{reachability}} = \Diamond_{[0,T]}(\|p_i(t) - p_i^{\text{goal}}\|_\infty < r_{\text{goal}})$. The complete mission specification combines both constraints: $\varphi := \varphi_{\text{safety}} \wedge \varphi_{\text{reachability}}$. The hierarchical decomposition assigns cBOT to handle reachability and static obstacle avoidance, while STL-KCBS manages inter-robot collision avoidance through pairwise conflict resolution.

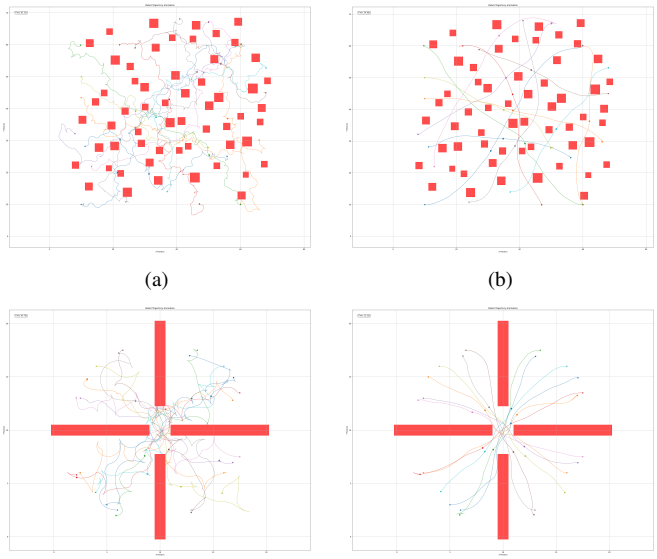
B. Indoor Experiments and Qualitative analysis

We conducted controlled indoor experiments using a heterogeneous multi-robot system comprising two iRobot Create 3 educational robots (radius 0.17 m each) and one Waveshare Rover robot (0.19 m width × 0.17 m width) operating within a 6.5 m × 5.5 m bounded workspace containing six static obstacles (0.2 m × 0.2 m each) at known positions, denoted by $\mathcal{O} = \{o_1, \ldots, o_6\}$. As illustrated in



(a) (b)

Fig. 1: STLcBOT planner applied to three UGVs navigating a cluttered indoor environment. Subfigures 1a and 1b demonstrate kinodynamically-constrained trajectories that enable accurate trajectory following and collision avoidance.

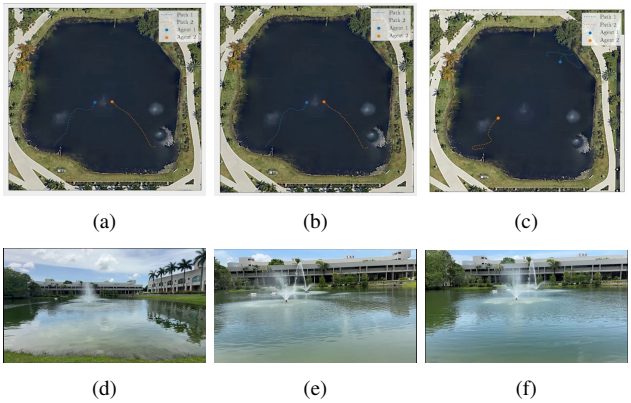


(a) (b) (c) (d)

Fig. 2: Comparison of RRT and cBOT planners for twelve robots under STL constraints in forest and cross-hall environments. Figures 2a and 2c show RRT-generated trajectories, while Figures 2b and 2d display cBOT-generated paths. cBOT produces notably shorter and smoother trajectories compared to the longer, less structured paths from RRT-based planning.

