# Robust Multi-Objective Path Planning for Flying Robots under Wind Disturbance

Yoonseon Oh, Kyunghoon Cho, and Songhwai Oh

*Abstract*— **This paper proposes a robust multi-objective path planning algorithm for flying robots carrying out complex missions. When a robot is put into the field, the robot is required to perform complex missions such as visiting sequential goals. We specify these missions using a linear temporal logic and search the path to accomplish the mission for flying robots. Since flying robots are more sensitive to air flow than ground robots, we should plan a path more carefully so that the disturbance by airflow does not cause mission failures or collision with obstacles. In addition, it is important to increase energy effectiveness for stability of flying robots. To achieve these purposes, we propose a multi-objective path planning problem which minimizes the mission failure probability and the moving distance while guaranteeing the safety of the robot and mission completion. We introduce a multi-layer path planning algorithm, where the high-level planner guides the low-level planner by generating a discrete path to accomplish the mission and the low-level planner searches the path to optimize multiple objective functions using a sampling-based RRT search tree. The presented low-level planner can improve the *Pareto* optimality of the trajectory effectively. We analyze the effectiveness theoretically and evaluate the performance by simulations.**

## I. Introduction

As robots begin to be used in everyday life, path planning is required to perform more complex tasks. While path planning deals with searching a feasible path from a start position to a goal region in the past [1]–[3], it has been extended to accomplish more complex missions. For example, some researchers are interested in accomplish a mission such as visiting sequential goals or covering multiple goals [4]–[7].

A complex mission of path planning can be represented using linear temporal logic (LTL) [8]. In recent years, some planning algorithms search a path to optimize a given cost function while performing a LTL mission. These algorithms formulate an optimization problem, where LTL missions are encoded by mixed-integer linear constraints [4], [9]. However, this approach suffers from difficulty in solving the problem when the number of LTL constraints is large or the dynamics of a robot is complicated. Hierarchical and sampling-based planning algorithms can efficiently incorporate LTL missions and robot dynamics [10]–[12].

In the real world, unfortunately, disturbances can make the planned path fail to accomplish the mission or collide with obstacles. When the model of the disturbances is available, chance constraints are employed to guarantee the safety of the planned trajectory [2], [13]. In order to robustly succeed in missions, probabilistic computation tree logic can be employed [14], [15]. While these methods solve the problem using a Markov decision process in a discrete space, we generate a robust path in a continuous space.

In this paper, we focus on planning a path for flying robots under disturbances. A flying robot can reduce the moving distance by considering the influence of the wind [16]. If a robot suffers from global positioning difficulties, our algorithm can be more powerful, since we can guarantee robust performance without knowing a global position of the robot. There are many such cases, for example, robots such as autonomous underwater vehicles or underground explorer robots are difficult to use a GPS system. In the case of a light indoor flying robot, it would be a burden to have a localization device. Hence, the proposed method generates a robust path against disturbances though we can not retrieve a global position.

In this paper, we assume that we can predict the wind speed and direction from the past data using Gaussian process regression. The disturbance is modelled as a Gaussian distribution which has a different mean and variance depending on the location and time, while our previous work [17] deals with the disturbance which has a constant distribution. Under the disturbances, we propose a path planning algorithm which performs an LTL mission with high probability and short moving distance. The algorithm has a hierarchical structure, where a high-level planner suggests a discrete plan satisfying the mission and a low-level planner builds a sampling-based search tree to optimize multiple objectives, the mission success probability and the moving distance. The proposed method improves the *Pareto* optimality of a planned trajectory by tree expansion.

This paper is structured as follows. Section II accounts for the concept of linear temporal logic in a path planning problem. In Section III, we describe the problem formulation for a path-planning problem which performs complex missions with safety and performance guarantees. We propose a planning algorithm and analyze its properties theoretically in Section IV. We evaluate our algorithm in simulations in Section V.

## II. Preliminaries

### A. Path Planning with Linear Temporal Logic

The mission of path planning is formulated by linear temporal logic (LTL). LTL formulas consist of a set of atomic propositions, boolean operators and temporal operators [8].

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B2006136). Y. Oh, K. Cho, 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, songhwai.oh}@cpslab.snu.ac.kr).

An atomic proposition denoted by $\pi_i$ is a statement which is either true or false. A set of atomic propositions is denoted by $\Pi = \{\pi_i, \ldots, \pi_N\}$. An LTL formula is a combination of propositions and temporal operators. Since we consider the path planning over a finite time, the mission is assumed to be a syntactically co-safe LTL formula (sc-LTL) [5], [18].

