Robust Multi-Layered Sampling-Based Path Planning for Temporal Logic-Based Missions

Yoonseon Oh, Kyunghoon Cho, Yunho Choi, and Songhwai Oh

***Abstract*— We investigate a path planning algorithm for generating robust and safe paths, which satisfy mission requirements specified in linear temporal logic (LTL). When robots are deployed to perform a mission, there can be disturbances which can cause mission failures or collisions with obstacles. Hence, a path planning algorithm needs to consider safety and robustness against possible disturbances. We present a robust path planning algorithm, which maximizes the probability of success in accomplishing a given mission by considering disturbances in robot dynamics while minimizing the moving distance of a robot. The proposed method can guarantee the safety of the planned trajectory by incorporating an LTL formula and chance constraints in a hierarchical manner. A high-level planner generates a discrete plan satisfying the mission requirements specified in LTL. A low-level planner builds a sampling-based RRT search tree to minimize both the mission failure probability and the moving distance while guaranteeing the probability of collision with obstacles to be below a specified threshold. We validate the robustness and safety of paths generated by the algorithm in simulation and experiments using a quadrotor.**

I. Introduction

Path planning has been widely investigated with a goal to find a feasible path from a start state to a target region without collision [1]–[3]. In recent years, path planning has been extended to carry out more complicated missions, such as visiting sequential goals and covering multiple goals [4]–[6]. For example, we can assign a mission, such as "eventually visit region A, B or C after visiting region D". These types of missions can be represented using linear temporal logic (LTL) [7].

Recently, path planning algorithms find not only a feasible path to accomplish a mission but they look for a high quality path to optimize a given cost function, such as the energy consumption or travelled distance [8]. These algorithms formulate a path planning problem as an optimization problem, where constraints provided in an LTL formula are encoded using mixed-integer linear constraints [4], [9]. However, it becomes more challenging to solve the problem as the number of LTL constraints increases or the dynamics of a robot becomes more complex. In order to incorporate both LTL constraints and the robot dynamics more efficiently, hierarchical planning methods have been proposed [10], [11]. They suggest a high-level planner to satisfy LTL constraints and a low-level planner to find a feasible path considering the robot dynamics.

Existing LTL-based path planning approaches often assume that robot systems are purely deterministic. Unfortunately, in the real world, there can be disturbances which make the planned trajectory infeasible due to collision with obstacles. Chance constraints can provide a probabilistic safety guarantee for stochastic robot systems [3], [12]. However, they only consider the feasibility of the planned trajectory. To ensure the success of the planned trajectory, probabilistic computation tree logic can be used to synthesize a trajectory to satisfy a given probabilistic threshold of success [13], [14]. However, they are formulated using a Markov decision process (MDP) in a discretized space and cannot be applied to continuous or large state spaces.

In this paper, we address the path planning problem to perform LTL missions under disturbances in a continuous state space. The goal of this paper is to generate the optimal path which minimizes the probability of mission failures while guaranteeing safety. We propose a hierarchical path planning algorithm, where a high-level planner generates a discrete plan satisfying a mission specified in LTL and a low-level planner builds a rapidly-exploring random tree (RRT) search tree [1] to guarantee both robustness and safety. The proposed method first ensures that the probability of collision is less than a specified threshold by employing chance constraints [15]. We have derived an upper bound on the probability of mission failure. By minimizing this upper bound, the algorithm can find a trajectory with the largest probabilistic guarantee of mission success. The probability of mission success is maximized by redesigning the rewiring step of RRT* [2]. An extensive set of simulations shows that the proposed method completes LTL missions under disturbances with higher success probabilities. Furthermore, we have successfully demonstrated the proposed algorithm in field experiments using a quadrotor robot.

This paper is structured as follows. Section II describes the concept of linear temporal logic and path planning. In Section III, we formalize a path-planning problem for safety and robustness under perturbations. The proposed method is described in Section IV. Simulation and experimental results are given in Section V.

II. Preliminaries

Linear temporal logic (LTL) formulas consist of a set of atomic propositions, boolean operators and temporal operators [7]. An atomic proposition (AP) is a statement which is either true or false. We denote a set of atomic propositions

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (NRF-2017R1A2B2006136).

Y. Oh, K. Cho, Y. Choi, and S. Oh are with the Department of Electrical and Computer Engineering, ASRI, Seoul National University, Seoul 08826, Korea (e-mail: {yoonseon.oh, kyunghoon.cho, yunho.choi, songhwai.oh}@cpslab.snu.ac.kr).

by $\Pi = \{\pi_i, \ldots, \pi_N\}$, where π_i is an atomic proposition. An LTL formula is represented by propositions and temporal operators.

In this work, the goal of path planning is represented by an LTL formula. A mission is assumed to be a syntactically co-safe LTL formula (sc-LTL), which indicates that any infinite trace satisfying φ has a finite prefix which still satisfies φ [16]. Since the path planning problem is considered over a finite time, the form of LTL formulas is limited to co-safe LTL formulas [5].