Fig. 1, three color-coded robots were tasked with navigating from initial positions $(0.50, 1.35)$ m, $(-1.87, -0.57)$ m, and $(0.16, -2.00)$ m to goal locations $(-0.18, -0.9)$ m, $(2.8, 0.1)$ m, and $(0.05, 0.82)$ m respectively, while avoiding the magenta-colored static obstacles. The STLcBOT solver successfully computed collision-free trajectories within 42 $\mathrm{s^{-3}}$, generating paths of lengths 2.933 m, 4.73 m, and 3.84 m for the three robots with corresponding average velocities of 0.16 $\mathrm{m\,s^{-1}}$, 0.18 $\mathrm{m\,s^{-1}}$, and 0.16 $\mathrm{m\,s^{-1}}$, totaling 11.503 m in combined path length. Despite spatial intersections in the planned paths, temporal coordination through differentiated control inputs ensured collision-free execution throughout the mission. The approach leverages the cBOT planner for individual trajectory generation while the STL-enhanced K-CBS planner employs merge and restart heuristics for efficient conflict validation, achieving high success rates in cluttered environments as demonstrated in the accompanying project website video.

To further evaluate the effectiveness of our approach, Fig. 2 presents a qualitative comparison between STLcBOT and RRT-based planners in two challenging environments: a Poisson forest and a cross-hall configuration. The compari-



(a) (b) (c)

(d) (e) (f)

Fig. 3: Field validation of STLcBOT using two ASVs in a lake environment with complex navigation scenarios. Figs.a–c show planned trajectories, while Figs.d–f display ASVs executing missions in the operational area.

son reveals two key qualitative advantages of our method. First, trajectory smoothness is significantly improved, as cBOT generates visually smoother paths compared to the characteristically jagged trajectories produced by RRT algorithms. Second, trajectory efficiency is enhanced through shorter path lengths, demonstrating superior navigation performance. The contrast between Figs 2a and 2c versus Figs 2b and 2d clearly illustrates that RRT produces longer and less structured paths, while cBOT yields more organized and coordinated motion patterns. This visual difference is particularly pronounced in the cross-hall environment, where the structured nature of cBOT trajectories becomes evident.

C. Outdoor Experiments in a Lake environment

We conducted field experiments in a lake environment using fleets of two and three Autonomous Surface Vehicles (ASVs), each equipped with YSI EXO2 sonde sensors for measuring water quality parameters. The experimental workspace encompassed a $25\,\mathrm{m} \times 35\,\mathrm{m}$ lake area containing two circular fountain obstacles with $2\,\mathrm{m}$ radii. Localization was achieved through sensor fusion of GPS and IMU data, providing accurate position estimates for trajectory tracking. For the two-ASV configuration, we designed three challenging scenarios that tested the planner's capability to handle complex spatial coordination: X-pattern navigation where both vehicles traverse intersecting paths with high point density in the central lake region, and position-swapping maneuvers where ASV 1's initial position served as ASV 2's goal location and vice versa. The three-ASV experiment introduced additional complexity by requiring ASV 3 to navigate to a goal location positioned at the intersection of the planned trajectories for ASV 1 and ASV 2, creating a multi-vehicle coordination challenge that demands precise temporal and spatial planning to avoid collisions while maintaining mission objectives.

In all cases, STLcBOT computed collision-free trajectories in under one second. The three robots achieved average path lengths of $80.10\ \mathrm{m}$, $73.82\ \mathrm{m}$, and $72.62\ \mathrm{m}$, with corresponding average velocities of $0.1676\ \mathrm{m\,s^{-1}}$, $0.1651\ \mathrm{m\,s^{-1}}$, and $0.1669\ \mathrm{m\,s^{-1}}$, totaling $226.54\ \mathrm{m}$. Despite lower localization accuracy compared to indoor tests and intersecting

(a) (b)

Fig. 4: STLcBOT planner applied to three ASVs navigating in a lake environment with challenging spatial configurations. Subfigures (4a) and (4b) show trajectory execution where ASVs reach their respective goals while avoiding fountain obstacles.

trajectories, differentiated control inputs enabled collision-free execution. These results demonstrate STLcBOT's ability to handle heterogeneous vehicles and their kinodynamic constraints in real-world aquatic environments.