A robot moves in cluttered environments with obstacles and regions of interest. The workspace and the occupied region by obstacles are denoted by $\mathcal{X}$ and $\mathcal{X}_{obs}$, respectively. When a region of interest is $R_i$, an atomic proposition $\pi_i \in \Pi$ represents that the robot is in the region $R_i$. The atomic proposition $\pi_0$ is assumed to be true if the robot is in the free space, i.e., $\mathbf{x} \in \mathcal{X} \backslash ((\cup_i R_i \cup \mathcal{X}_{obs})$.

An LTL formula can be translated into a nondeterministic finite automation (NFA) by a number of open source software tools [8], [19]. For convenience, an NFA is converted to a deterministic finite automaton (DFA). The definition of a DFA $\mathcal{A}_\varphi$ over $2^\Pi$ with respect to $\varphi$ is 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

We assume that a flying robot moves on a 2D plane at a fixed altitude, $\mathcal{X} \subset \mathbb{R}^2$. The position of the robot at time $t$ is denoted by $\mathbf{x}_t = [x_t \ y_t]^T$ and the trajectory of the robot from time $t_0$ to $t_1$ is denoted by $\mathbf{x}^{t_0:t_1}$. We consider a path planning problem to perform the mission $\varphi$ in cluttered environments with $m$ obstacles. The mission is visiting several regions of interest $\{R_i\}$ in a certain order while avoiding obstacles. The achievement of the mission $\varphi$ is denoted by $\mathbf{x}^{t_0:t_1} \Rightarrow \varphi$. In this paper, we assume that the shape of regions of interest and obstacles is convex polygons. The region of interest $R_i \subset \mathcal{X}$ is represented by 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 $j$th side of $R_i$ is denoted by $\mathbf{a}_{ij} \in \mathbb{R}^2$ and the distance between the $j$th side and the origin $(0, 0)$ is denoted by a scalar $b_{ij}$. The occupied region by the $j$th obstacle is denoted by $\mathcal{X}_{obs}^j \subset \mathcal{X}$ and represented by an $m_j$-sided polygon consisting of the outward normal vector $\mathbf{c}_{jk} \in \mathbb{R}^2$ and distances $d_{jk}$, i.e., $\mathbf{x} \in \mathcal{X}_{obs}^j$

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

The occupied region by a set of obstacles is denoted by $\mathcal{X}_{obs} = \cup_{j=1}^m \mathcal{X}_{obs}^j$.

We assume that the following dynamic model for a robot, $\mathbf{x}_{t+1} = A_t \mathbf{x}_t + B_t \mathbf{u}_t + \mathbf{w}(\mathbf{x}_t, t)$, where $\mathbf{u}_t = [u_x(t), u_y(t)]^T$ is a control input to the robot in $xy$-direction . The notation $\mathbf{w}(\mathbf{x}_t, t)$ refers a time-varying disturbance by the wind. We assume that the disturbance follows a Gaussian distribution, $\mathcal{N}(\mu_w(\mathbf{x}_t, t), \Sigma_w(\mathbf{x}_t, t))$. We approximate the distribution of the position of the robot $\mathbf{x}_t$ as a Gaussian distribution $\mathcal{N}(\hat{\mathbf{x}}_t, \Sigma_t)$. The mean and covariance of $\mathbf{x}_t$ are estimated as follows,

$$\begin{aligned} \hat{\mathbf{x}}_{t+1} &= A_t \hat{\mathbf{x}}_t + B_t \mathbf{u}_t + \mathbb{E}[\mathbf{w}(\mathbf{x}_t, t)] \\ &\approx A_t \mathbf{x}_t + B_t \mathbf{u}_t + \mu_w(\hat{\mathbf{x}}_t, t) \\ \Sigma_{t+1} &\approx A_t Cov[\mathbf{x}_t] A_t^T + Cov[\mathbf{w}(\mathbf{x}_t, t)]. \end{aligned}$$

Since $\mathbf{x}_t$ follows the Gaussian distribution, we approximate $Cov[\mathbf{w}(\mathbf{x}_t, t)]$ as the covariance matrix with the maximum uncertainty near the mean of $\mathbf{x}_t$. We let $B(\hat{\mathbf{x}}_t, r)$ be a ball where the center is $\hat{\mathbf{x}}_t$ and the radius is $r$ which is proportional to the variance. Then the covariance matrix is derived as $Cov[\mathbf{w}(\mathbf{x}_t, t)] \approx \Sigma_w(\hat{\mathbf{x}}_t^*, t)$, where $\hat{\mathbf{x}}_t^* = argmax_{\mathbf{x} \in B(\hat{\mathbf{x}}_t, r)} tr(\Sigma_w(\mathbf{x}, t))$ and we approximate the value by sampling.