For the LTL formula φ, a non deterministic finite automaton (NFA) can be constructed [17]. A number of open source software tools can be used to translate an LTL formula into an NFA [17]. For convenient computation, an NFA is converted to a deterministic finite automaton (DFA). We define a DFA $\mathcal{A}_\varphi$ over 2^Π with respect to φ as follows:

Definition 1: A deterministic finite automaton DFA is a tuple $\mathcal{A}_\varphi = (Q, \Sigma, \delta, q_{init}, Q_{acc})$, consisting of (1) a finite set of states Q, (2) a finite alphabet $\Sigma = 2^\Pi$, (3) a transition relation $\delta : Q \times \Sigma \to Q$, (4) a set of initial states $q_{init} \subseteq Q$, and (5) a set of accepting states $Q_{acc} \subseteq Q$.

III. Problem Formulation

The proposed problem considers a robot moving on a 2D plane, $\mathcal{X} \subset \mathbb{R}^2$. The position of the robot at time t is denoted by $\mathbf{x}_t = [x_t \; y_t]^T$ and the trajectory from time t_0 to t_1 is denoted by $\mathbf{x}^{t_0:t_1}$. The heading of the robot at time t is denoted by θ_t. The moving distance of the trajectory $\mathbf{x}^{0:T}$ is denoted by $d(\mathbf{x}^{0:T})$. We assume that the robot has a discrete-time unicycle model: $\dot{x} = v\cos(\theta),\ \dot{y} = v\sin(\theta),\ \dot{\theta} = \omega$, where v and ω are directional and angular velocities, respectively. The control input to the robot at time t is denoted by $\mathbf{u}_t = [v \; \omega]^T$. Then, a discrete-time dynamic model of the robot can be expressed as: $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\theta_t, \mathbf{u}_t),\ \theta_{t+1} = \theta_t + \omega\Delta t$, where Δt is a unit time interval.

The robot moves in cluttered environments with m obstacles. Let $\mathcal{X}_{obs}^j \subset \mathcal{X}$ be the region occupied by the jth obstacle. A set of all regions occupied by obstacles is denoted by $\mathcal{X}_{obs} = \cup_{j=1}^m \mathcal{X}_{obs}^j$. The robot performs a mission φ by visiting some regions of interest $\{R_i\}$ in a certain order while avoiding obstacles. If the trajectory $\mathbf{x}^{t_0:t_1}$ satisfies the mission requirements successfully, we denote it as $\mathbf{x}^{t_0:t_1} \Rightarrow \varphi$. In this paper, we assume that the shape of regions of interest and obstacles is convex and can be represented using polygons. The region of interest R_i is assumed to be an n_i-sided polygon consisting of n_i inequalities, i.e.,

$$\mathbf{x} \in R_i \Leftrightarrow \mathbf{x} \in \left\{\mathbf{x} | \mathbf{a}_{ij}^T \mathbf{x} \leq b_{ij}, \text{ for } j = 1, \cdots, n_i\right\}. \quad (1)$$

The outward normal vector of the jth side of R_i is denoted by $\mathbf{a}_{ij} \in \mathbb{R}^2$ and the distance between the jth side and the origin $(0,0)$ is denoted by a scalar b_{ij}. Similarly, the obstacle $\mathcal{X}_{obs}^j$ is also an m_j-sided polygon consisting of normal vectors $\mathbf{c}_{jk} \in \mathbb{R}^2$ and distances d_{jk}, such that

$$\mathbf{x} \in \mathcal{X}_{obs}^j \Leftrightarrow \mathbf{x} \in \left\{\mathbf{x} | \mathbf{c}_{jk}^T \mathbf{x} \leq d_{jk}, \text{ for } k = 1, \cdots, m_j\right\}. \quad (2)$$

If there is no disturbance, we can formulate a path planning problem as an optimization to minimize the moving distance $d(\mathbf{x}^{t_0:t_1})$ with constraints $\mathbf{x}^{0:T} \Rightarrow \varphi$. Unfortunately, in the real world, there can be disturbances which can influence the state of the robot. The dynamic model under disturbances can be represented as: $\mathbf{x}_{t+1} = \mathbf{x}_t + f(\theta_t, \mathbf{u}_t) + \mathbf{w}_t$, where $\mathbf{w}_t \sim \mathcal{N}(0, \sigma^2\mathbf{I})$. Here, a simplifying assumption is made such that the disturbance influences the position, and not the heading of the robot. Then the robot position under the disturbance can be computed as:

$$\mathbf{x}_t \;=\; \mathbf{x}_0 + \sum_{k=0}^{t-1} f(\theta_k, \mathbf{u}_k) + \sum_{k=0}^{t-1} \mathbf{w}_k = \bar{\mathbf{x}}_t + \nu_t, \quad (3)$$

where $\bar{\mathbf{x}}_t = \mathbf{x}_0 + \sum_{k=0}^{t-1} f(\theta_k, \mathbf{u}_k)$ and $\nu_t \sim \mathcal{N}(0, \sigma(t)^2\mathbf{I})$. The position $\mathbf{x}_t$ follows the Gaussian distribution with mean $\bar{\mathbf{x}}_t$ and variance $\sigma(t)^2\mathbf{I}$. The mean position $\bar{\mathbf{x}}_t$ has the transition model $\bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t + f(\theta_t, \mathbf{u}_t)$ and a sequence of $\bar{\mathbf{x}}_t$ is denoted by $\bar{\mathbf{x}}^{0:T}$. The variance of disturbances increases as time passes and we have $\sigma^2(t) = t\sigma^2$. Since the trajectory of the robot under disturbances, $\mathbf{x}^{0:T}$, is a random variable, we cannot deterministically guarantee the mission completion or safety of $\mathbf{x}^{0:T}$.