D. Benchmark Results

We evaluate the proposed STLcBOT framework against nine baseline planners. All methods were executed for 10 independent trials under identical conditions. The baselines span a diverse set of planning paradigms and implementations. STLcBOT (proposed) and STLRRT extend KCBS with STL-guided specifications, while KcBOT and KRRT are KCBS-based methods implemented in C++ without STL monitoring. The priority-based approaches PPcBOT and PPRRT are also implemented in C++. In contrast, PBS+STGCS, SP+STGCS, and PP+STGCS are convex optimization–based planners implemented in Python and solved with the MOSEK optimizer, while SP+STRRT* represents a sampling-based STL variant developed in Python.

Experiments were conducted across four representative environments (Fig. 5) with robot team sizes of 2, 4, 6, 8, 10, 12, and 50. Env.1 is an open workspace designed to evaluate baseline navigation efficiency. Env.2 introduces bottleneck corridors that require cooperative conflict resolution. Env.3 is a cluttered forest environment with dense obstacles and narrow passages, stressing scalability under constrained maneuvering. Finally, Env.4 is a canonical bugtrap environment combining wide-open and congested regions, requiring both global coordination and local adaptability.

Success Rates. The proposed STLcBOT consistently achieved 100% success across all environments and team sizes, demonstrating strong robustness to both structural complexity and increasing density. Among the baselines, KcBOT and PPcBOT also maintained high completeness. By contrast, RRT-based methods exhibited significant degradation as the number of robots increased. In Env.3, both KRRT and STLRRT failed beyond 12 robots, while in Env.2, KRRT degraded to about 80% success and STLRRT dropped as low as 40% at 50 robots. PPRRT proved comparatively more resilient, though its performance also declined in dense and cluttered scenarios. Convex optimization–based planners struggled severely in cluttered environments. In particular, all STGCS-based methods failed to find any feasible solutions in Env. 3, where narrow passages and high density created spatiotemporal bottlenecks. SP+STGCS and RP+STGCS broke down beyond 4 robots even in simpler environments.