We now provide the problem formulation for generating a robust and efficient trajectory with the given mission. For robustness, we minimize the probability that the robot fails to achieve the mission. At the same time, we minimizes the moving distance to improve the efficiency of the planned trajectory. The moving distance of trajectory $\mathbf{x}^{0:T}$ is defined as $d(\mathbf{x}^{0:T}) = \sum_{t=0}^{T-1} ||\mathbf{x}_{t+1} - \mathbf{x}_t||_2$. We minimize the length of the trajectory which is shifted by disturbances $d(\hat{\mathbf{x}}^{0:T})$. Then the problem is formulated as an optimization problem with dual objective functions as follows:

$$\begin{aligned} \text{minimize} \quad & \left[P(\mathbf{x}^{0:T} \not\Rightarrow \varphi | \mathbf{u}^{0:T-1}), \ d(\hat{\mathbf{x}}^{0:T}))\right] \\ \text{subject to} \quad & \hat{\mathbf{x}}^{0:T} \Rightarrow \varphi, \ \mathbf{x}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \Sigma_t), \quad (3) \\ & P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon / m, \ \forall j, \ \forall t. \quad (4) \end{aligned}$$

The mean of the disturbed trajectory $\hat{\mathbf{x}}^{0:T}$ achieves the LTL mission by the constraint (3). The constraint (4) aims to bound the probability of collision with obstacles below $\epsilon$.

### A. Air Flow Estimation

The position of a flying robot is influenced by the air flow which is represented by a vector field. We use wind patterns across the United States from National Oceanic and Atmospheric Administration (NOAA). Let the disturbance by air flow be $\mathbf{w}(\mathbf{x}, t)$ which is a function of the position and time. The observed disturbance is represented by a vector, $\mathbf{w}_t^i = [w_{x_t}^i, w_{y_t}^i]$. The $w_{x_t}^i$ is the east-west component of wind and $w_{y_t}^i$ is the north-south component of the wind. We have a dataset $\mathcal{D}_k$ which contains $n^d$ observations at time $t_k^d$, i.e. $D_k = \{(\mathbf{x}_t^i, \mathbf{w}_t^i) | i = 1, \ldots, n_d, t = t_k^d\}$. The union of subsets $\mathcal{D}_k$ is denoted by $\mathcal{D}$, where the number of subsets is $N_d$. Let $\mathbf{X}_k = \{\mathbf{x}_t^1, \ldots, \mathbf{x}_t^{n_d} | t = t_k^d\}$ and $\mathbf{w}_x(k) = [w_{x_t}^1, \ldots, w_{x_t}^{n_d}]^T$, where $t = t_k^d$. Let $\mathbf{w}_y(k) = [w_{y_t}^1, \ldots, w_{y_t}^{n_d}]^T$. We assume that $xy$ components of disturbances are independent and we predict $\mathbf{w}^* = [w_x^*, w_y^*]$ for a new input $\mathbf{x}_*$ using Gaussian process regression [20] at $t_k^d$. Then the conditional distribution of $\mathbf{w}^*$ is estimated by a following Gaussian distribution: $\mathbf{w}^* \sim \mathcal{N}(\mu_w(\mathbf{x}_*, t_k^d), \Sigma_w(\mathbf{x}_*, t_k^d))$.

### B. Mission completion

The LTL mission can be translated into an NFA. Let the automaton state of the trajectory $\mathbf{x}^{t_0:t_1}$ be $q(\mathbf{x}^{t_0:t_1})$.

While the robot is moving, visiting regions of interest causes transitions of the automaton state. If the trajectory completes the mission, the automaton state of the trajectory becomes $Q_{acc}$. We denote the number of transitions from $q_0$ to $Q_{acc}$ as $d^{\mathcal{A}}(q_0)$. Then the constraint of mission completion (3) is equal to $q(\hat{\mathbf{x}}^{0:T}) = Q_{acc}$ or $d^{\mathcal{A}}(q(\hat{\mathbf{x}}^{0:T})) = 0$.