Hence, we propose a new problem formulation to plan a path which minimizes the probability of mission failures and guarantees the probability of collision to be small.

$$\begin{aligned}
\text{minimize} \quad & \left[f_o(\bar{\mathbf{x}}^{0:T}),\ d(\bar{\mathbf{x}}^{0:T}))\right] && (4)\\
\text{subject to} \quad & \bar{\mathbf{x}}^{0:T} \Rightarrow \varphi,\ \bar{\mathbf{x}}_{t+1} = \bar{\mathbf{x}}_t + f(\theta_t, \mathbf{u}_t), && (5)\\
& \mathbf{x}_t \sim \mathcal{N}(\bar{\mathbf{x}}_t, \sigma(t)^2\mathbf{I}), && (6)\\
& P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon/m,\ \forall j,\ \forall t, && (7)
\end{aligned}$$

where $f_o(\bar{\mathbf{x}}^{0:T})$ is the upper bound on the probability of mission failures under disturbance for a given deterministic trajectory $\bar{\mathbf{x}}^{0:T}$. The deterministic trajectory $\bar{\mathbf{x}}^{0:T}$ satisfies the LTL mission as shown in the constraint (5). For given $\bar{\mathbf{x}}^{0:T}$, both of the mission failure probability and the moving distance are minimized in (4). The constraint (7) aims to bound the probability of collision with obstacles below ϵ.

A. Mission Failure Probability

When $\bar{\mathbf{x}}^{0:T} \Rightarrow \varphi$ is given, the upper bound on the probability that $\mathbf{x}^{0:T}$ does not accomplish the mission successfully is derived without considering obstacles. Suppose that $\bar{\mathbf{x}}^{0:T}$ completes the mission by visiting a sequence of n_φ regions of interest $R_{l(1)},\ R_{l(2)}, \cdots,\ R_{l(n_\varphi)}$, where $R_{l(i)}$ is the ith visited region of interest. Let A be the event $\mathbf{x}^{0:T} \Rightarrow \varphi$ and B be the event $\bar{\mathbf{x}}^{0:T} \Rightarrow \varphi$. We define T_i as a set of time indices when the robot stays at $R_{l(i)}$, i.e., $T_i = \left\{t | \bar{\mathbf{x}}_t \in R_{l(i)}\right\}$. The task of visiting the region $R_{l(i)}$ is denoted by π_i. The notation $\mathbf{x}^{T_i} \Rightarrow \pi_i$ denotes that the robot visits the region $R_{l(i)}$ once and successfully performs the task π_i during T_i, where $\mathbf{x}^{T_i} = \{\mathbf{x}_t | t \in T_i\}$. The event $\mathbf{x}^{T_i} \Rightarrow \pi_i$ is denoted by A_i. When the event B occurs, the probability of failure in accomplishing the mission is $P(A^c|B)$. This conditional mission failure probability can bounded by f_o as shown in the following theorem.

Theorem 1: If $\bar{\mathbf{x}}^{0:T} \Rightarrow \varphi$, the function $f_o(\bar{\mathbf{x}}^{0:T})$ is the upper bound on the failure probability of the mission, i.e., $P(A^c|B) \leq f_o(\bar{\mathbf{x}}^{0:T})$, where

$$f_o(\bar{\mathbf{x}}^{0:T}) \triangleq \sum_{i=1}^{n_\varphi} \min_{t \in T_i} \sum_{j=1}^{n_{l(i)}} \left(1 - \Phi\left(\frac{b_{l(i)j} - \mathbf{a}_{l(i)j}^T \bar{\mathbf{x}}_t}{\sigma(t)}\right)\right). \quad (8)$$

Proof: When the event B occurs, $\cup_{i=1}^n A_i^c$ contains A^c, since the event A contains the event $\cap_{i=1}^n A_i$. The upper bound on $P(A^c|B)$ is derived as follows,

$$P(A^c|B) \quad \leq \quad \sum_{i=1}^{n_\varphi} P(A_i^c|B) = \sum_{i=1}^{n_\varphi} (1 - P(A_i|B)). \quad (9)$$

Note that we omit the conditioning event B for notational convenience in the following discussion.

We can find the lower bound on $P(A_i)$ as follows:

$$P(A_i) = P(\vee_{t \in T_i}(\mathbf{x}_t \Rightarrow \pi_i)) \geq \max_{t \in T_i} P(\mathbf{x}_t \Rightarrow \pi_i). \quad (10)$$

By combining (9) and (10),

$$P(A^c) \leq \sum_{i=1}^{n_\varphi} (1 - \max_{t \in T_i} P(\mathbf{x}_t \Rightarrow \pi_i)). \quad (11)$$