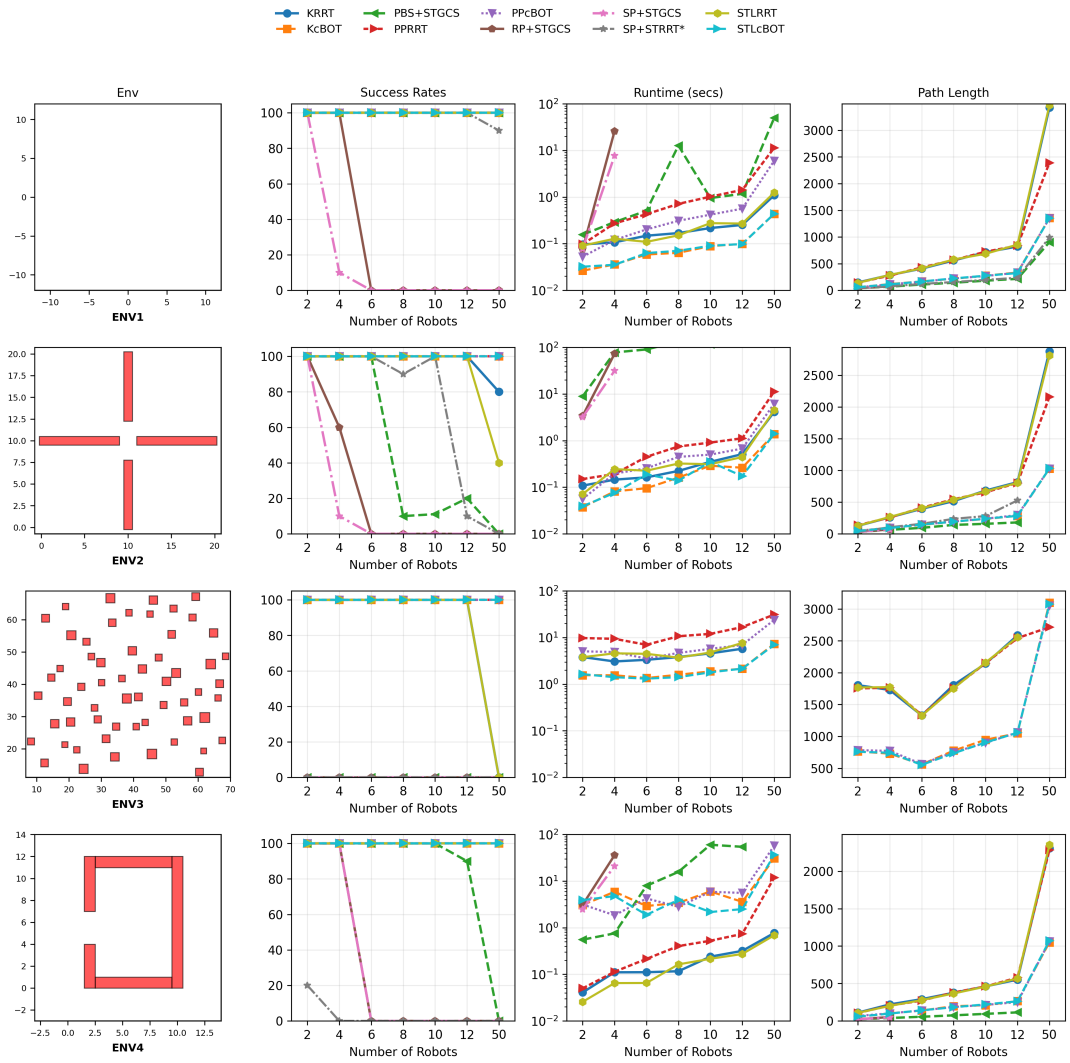


Fig. 5: Benchmark comparison of multi-robot motion planning algorithms across four representative environments (Env. 1: empty, Env. 2: cross-hall, Env. 3: forest, Env. 4: bugtrap). The columns report success rates, runtimes, and path lengths as the number of robots increases. The proposed STLcBOT, together with other cBOT-based methods (KcBOT, PPcBOT) [27, 16], consistently achieves near-perfect success and efficient paths with tractable runtimes. RRT-based methods (PPRRT, STLRRT, KRRT) [28, 27, 16, 1] degrade under clutter and density, while convex optimization–based approaches (PBS+STGCS, SP+STGCS, RP+STGCS) [26] and temporal sampling extension (SP+STRRT*) [26] generate competitive paths when successful but fail to scale, particularly in Env. 3.

SP+STRRT* achieved limited success only in the empty map but collapsed in Env.4 after 2 robots. PBS+STGCS performed adequately in Env.1 and showed limited success in Env.4, but failed in more challenging maps, particularly Env.2.

Runtime. STLcBOT and KcBOT achieved runtimes below 1 second in Envs. 1 and 2 across all team sizes, with the sole exception of 50 robots in Env. 2, confirming their efficiency. Despite the additional STL monitoring overhead, STLcBOT remained tractable by pruning conflicts through robustness metrics. In contrast, RRT-based planners exhibited exponential growth with team size, exceeding 10^1 seconds at 50 robots, while STLRRT and KRRT consistently occupied a middle ground between STLcBOT and the slower PPRRT. In Envs.2 and 3, cBOT-based planners such as STLcBOT and KcBOT clearly outperformed the RRT-based methods. STGCS-based planners were not competitive in runtime, as their reliance on large-scale convex optimization caused rapid escalation even in relatively simple environments. In Env.3, runtime analysis was moot for STGCS-based methods since none succeeded, whereas STLcBOT and KcBOT remained tractable. In Env.4, RRT-based methods achieved faster runtimes than cBOT-based ones, albeit with substantially longer trajectory length.