### C. Mission Failure Probability

In this section, we derive the upper bound on the mission failure probability $P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\mathbf{u}^{0:T-1})$. Suppose that a robot needs to visit $n_\varphi$ regions of interest $R_1, \cdots, R_{n_\varphi}$ in order to complete a mission $\varphi$. We define $T_i$ as a set of time indices when the robot is planned to stay at $R_i$, i.e., $T_i = \{t|\hat{\mathbf{x}}_t \in R_i\}$. Let the notation $\mathbf{x}^{T_i} \Rightarrow \pi_i$ denote that the robot visits the region $R_i$ 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 $A_i$.

First, we propose the upper bound on the probability that the automaton state of the disturbed trajectory is different from the automaton state of the mean trajectory in the following theorem:

*Theorem 1:* $P\left(q(\mathbf{x}^{0:T}) \neq q(\hat{\mathbf{x}}^{0:T})|\hat{\mathbf{x}}^{0:T}\right) \leq f_o(\hat{\mathbf{x}}^{0:T}))$, where $f_o(\hat{\mathbf{x}}^{0:T}) \triangleq \sum_{i=1}^{n_\varphi} f_o^i(\hat{\mathbf{x}}^{0:T})$ and $f_o^i(\hat{\mathbf{x}}^{0:T})$

$$= \begin{cases} \min_{t\in T_i} \sum_{j=1}^{n_i} \left(1 - \Phi\left(\frac{b_{ij} - \mathbf{a}_{ij}^T \hat{\mathbf{x}}_t}{\sigma_{ij}(t)}\right)\right) & T_i \neq \emptyset \\ 0 & T_i = \emptyset \end{cases}.$$

*Proof:* When we denote the event $q(\mathbf{x}^{0:T}) = q(\hat{\mathbf{x}}^{0:T})$ as $A$, $A$ contains $\cap_{i:T_i\neq\emptyset} A_i$. Then the upper bound on $P(A^c)$ is derived as follows:

$$P(A^c) \leq \sum_{i:T_i\neq\emptyset} (1 - \max_{t\in T_i} P(\mathbf{x}_t \Rightarrow \pi_i)). \tag{5}$$

The detail derivation is described in [17]. Using (1), the lower bound on the probability $P(\mathbf{x}_t \Rightarrow \pi_i)$ can be computed as:

$$P(\mathbf{x}_t \Rightarrow \pi_i) \geq 1 - \sum_{j=1}^{n_i} \left(1 - \Phi\left(\frac{b_{ij} - \mathbf{a}_{ij}^T \hat{\mathbf{x}}_t}{\sigma_{ij}(t)}\right)\right), \tag{6}$$

where $\mathbf{a}_{ij}^T \mathbf{x}_t$ follows the Gaussian distribution $\mathcal{N}(\mathbf{a}_{ij}^T \hat{\mathbf{x}}_t, (\sigma_{ij}(t))^2)$. The variance is $(\sigma_{ij}(t))^2 = \mathbf{a}_{ij}^T \Sigma_t \mathbf{a}_{ij}$ and $\Phi(x)$ is the cumulative distribution function of the standard normal distribution. By combining (5) and (6), we derive the desired result. ■

Using Theorem 1, we can derive the upperbound on the mission failure probability as follows,

*Theorem 2:* When $q(\hat{\mathbf{x}}^{0:T}) = Q_{acc}$, the following inequality holds: $P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\mathbf{u}^{0:T-1}) \leq f_o(\hat{\mathbf{x}}^{0:T})$.

*Proof:* Since $\hat{\mathbf{x}}$ is deterministically derived for the given $\mathbf{u}$, we can derive following equalities:

$$\begin{aligned} P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\mathbf{u}^{0:T}) &= P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\hat{\mathbf{x}}^{0:T}) \\ &= P(q(\mathbf{x}^{0:T}) \neq q(\hat{\mathbf{x}}^{0:T})|\hat{\mathbf{x}}^{0:T}) \end{aligned} \tag{7}$$