The probability $P(\mathbf{x}_t \Rightarrow \pi_i)$ is equal to $P(\mathbf{x}_t \in R_{l(i)})$. Using (1), the lower bound on the probability $P(\mathbf{x}_t \Rightarrow \pi_i)$ can be computed as:

$$\begin{aligned} P(\mathbf{x}_t \Rightarrow \pi_i) &= P(\cap_{j=1}^{n_{l(i)}} \mathbf{a}_{l(i)j}^T \mathbf{x}_t \leq b_{l(i)j}) \\ &= 1 - P(\cup_{j=1}^{n_{l(i)}} (\mathbf{a}_{l(i)j}^T \mathbf{x}_t \leq b_{l(i)j})^c) \\ &\geq 1 - \sum_{j=1}^{n_{l(i)}} \left(1 - P(\mathbf{a}_{l(i)j}^T \mathbf{x}_t \leq b_{l(i)j})\right) \\ &= 1 - \sum_{j=1}^{n_{l(i)}} \left(1 - \Phi\left(\frac{b_{l(i)j} - \mathbf{a}_{l(i)j}^T \bar{\mathbf{x}}_t}{\sigma(t)}\right)\right), \end{aligned} \quad (12)$$

where $\mathbf{a}_{l(i)j}^T \mathbf{x}_t$ follows the Gaussian distribution $\mathcal{N}(\mathbf{a}_{l(i)j}^T \bar{\mathbf{x}}_t, \sigma(t)^2 \mathbf{I})$ and $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. By combining (11) and (12), we derive the desired result. ■

B. Feasibility

Under the disturbance, we assume that the state is feasible if it satisfies the chance constraint $P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon$. To simplify its computation, we use conservative constraints (7) based on [15]. If the chance constraint (7) is satisfied, the constraint $P(\mathbf{x}_t \in \mathcal{X}_{obs}) \leq \epsilon$ is satisfied since $P(\mathbf{x}_t \in \mathcal{X}_{obs}) \leq \sum_{j=1}^m P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon$. Hence, the probability of collision is bounded below ϵ at all times.

IV. Proposed Method

The proposed safe and robust path planning algorithm has a two-layered structure consisting of a high-level planner and a low-level planner. The high-level planner finds a path which can successfully perform the LTL mission given in (5). The low-level planner finds a path which can guarantee the safety requirement (7) and reduce both the mission failure probability and the moving distance (4).

Algorithm 1 Path planning

1: $\mathcal{A}_\varphi \leftarrow$ Automaton(φ)
2: $\mathcal{G} = (\mathcal{R}, E) \leftarrow$ Work Space Decomposition
3: $\mathcal{T}.\mathcal{V} \leftarrow \{\mathbf{x}_{init}, q_{init}\}, \mathcal{T}.\mathcal{E} \leftarrow \emptyset$
4: **while** the number of vertices$< N_{\max}$ **do**
5: $[Z_s, r_g] \leftarrow$ HighLevelPlanner($\mathcal{T}, G, A_\varphi$)
6: **if** $[Z_s, r_g]$ is not empty **then**
7: $\mathcal{T} \leftarrow$ LowLevelPlanner($\mathcal{A}_\phi, \mathcal{T}, \Gamma_{Z_s}, r_t$)
8: **end if**
9: **end while**
10: **if** $\exists v_i \in \mathcal{T}.\mathcal{V}$ s.t. $v_i.q \in \mathcal{A}_\phi.Q_{acc}$ **then**
11: $return\ \mathbf{u^{0:T-1}} \leftarrow$ OptimalPath($\mathcal{T}$)
12: **else**
13: $return\ null$
14: **end if**

A. Data Structure

1) Workspace Decomposition: For the high-level planner, the workspace is decomposed into a number of non-overlapping regions $\mathcal{R}$. The geometric adjacency among them is represented by a graph $G = (\mathcal{R}, E)$, with a set of edges $E = \{(r_i, r_j) | r_i, r_j \in \mathcal{R},\ r_i$ and r_j are adjacent to each other$\}$.

2) Search Tree: In the low-level layer, a trajectory is generated based on RRT. An RRT search tree $\mathcal{T}$ consists of vertices $\mathcal{T}.\mathcal{V}$ and edges $\mathcal{T}.\mathcal{E}$. The root of the tree contains the initial state of the robot. Each vertex $v \in \mathcal{T}.\mathcal{V}$ contains the state, the discrete decomposition region, the automaton state, and the path from the root to the current node. An edge $\mathcal{T}.\mathcal{E}$ between vertices i and j contains a control input $\mathbf{u}_{ij}$.

B. High-Level Planner

The high-level planner finds a discrete path which achieves the mission requirement in order to guide the low-level planner. The high-level planner is based on our previous work [18]. First of all, a DFA for φ is constructed and a workspace is decomposed into a finite number of regions (lines 1-2 in Algorithm 1). The state in the high-level planner is defined as $Z = \langle q, r \rangle$, where $r \in \mathcal{R}$ and $q \in A_\varphi.Q$ denote a decomposed region and an automaton state, respectively. The high-level planner determines a start state $Z_s = \langle q_s, r_s \rangle$ and a target region r_g, where there exists a feasible path from Z_s to r_g (line 5). The path induces q to reach the accepting state $A_\varphi.Q_{acc}$. Detail of the high-level planner is described in [18]. The low-level planner expands the tree based on the high-level plan (lines 6-8). The high-level planner and the low-level planner are executed iteratively until the number of vertices in the tree exceeds $N_{\max}$. In line 11, OptimalPath($\mathcal{T}$) selects a path with the minimum value of f_o among the paths that have reached the accepting state. When there are multiple paths with the same minimum value of f_o, OptimalPath($\mathcal{T}$) selects a path with the minimum travelled distance.