Trajectory Efficiency. The proposed STLcBOT produced short, specification-compliant trajectories that scaled sublinearly with team size and remained below 1200 m in Env.4, even with 50 robots. KcBOT and PPcBOT achieved similarly compact paths, whereas RRT-based methods exhibited severe path inflation, with lengths exceeding 3000m in Env.1 and 2500m in Env.2. In Env.3, the cBOT-based planners generated slightly longer trajectories than RRT-based methods at 50 robots, but the margin was negligible compared to their significantly higher success rates. Convex optimiza-

tion–based methods occasionally produced competitive path lengths; however, their frequent failures in cluttered environments, particularly Env.3, prevented consistent evaluation.

These results highlight that the proposed STLcBOT framework delivers the most reliable balance between completeness, runtime efficiency, and trajectory quality. Unlike RRT-based planners, which degrade under density and structural complexity, and unlike STGCS-based methods, which fail in cluttered maps such as Env.3, STLcBOT ensures consistent, scalable, and specification-compliant multi-robot planning across all environments and large team sizes.

VI. Conclusion

We presented a two-stage framework for multi-robot trajectory planning under Signal Temporal Logic (STL) specifications while respecting kinodynamic constraints. At the single-robot level, the proposed constrained Bayesian optimization–based tree search (cBOT) efficiently learns local cost maps and feasibility constraints to generate shorter, collision-free trajectories. At the multi-robot level, the STL-KCBS algorithm integrates STL monitoring into conflict-based search to ensure specification satisfaction with scalability and probabilistic completeness. We conducted extensive testing across diverse environments and team sizes, along with real-world experiments using ground and surface vehicles. Our planner generates shorter and smoother trajectories compared to existing RRT-based approaches. Moreover, our framework scales effectively to larger problems, successfully solving planning challenges for teams of up to 50 robots in complex environments where existing exact STL planners fail beyond 6 robots. Future research directions include developing theoretical foundations for the proposed framework and expanding its application scope to encompass task and motion planning domains.

References

[1] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *International symposium on formal techniques in real-time and fault-tolerant systems*, pp. 152–166, Springer, 2004.

[2] T. Wongpiromsarn, U. Topcu, N. Ozay, H. Xu, and R. M. Murray, "Tulip: a software toolbox for receding horizon temporal logic planning," in *Proceedings of the 14th international conference on Hybrid systems: computation and control*, pp. 313–314, 2011.

[3] S. Sadraddini and C. Belta, "Robust temporal logic model predictive control," in *2015 53rd Annual Allerton Conference on Communication, Control, and Computing (Allerton)*, pp. 772–779, IEEE, 2015.

[4] K. Leung, N. Aréchiga, and M. Pavone, "Backpropagation through signal temporal logic specifications: Infusing logical structure into gradient-based methods," *The International Journal of Robotics Research*, vol. 42, no. 6, pp. 356–370, 2023.

[5] A. Linard, I. Torre, E. Bartoli, A. Sleat, I. Leite, and J. Tumova, "Real-time rrt* with signal temporal logic preferences," in *International Conference on Intelligent Robots and Systems*, pp. 8621–8627, IEEE, 2023.

[6] J. Chatrola, A. Ajith, K. Leahy, and C. Chamzas, "Multi-layer motion planning with kinodynamic and spatio-temporal constraints," in *Proceedings of the 28th ACM International Conference on Hybrid Systems: Computation and Control*, pp. 1–7, 2025.

[7] P. Kapoor, A. Balakrishnan, and J. V. Deshmukh, "Model-based reinforcement learning from signal temporal logic specifications," *arXiv preprint arXiv:2011.04950*, 2020.

[8] Y. Meng and C. Fan, "Signal temporal logic neural predictive control," *IEEE Robotics and Automation Letters*, vol. 8, no. 11, pp. 7719–7726, 2023.

[9] D. Sun, J. Chen, S. Mitra, and C. Fan, "Multi-agent motion planning from signal temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 7, no. 2, pp. 3451–3458, 2022.