By using Theorem 1 and (7), the theorem is proved. ■

In this paper, we find the trajectory using a tree search algorithm and many paths in the tree do not accomplish the mission. It makes difficult to compute $P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\mathbf{u}^{0:T-1})$ for all paths in tree. Instead, we decompose the objective function $P(\mathbf{x}^{0:T} \not\Rightarrow \varphi|\mathbf{u}^{0:T-1})$ into $f_o(\hat{\mathbf{x}}^{0:T})$ and $d^{\mathcal{A}}(q(\hat{\mathbf{x}}^{0:T}))$. Minimizing $d^{\mathcal{A}}(q(\hat{\mathbf{x}}))$ induces the tree to find a path to perform a mission. Minimizing $f_o^i(\hat{\mathbf{x}}^{0:T})$ increases robustness of the planned trajectory. Then the problem can be reformulated as follows:

$$\text{minimize} \quad \left[d^{\mathcal{A}}(q(\hat{\mathbf{x}}^{0:T})), f_o^i(\hat{\mathbf{x}}^{0:T}),\ d(\hat{\mathbf{x}}^{0:T})\right],\ \forall i \tag{8}$$

$$\text{subject to} \quad \hat{\mathbf{x}}^{0:T} \Rightarrow \varphi,\ \mathbf{x}_t \sim \mathcal{N}(\hat{\mathbf{x}}_t, \Sigma_t),$$
$$P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon/m,\ \forall j,\ \forall t. \tag{9}$$

### D. Feasibility

To guarantee the safety of the planned trajectory under disturbances, the constraint (9) is employed. The probability of collision with obstacles is bounded as $P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \epsilon$ and it is derived in [17]. From (2), the probability of collision with the $j$th obstacle can be approximated as:

$$P(\mathbf{x}_t \in \mathcal{X}_{obs}^j) \leq \min_k \Phi\left(\frac{d_{jk} - \mathbf{c}_{jk}^T \hat{\mathbf{x}}_t}{\sqrt{\mathbf{c}_{jk}^T \Sigma_t \mathbf{c}_{jk}}}\right). \tag{10}$$

If (10) is smaller than $\epsilon/m$ for all obstacles, the state $\mathbf{x}_t$ is feasible. Thus we can ensure the safety of the generated path.

## IV. Proposed Method

The proposed method consists of a high-level planner and a low-level planner. The high-level planner generates a discrete path to perform a mission successfully and the generated path guides the tree expansion of the low-level planner. The low-level planner considers multiple objective functions such as the number of transitions towards acceptance automaton states, the mission failure probability, and the moving distance. It improves optimality of these functions by modification of rewiring in RRT* [1]. We construct multiple trees with various important weights of objective functions and the best path is selected from trees.

### A. Data Structure

*1) Workspace Decomposition:* For the high-level planner, we decompose the workspace $\mathcal{X}$ into regions of interest and a number of non-overlapping free regions $R'_1, \cdots, R'_l$. The decomposed workspace is denoted by $\mathcal{R} = \{R'_1, \cdots, R'_l, R_1, \cdots, R_n\}$. We construct a graph to represent geometric adjacency among decomposed regions: $G = (\mathcal{R}, E)$, 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 of the robot $\mathbf{x} \in \mathcal{X}$ (denoted by $v.x$), the discrete decomposition region $r \in \mathcal{R}$ (denoted by $v.r$), the automaton state $q \in Q$ (denoted by $v.q$), and the path from the root to the current node (denoted by $v.path$). An edge $\mathcal{T}.\mathcal{E}$ between vertices $i$ and $j$ contains a control input $\mathbf{u}_{ij}$.

**Algorithm 1** Path planning

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

### B. High-Level Planner

The high-level planner guides the low-level planner to find a path which satisfies the mission. We employ the high-level planner described in our previous work [17]. First, we construct a DFA for the mission and decompose the workspace (line 1 in Algorithm 1). The high-level state consists of the decomposed region $r \in R$ and the automaton state $q \in \mathcal{A}_\varphi.Q$. 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). When we define a set of vertices of which high-level state is $Z' = \langle q', r' \rangle$ in the tree as $\Gamma_{\langle q', r' \rangle} = \{v \in T.\mathcal{V} | v.q = q', v.r = r'\}$, the low-level planner expands the tree from a vertex in $\Gamma_{Z_s}$ towards a point in $r_g$ (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 12, OptimalPath($\mathcal{T}$) selects the *Pareto* optimal path from trees.