Algorithm 2 LowLevelPlanner($\mathcal{A}_\phi, \mathcal{T}, \Gamma_{Z_s}, r_t$)

1: $p_t \leftarrow$ SelectTargetPoint(r_t)
2: $v_s \leftarrow$ SelectStartVertex($\mathcal{T}, \Gamma_{Z_s}, p_t$)
3: $\zeta_{seg} \leftarrow Steer(v_s.x, p_t)$
4: **if** FeasibilityCheck(ζ_{seg}) **then**
5: $\quad \mathcal{T} \leftarrow$ UpdateTree($\mathcal{A}_\phi, \mathcal{T}, v_s, \zeta_{seg}$)
6: **end if**
7: return $\mathcal{T}$

C. Low-Level Planner

The structure of the low-level planner is shown in Algorithm 2. First, the target point and the vertex to be extended are selected based on the guidance of the high-level planner (line 1-2). The start vertex v_s and the target point p_t are connected by the path segment ζ_{seg} which is generated by $Steer$ function in [2] (line 3). If ζ_{seg} is feasible based on chance constraints (7), it is added to the search tree $\mathcal{T}$ (line 4-5) as CC-RRT does [3]. During tree expansion, the mission failure probability and the moving distance are minimized by modifying the rewiring step of RRT* [2].

1) Selection of a Target Point and Vertex to Extend: The target position is sampled from the decomposed region r_g computed by the high-level planner. If $r_g \notin R_{l(i)}$, the probability $P(\mathbf{x} \Rightarrow \pi_{l(i)})$ is close to zero and a target point is uniformly sampled from r_g. If $r_g \in R_{l(i)}$, we increase $P(\mathbf{x} \Rightarrow \pi_{l(i)})$ by sampling a target point from $\mathcal{N}(\mathbf{x}_R^i, \sigma_s^2 \mathbf{I})$, where $\mathbf{x}_R^i$ is a solution to

$$\max_{\mathbf{x}} P(\mathbf{x} \Rightarrow R_{l(i)}) \approx \max_{\mathbf{x}} \sum_{j=1}^{n_{l(i)}} \Phi\left(\frac{b_{l(i)j} - \mathbf{a}_{l(i)j}^T \mathbf{x}}{\sigma_s}\right).$$

The vertex to expand is the closest vertex to the target in Γ_{Z_s}, as RRT [1] does.

2) Check Feasibility: Once a path segment is generated, we check its feasibility if the path satisfies the condition (7) and its automaton state is feasible (line 4). From (2), the probability of collision at $\mathbf{x}_t$ can be approximated as:

$$\begin{aligned} &P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) = P(\cap_{k=1}^{m_j} \mathbf{c}_{jk}^T \mathbf{x}_t \le d_{jk}) \\ &\le \min_k P(\mathbf{c}_{jk}^T \mathbf{x}_t \le d_{jk}) = \min_k \Phi\left(\frac{d_{jk} - \mathbf{c}_{jk}^T \bar{\mathbf{x}}_t}{\sigma(t)}\right). \quad (13) \end{aligned}$$

If 13 is smaller than ϵ/m for all obstacles, the state $\mathbf{x}_t$ is feasible. Thus we can ensure the safety of the generated path.

3) Update Tree: While updating the tree, the mission failure probability is minimized by modifying the rewiring step in RRT* [2]. In Algorithm 3, the rewiring of RRT* finds a better solution by examining neighbors of the current vertex. In line 8, V_{near} indicates a set of neighboring vertices of $v_i.x$, where the radius of the neighborhood is determined as RRT*. In line 9-15, if a connection from v_i to $v_{near} \in V_{near}$ is feasible and improves the cost, $v_{near} \in V_{near}$ becomes a new parent of v_i by rewiring. Similarly, if a connection from $v_{near} \in V_{near}$ to v_i improves the cost, v_{near} becomes a child of v_i (line 17-25). In the original

Algorithm 3 UpdateTree($\mathcal{A}_\phi, \mathcal{T}, v_s, \zeta_{seg}$)