[10] J. Eappen, Z. Xiong, D. Patel, A. Bera, and S. Jagannathan, "Scaling safe multi-agent control for signal temporal logic specifications," in *Conference on Robot Learning*, vol. 270, pp. 3516–3535, PMLR, 2024.

[11] Z. Liu, J. Dai, B. Wu, and H. Lin, "Communication-aware motion planning for multi-agent systems from signal temporal logic specifications," in *2017 American Control Conference (ACC)*, pp. 2516–2521, IEEE, 2017.

[12] L. Lindemann and D. V. Dimarogonas, "Barrier function based collaborative control of multiple robots under signal temporal logic tasks," *IEEE Transactions on Control of Network Systems*, vol. 7, no. 4, pp. 1916–1928, 2020.

[13] D. Gundana and H. Kress-Gazit, "Event-based signal temporal logic synthesis for single and multi-robot tasks," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3687–3694, 2021.

[14] X. Zhou, Y. Zou, S. Li, X. Li, and H. Fang, "Distributed model predictive control for multi-robot systems with conflicting signal temporal logic tasks," *IET Control Theory & Applications*, vol. 16, no. 5, pp. 554–572, 2022.

[15] G. Sharon, R. Stern, A. Felner, and N. R. Sturtevant, "Conflict-based search for optimal multi-agent pathfinding," *Artificial intelligence*, vol. 219, pp. 40–66, 2015.

[16] H. Ma, D. Harabor, P. J. Stuckey, J. Li, and S. Koenig, "Searching with consistent prioritization for multi-agent path finding," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 33, pp. 7643–7650, 2019.

[17] A. Andreychuk, K. Yakovlev, P. Surynek, D. Atzmon, and R. Stern, "Multi-agent pathfinding with continuous time," *Artificial Intelligence*, vol. 305, p. 103662, 2022.

[18] A. Sintov and A. Shapiro, "Time-based rrt algorithm for rendezvous planning of two dynamic systems," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 6745–6750, IEEE, 2014.

[19] M. Hüppi, L. Bartolomei, R. Mascaro, and M. Chli, "T-prm: Temporal probabilistic roadmap for path planning in dynamic environments," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 10320–10327, IEEE, 2022.

[20] F. Grothe, V. N. Hartmann, A. Orthey, and M. Toussaint, "St-rrt*: Asymptotically-optimal bidirectional motion planning through space-time," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 3314–3320, IEEE, 2022.

[21] T. Marcucci, P. Nobel, R. Tedrake, and S. Boyd, "Fast path planning through large collections of safe boxes," *IEEE Transactions on Robotics*, 2024.

[22] T. Cohn, M. Petersen, M. Simchowitz, and R. Tedrake, "Non-euclidean motion planning with graphs of geodesically convex sets," *The International Journal of Robotics Research*, p. 02783649241302419, 2023.

[23] V. Kurtz and H. Lin, "Temporal logic motion planning with convex optimization via graphs of convex sets," *IEEE Transactions on Robotics*, vol. 39, no. 5, pp. 3791–3804, 2023.

[24] S. Y. C. Chia, R. H. Jiang, B. P. Graesdal, L. P. Kaelbling, and R. Tedrake, "Gcs*: Forward heuristic search on implicit graphs of convex sets," *arXiv preprint arXiv:2407.08848*, 2024.

[25] T. Marcucci, M. Petersen, D. von Wrangel, and R. Tedrake, "Motion planning around obstacles with convex optimization," *Science robotics*, vol. 8, no. 84, p. eadf7843, 2023.

[26] J. Tang, Z. Mao, L. Yang, and H. Ma, "Space-time graphs of convex sets for multi-robot motion planning," *International Conference on Intelligent Robots and Systems*, 2025.

[27] J. Kottinger, S. Almagor, and M. Lahijanian, "Conflict-based search for multi-robot motion planning with kinodynamic constraints," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 13494–13499, IEEE, 2022.

[28] S. LaValle, "Rapidly-exploring random trees: A new tool for path planning," *Research Report 9811*, 1998.