### C. Low-Level Planner

The low-level planner selects the target point and the vertex to be extended based on the guidance of the high-level planner. The target position is sampled from the decomposed region $r_g$ and the vertex to expand is the closest vertex to the target position in $\Gamma_{Z_s}$, as RRT [3] does. The start vertex $v_s$ and the target point $p_t$ are connected by the path segment $\zeta_{seg}$. If $\zeta_{seg}$ is feasible based on chance constraints (9), it is added to the search tree $\mathcal{T}$ as CC-RRT does [2]. We compute the collision probability based on (10) and if it is less than the threshold, we reveal that the path is safe and add the path into the tree. The tree searches better solutions by modifying the rewiring step of RRT* [1].

*1) Update Tree to find the Pareto optimal solution:* We denote a feasible path as $\xi \in \Xi$, where $\Xi$ is a set of all feasible paths in the workspace. To generalize algorithms, we assume that there are $n_f$ objective functions, $f_1,\ f_2, \cdots,\ f_{n_f}$, where $f_i : \Xi \mapsto \mathbb{R}$. The objective function is not assumed to be monotonic and a vector of multiple objective functions is denoted by $f_p(\xi) = [f_1(\xi), \cdots, f_{n_f}(\xi)]^T$. We aim to minimize the objective functions by tree expansion. The cost of the path $\xi$, $f_p(\xi)$, strictly dominates $f_p(\xi')$, if for all $j$, $f_j(\xi) \leq f_j(\xi')$ is satisfied and there exists $j$ which satisfies $f_j(\xi) < f_j(\xi')$. It is written as $f_p(\xi') \prec f_p(\xi)$. The $\xi^*$ is the *Pareto optimal path* if $f_p(\xi^*)$ strictly dominates the cost of any other paths, i.e. $f_p(\xi') \prec f_p(\xi^*)$, for all $\xi' \in \Xi$ and $\xi^* \neq \xi'$. If there is not the *Pareto optimal path*, we can select a different path as the best path depending on the priority of multiple objective functions. We define a *Pareto path set* as a set of paths which any path do not strictly dominate, i.e.,

$$\Xi^* = \{\xi \in \Xi | \{\xi' \in \Xi |\ f_p(\xi) \prec f_p(\xi'),\ \xi \neq \xi'\} = \emptyset\}.$$

**Algorithm 2** 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: $\quad \mathcal{T}.\mathcal{V} \leftarrow \mathcal{T}.\mathcal{V} \cup \{v_i\}$
5: $\quad v_{min} \leftarrow v_{parent}$, $V_{near} \leftarrow$ Near($v_i.x$)
6: $\quad d_{min} \leftarrow d(v_i.path)$, $f_{\min} \leftarrow f_o(v_i.path)$
7: $\quad$ **for each** $v_{near} \in V_{near} \setminus \{v_{parent}\}$ **do**
8: $\quad\quad d_{near} \leftarrow d(v_{near}.path)$, $f_{near} \leftarrow f_o(v_{near}.path)$
9: $\quad\quad$ **if** IsCostImproved($v_{near}, v_i$) and FeasibilityCheck(edge($v_{near}$,$v_i$)) **then**
10: $\quad\quad\quad v_{min} \leftarrow v_{near}, d_{min} \leftarrow d_{near}, f_{min} \leftarrow f_{near}$
11: $\quad\quad$ **end if**
12: $\quad$ **end for**
13: $\quad v_{min} \leftarrow$ UpdateVertex($\mathcal{T}, v_{min}$)
14: $\quad \mathcal{T}.\mathcal{E} \leftarrow \mathcal{T}.\mathcal{E} \cup \{(v_{min}, v_i)\}$, $v_{parent} = v_i$
15: **end for**

The low-level planner searches the *Pareto optimal path* in $\Xi$ by modifying the rewiring step in RRT* [1]. The rewiring improves the solution by examining neighbors of the current target vertex. In the original RRT*, $v_i$ becomes a new parent of $v_{near} \in V_{near}$, if the rewiring from $v_i$ to $v_{near}$ improves the cost. However, since the objective functions we consider are not limited to monotonic functions, the procedure can generate cycles in the tree [1] and hence we eliminate the procedure. To consider multiple objective functions, we propose IsCostImproved in Algorithm 2 which is true if following inequalities are satisfied:

$$\forall j \in J,\ f_j(v_{near}.path)) \leq f_j(v_i.path) + \beta_j, \tag{11}$$
$$\exists j \in J,\ f_j(v_{near}.path)) < f_j(v_i.path), \tag{12}$$

where $\beta_j$ is a non-negative real parameter and $J = \{1, \cdots, n_f\}$. If all $\beta_j$s are zeros, we execute rewiring when a new connection strictly dominates the previous connection. Otherwise, rewiring occurs if there is at least one objective function that is better for the new connection. The parameters $\beta_j$ in the constraint (11) prevents the other objective functions which does not satisfy (12) from getting much worse. We expands multiple trees with different parameters $\beta$. The procedure improves the effectiveness to improve the *Pareto path set* of trees and the detail is described in the next section.