1: $v_{parent} = v_s$
2: **for** $\mathrm{x}_i \in \zeta_{seg} : \{\mathrm{x}_i \in \mathcal{X}\}$ **do**
3: $\quad v_i \leftarrow$ UpdateVertex($\mathcal{T}, \mathrm{x}_i$)
4: $\quad$ **if** $\langle v_i.q, v_i.r \rangle \notin Z_{feas}$ **then**
5: $\quad\quad$ break;
6: $\quad$ **end if**
7: $\quad \mathcal{T}.\mathcal{V} \leftarrow \mathcal{T}.\mathcal{V} \cup \{v_i\}$
8: $\quad v_{min} \leftarrow v_{parent}$, $V_{near} \leftarrow$ Near($v_i.x$)
9: $\quad d_{min} \leftarrow d(v_i.path)$, $f_{\min} \leftarrow f_o(v_i.path)$
10: $\quad$ **for each** $v_{near} \in V_{near} \setminus \{v_{parent}\}$ **do**
11: $\quad\quad d_{near} \leftarrow d(v_{near}.path)$, $f_{near} \leftarrow f_o(v_{near}.path)$
12: $\quad\quad$ **if** IsCostImproved(v_{near}, v_i) and FeasibilityCheck(edge(v_{near},v_i)) **then**
13: $\quad\quad\quad v_{min} \leftarrow v_{near}, d_{min} \leftarrow d_{near}, f_{min} \leftarrow f_{near}$
14: $\quad\quad$ **end if**
15: $\quad$ **end for**
16: $\quad v_{min} \leftarrow$ UpdateVertex($\mathcal{T}, v_{min}$),
17: $\quad \mathcal{T}.\mathcal{E} \leftarrow \mathcal{T}.\mathcal{E} \cup \{(v_{min}, v_i)\}$, $V'_{near} \leftarrow$ Near($v_i.x$)
18: $\quad$ **for each** $v_{near} \in V'_{near}$ **do**
19: $\quad\quad d_{new} \leftarrow d(v_{near}.path)$, $f_{new} \leftarrow f_o(v_{near}.path)$
20: $\quad\quad$ **if** IsCostImproved(v_{near}, v_i) and FeasibilityCheck(edge(v_i,v_{near})) **then**
21: $\quad\quad\quad v_{near} \leftarrow$ UpdateVertex($\mathcal{T}, v_{near}$)
22: $\quad\quad\quad \mathcal{T}.\mathcal{E} \leftarrow \mathcal{T}.\mathcal{E} \setminus \{$edge(Parent($v_{near}$), v_{near})$\}$
23: $\quad\quad\quad \mathcal{T}.\mathcal{E} \leftarrow \mathcal{T}.\mathcal{E} \cup \{$edge($v_i, v_{near}$)$\}$
24: $\quad\quad$ **end if**
25: $\quad$ **end for**
26: $\quad v_{parent} = v_i$
27: **end for**

RRT* algorithm, IsCostImproved in line 12 and 20 is true if the cost of v_{near} is smaller than the cost of v_i.

The RRT* algorithm cannot be directly applied to our problem since RRT* assumes that the cost function is monotonic. Our objective function, The mission failure probability (8), one of the objectives in our algorithm, is not a monotonic function and rewiring using this function can generate cycles in the tree [2]. A path is a continuous function $\xi : [0, 1] \to \mathbb{R}^d$ and a set of all paths is Ξ. Let $c : \xi \to \mathbb{R}_{\ge 0}$ be the cost function. For the optimality of RRT*, the cost function should be monotonic, in the sense that for all ξ_1, $\xi_2 \in \Xi$, $c(\xi_1) \le c(\xi_1|\xi_2)$ and bounded [2]. Then we prove that the objective function f_o is not a monotonic function in the longer version.

Since f_o is not a suitable function for RRT*, we use a moving distance as an additional cost function, which is monotonic. We let IsCostImproved in Algorithm 3 (line 15) be true if both inequalities below are satisfied:

$$f_o(v_{near}.path) \le f_o(v_i.path) + \beta \quad (14)$$
$$d(v_{near}.path) \le d(v_i.path). \quad (15)$$

Constraint (15) prevents the generation of a cycle in the tree since a cycle increases the moving distance. The parameter $\beta \in [0, 1]$ is added to adjust the importance between the moving distance and the mission failure probability. Then

we can find a trajectory with a small mission failure probability while reducing the moving distance. If $\beta = 0$, the upper bound of the mission failure probability of a planned trajectory does not increase as the tree expands. In addition, the upper bound on the mission failure probability is not reduced as the tree expands, the moving distance does not increase.

V. Experimental Results

We have conducted an extensive set of simulations and experiment to evaluate the performance of the proposed algorithm.

A. Simulation Study

The proposed method is evaluated for three scenarios shown in Figure 1. For the ith scenario, the robot aims to perform the corresponding mission φ_i, where

$$\begin{aligned} \varphi_1 &= \Diamond(a) \wedge \Diamond(c) \wedge \Diamond(b), \\ \varphi_2 &= \Diamond(a) \wedge \Diamond(c) \wedge \Diamond(b \wedge (\Diamond d)), \\ \varphi_3 &= \Diamond(a \wedge (\Diamond c \wedge (\Diamond b))). \end{aligned}$$

The mission φ_1 asks to "cover all regions of interest, a, b and c" (Figure 1(a)). The mission φ_2 is more complex and requires to "visit all regions of interest, a, b, c and d, but visit d after visiting b" (Figure 1(b)). In Figure 1(c), the robot accomplishes the mission φ_3 by visiting sequential regions in a complex environment. For each mission, we generated five paths by running the algorithm five times. Then, for each generated path, we have simulated the robot motion under the disturbance for $10,000$ times. The robot motion is simulated in open loop as it can represent more challenging mission scenarios in which state estimation is difficult, e.g., operations in a GPS-denied area with unreliable visual odometry or localization. In Figure 1(a),1(c), and 1(b), a red line indicates the planned trajectory and a blue line indicates an actual path taken by the robot under the disturbance.