### D. Analysis

This section analyze several properties of the proposed algorithm. Let a set of all paths in the tree $\Xi_n$, where $n$ is

the number of nodes in the tree.

*Theorem 3:* The *Pareto path set* of $\Xi_n^*$ is not getting worse as the tree expands.

*Proof:* By the definition, the *Pareto path set* of $\Xi_{n+1}$ is defined by $\Xi_{n+1}^* = \{\xi \in \Xi_{n+1} |\ \{\xi' \in \Xi_{n+1} |\ f_p(\xi) \prec f_p(\xi'),\ \xi \neq \xi'\} = \emptyset\}$. Since we do not eliminate any paths in the tree during tree expansion, $\Xi_n$ and $\Xi_n^*$ are subsets of $\Xi_{n+1}$. Then for all $\xi \in \Xi_{n+1}^*$, the following equation holds: $\{\xi' \in \Xi_n^* \subset \Xi_{n+1} |\ f_p(\xi) \prec f_p(\xi'),\ \xi \neq \xi'\} = \emptyset$. Hence, there is not any path in $\Xi_n^*$ which dominates the paths in $\Xi_{n+1}^*$. ■

Let $\Xi_{new}$ be a set of all possible paths that can be newly generated by tree expansion from $\Xi_n$ to $\Xi_{n+1}$. The generated path without rewiring is denoted by $\xi_n \in \Xi_{new}$ and the generated path with rewiring is denoted by $\xi_n^* \in \Xi_{new}$.

*Theorem 4:* If $\beta_i = 0$ for all $i$, the probability that there is improvement in $\Xi_n^*$ by $\xi_n^*$ is larger than that by $\xi_n$.

*Proof:* By (11) and (12)

$$f_p(\xi_n^*) = \begin{cases} f_p(\xi_n') & \exists \xi_n',\ f_p(\xi_n) \prec f_p(\xi_n'),\ \xi_n' \neq \xi_n \\ f_p(\xi_n) & o.w. \end{cases} \quad (13)$$

the probability that there is improvement in $\Xi_n^*$ by $\xi_n^*$ is formally defined as follows: $P = P\left(\cup_{\xi \in \Xi_n^*} (f_p(\xi) \prec f_p(\xi_n^*))\right)$. Since $\{\xi \in \Xi_n^* | f_p(\xi) \prec f_p(\xi_n)\}$ is the subset of $\{\xi \in \Xi_n^* | f_p(\xi) \prec f_p(\xi_n^*)\}$, we derive the following inequalities: $P \geq P(\cup_{\xi \in \Xi_n^*} (f_p(\xi) \prec f_p(\xi_n))$.Hence, if $\beta_i = 0$, we can guarantee that tree expansion with rewiring is more efficient to improve the optimality than tree expansion without rewiring is. ■

The multi-objective problem can be decomposed into a set of subproblems [21]. The importance of objective functions is represented by weights $\lambda = [\lambda_1, \cdots, \lambda_{n_f}]$, where $\sum_1^{n_f} \lambda_i = 1$ and $\lambda_i \geq 0$. Then the multi-objective optimization problem can be approximated into an optimization problem with a single objective function $g_w(\xi, \lambda) = \sum_{j=1}^{n_f} \lambda_j f_j(\xi)$, where $\xi$ is a path [21]. As many as subproblems with different $\lambda$ we solve, we can find a better *Pareto path set* with various importance weights.

*Theorem 5:* If rewiring occurs, there always exists a subproblem where $\xi_n^*$ is a better solution than $\xi_n$. That is, $\exists \lambda,\ g_w(\xi_n^*, \lambda) < g_w(\xi_n, \lambda)$.