The performance of the proposed method is compared to three multi-layered planning algorithms: multiple-layered RRT (ML-RRT), multiple-layered RRT* (ML-RRT*), and multiple-layered probabilistic RRT (ML-PrRRT). The proposed method is denoted by multiple-layered probabilistic RRT* (ML-PrRRT*). Four algorithms use the same high-level planner and choose a different low-level planner among RRT, RRT*, PrRRT and PrRRT*, respectively. ML-RRT and ML-RRT* aim to solve a typical path planning problem, which minimizes the moving distance without considering disturbances. ML-RRT extends an RRT search tree and finds a feasible path which accomplishes the mission. In the end, it selects a trajectory with the minimum moving distance. ML-RRT* employs RRT* to optimize the moving distance in the low-level layer. ML-PrRRT refers to our proposed method without rewiring. It extends the tree using RRT and selects a trajectory with the minimum value of f_o. It shows the importance of the objective function (8) and the constraints to avoid collision (7). ML-PrRRT* refers to the proposed method, which reduces the moving distance while trying to keep the mission success probability high in the low-level planner.

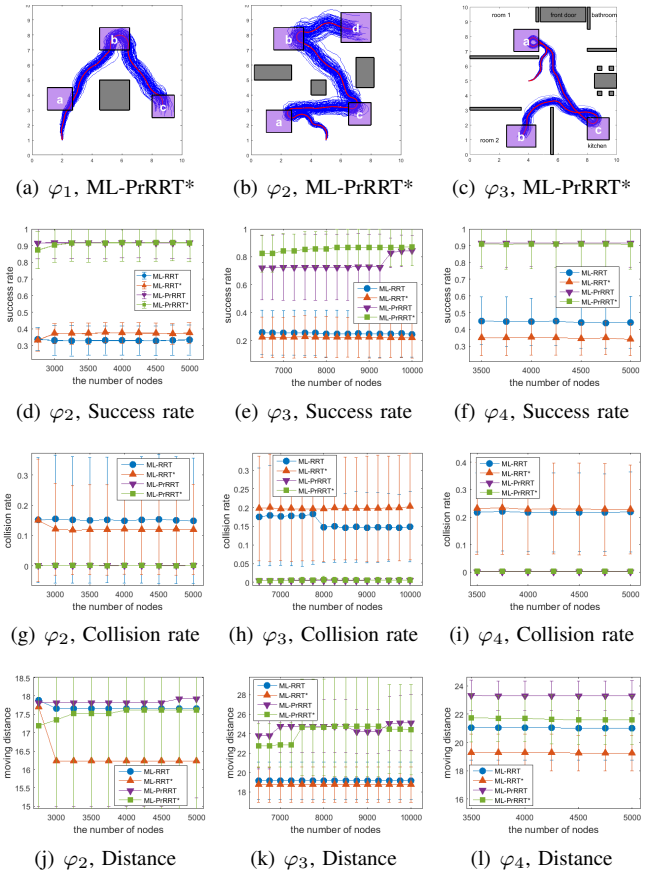
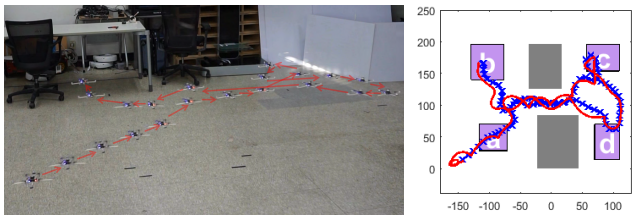


Fig. 1. Simulations with φ_1, φ_2, and φ_3. Red lines and blue lines in the first row show the planned trajectory and the actual trajectory of the robot under disturbance, respectively. For each mission, average success rates, collision rates and moving distance are shown corresponding to the number of nodes $N_{\max}$. The average and standard deviation values of each algorithm are computed from 5 independent trials (10,000 runs for each trial).

In Figure 1, the proposed method plans for missions, where we set $\sigma = 0.02$, $\beta = 0$, and $N_{\max} = 5000$. For three scenarios, the success rate, collision rate, and moving distance at different RRT tree sizes are shown in Figure 1. Success rates of all missions are shown in the second column, showing that the success rates of ML-PrRRT and ML-PrRRT* are much higher than those of ML-RRT and ML-RRT*. Collision rates shown in the third column demonstrate that the collision rates of ML-PrRRT and ML-PrRRT* are much less than those of ML-RRT and ML-RRT* as the chance constraints (7) reduce potential collisions. For better success rates, ML-PrRRT and ML-PrRRT* usually require more moving distances as shown in the fourth row of Figure 1. However, the proposed method ML-PrRRT* can reduce the moving distance while maintaining a high success rate. On the other hand, ML-RRT* does not have a better success rate than ML-RRT, since ML-RRT* reduce the moving distance without considering the mission failure probability. A shorter moving distance sometimes causes more collisions with obstacles as shown in Figure 1(g),



(a) (b)

Fig. 2. A Crazyflie experiment for mission φ_5. (a) An overlaid snapshot sequence from the experiment. (b) The red line is the planned trajectory and the blue line is the actual trajectory of the Crazyflie quadrotor.

1(h), and 1(i). For more complex mission φ_2, while a larger number of nodes in the tree are required by all algorithms to find any feasible trajectory, the proposed algorithm achieves good performance with a smaller number of nodes than other algorithms.

The required computation times of ML-RRT and ML-PrRRT with $N_{\max} = 1000$ are 1.70s and 1.96s, respectively, for mission φ_1. For $N_{\max} = 3000$, ML-RRT and ML-PrRRT require 5.08s and 5.91s, respectively. ML-RRT* and ML-PrRRT* require longer computation times: 11.52s and 18.35s for $N_{\max} = 1000$, respectively, and 79.11s and 87.78s for $N_{\max} = 3000$, respectively. The rewiring step in ML-RRT* and ML-PrRRT* demands a longer computation time to generate the same number of nodes but it can generate better trajectories.

B. Experiments

Physical experiments using a quadrotor have been conducted for the mission $\varphi_4 = a \wedge (\Diamond(c \wedge (\Diamond d \wedge (\Diamond b))))$ (see Figure 2). The quadrotor platform is Crazyflie 2.0 developed by Bitcraze. The quadrotor moves along the waypoints generated by the planner. We assume that the disturbance follows the Gaussian distribution $\mathbf{x}_t \sim \mathcal{N}(\bar{\mathbf{x}}_t, \sigma^2\mathbf{I})$, where $\mathbf{x}_t$ is the position of the quadrotor at time t. The altitude of the robot is set to the constant value. In experiments, this small quadrotor never moves exactly as planned as it gets disturbed by air current and other sources. The average standard deviation of disturbance is estimated from experiments as $\sigma = 0.118\ (m)$. We have conducted experiments 10 times using four different algorithms. If collisions are ignored, the success rates of completing the mission are 0.7, 0.8, 1.0, and 1.0, respectively. The collision rates of ML-RRT and ML-RRT* are 1.0 and 0.6 while ML-PrRRT and ML-PrRRT* does not cause collision with obstacles. The planned trajectories of ML-RRT and ML-RRT* are much closer to obstacles than the other two algorithms. The success rates of the mission without collision are 0.0, 0.3, 1.0, and 1.0, respectively. The experiment clearly shows the importance of considering disturbances when planning paths and demonstrates the safety and robustness of the proposed algorithm.

VI. Conclusion

In this paper, we have presented a robust and safe path planning algorithm for LTL missions under disturbances. When a robot cannot follow the planned trajectory exactly because of disturbances, the proposed method can guarantee a high probability of mission success. The problem is formulated to minimize the mission failure probability and the moving distance while satisfying safety constraints. Our novel multi-layered path planning algorithm finds a trajectory which completes the mission with a higher success rate and a shorter moving distance as demonstrated in simulations and field experiments.

References

[1] S. M. LaValle and J. James J. Kuffner, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, Jun. 2011.

[3] B. Luders, M. Kothari, and J. How, "Chance constrained rrt for probabilistic robustness to environmental uncertainty," in *Proc. of the AIAA Guidance, Navigation and Control Conference*, 2010.

[4] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based trajectory generation with linear temporal logic specifications," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014.

[5] A. Bhatia, L. E. Kavraki, and M. Y. Vardi, "Sampling-based motion planning with temporal goals," in *Proc. of the IEEE International Conference on Robotics and Automation*, May 2010.

[6] E. Plaku, *Planning in Discrete and Continuous Spaces: From LTL Tasks to Robot Motions*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012.

[7] O. Kupferman and M. Y. Vardi, "Model checking of safety properties," *Formal Methods in System Design*, vol. 19, pp. 291–314, 2001.

[8] J. Suh, J. Gong, and S. Oh, "Fast sampling-based cost-aware path planning with nonmyopic extensions using cross entropy," *IEEE Transactions on Robotics*, 2017. (To appear).

[9] S. Karaman and E. Frazzoli, "Linear temporal logic vehicle routing with applications to multi-uav mission planning," *International Journal of Robust and Nonlinear Control*, vol. 21, no. 12, pp. 1372–1395, 2011.

[10] G. E. Fainekos, A. Girard, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for dynamic robots," *Automatica*, vol. 45, no. 2, pp. 343 – 352, 2009.

[11] J. McMahon and E. Plaku, "Sampling-based tree search with discrete abstractions for motion planning with dynamics and temporal logic," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014.

[12] A. Bry and N. Roy, "Rapidly-exploring random belief trees for motion planning under uncertainty," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011.

[13] C. Yoo, R. Fitch, and S. Sukkarieh, "Provably-correct stochastic motion planning with safety constraints," in *Proc. of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2013.

[14] M. Lahijanian, S. B. Andersson, and C. Belta, "Temporal logic motion planning and control with probabilistic satisfaction guarantees," *IEEE Transactions on Robotics*, vol. 28, no. 2, pp. 396–409, April 2012.

[15] L. Blackmore and M. Ono, "Convex chance constrained predictive control without sampling," in *Proc. of the AIAA Guidance, Navigation and Control Conference*, August 2009.

[16] A. P. Sistla, "Safety, liveness and fairness in temporal logic," *Formal Aspects of Computing*, vol. 6, no. 5, pp. 495–511, Sep. 1994.

[17] C. Baier, J.-P. Katoen, and K. G. Larsen, Eds., *Principles of model checking*. MIT press Cambridge, 2008.

[18] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2308–2315, Oct. 2017.