*Proof:* If rewiring does not occur, $g_w(\xi_n^*) = g_w(\xi_n)$. When the rewiring occurs, we denote the index of the objective function which satisfies the constraint (12) as $j^*$. By the definition, $\Delta g_w = g_w(\xi_n^*, \lambda) - g_w(\xi_n, \lambda) = \sum_{j=1}^{n_f} \lambda_j (f_j(\xi_n^*) - f_j(\xi_n)) < \sum_{j \neq j^*} \lambda_j \beta_j + \lambda_{j^*}(f_j(\xi_n^*) - f_j(\xi_n))$. If there exists $\lambda$ so that

$$\sum_{j \neq j^*} \lambda_j \beta_j + \lambda_{j^*}(f_{j^*}(\xi_n^*) - f_{j^*}(\xi_n)) \leq 0, \quad (14)$$

the theorem is completely proved for $g_w$. When $\beta^* = \max_{j \neq j^*} \beta_j$ and $\Delta f_j = f_j(\xi_n^*) - f_j(\xi_n)$, we can derive the conservative constraint of (14):

$$\sum_{j \neq j^*} \lambda_j \beta_j + \lambda_{j^*} \Delta f_{j^*} \quad \leq \quad \sum_{j \neq j^*} \lambda_j \beta^* + \lambda_{j^*} \Delta f_{j^*} \leq 0.$$

Then for all subproblems with $\lambda_{j^*} \geq \frac{\beta^*}{\beta^* - \Delta f_{j^*}}$, $\Delta g_w$ is less than zero. Since $\Delta f_{j^*} < 0$, $\frac{\beta^*}{\beta^* - \Delta f_{j^*}}$ is less than 1 and then $\lambda$ always exists. We can improve various subproblems with a given single parameter $\beta$. ■

## V. Simulations

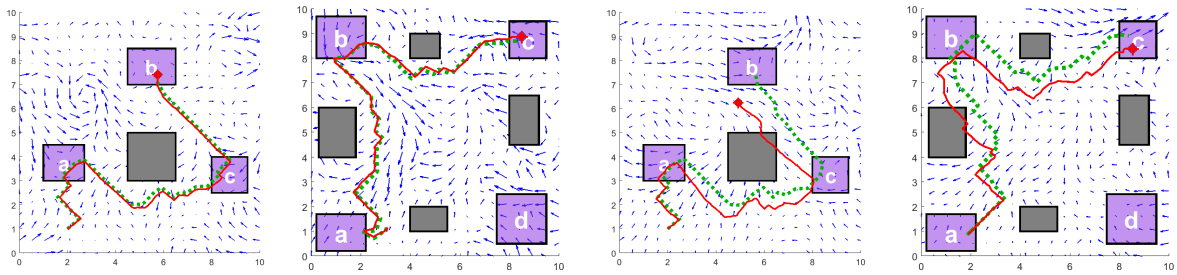
We have evaluated the proposed method by extensive simulations. In Figure 1, we give a flying robot missions when there are time-varying disturbances by wind. We let the dynamic model be $A_t = \mathbf{I}_{2\times 2}$ and $B_t = \mathbf{I}_{2\times 2}$. We assume that the air flow is estimated based on NOAA dataset[1] which has directions and velocities of wind. The blue arrows represent wind flows and we estimate the flow using Gaussian process as described in Section III-A. Given missions are as follows: $\varphi_1 = \diamond(a \wedge \diamond(b) \wedge \diamond(c))$, $\varphi_2 = \diamond(a \wedge \diamond(b \wedge \diamond(c))) \vee \diamond(a \wedge \diamond(d \wedge \diamond(c)))$. The first mission is to "visit regions of interest $b$ and $c$ after visiting $a$. The second mission is to evaluate the proposed algorithm in the more complex environments in Figure 1(b).

If we generate a path without taking into account the disturbance, the robot will move along a completely different path from the planned path. We generate a path using ML-RRT [17] which does not consider the disturbance. Then we let the robot follow the trajectory represented by the green dashed line under air flow in Figure 1(c) and 1(d). Then the robot moves along the red solid line which fails to accomplish the mission and collides with obstacles. Since the disturbed trajectory can cause serious dangers, our algorithm plans a robust path against disturbances by considering air flow.

The low-level planner constructs ten trees with random parameters $\beta$ and computes the multiple objective functions of nodes in the tree. The mission completion has a top priority over other objective functions and the mission failure probability is assumed to have priority over the moving distance. Among paths reaching the acceptance goal in trees, we select a path to be deployed according to the priority of objective functions. When the number of nodes in each tree is $2500$, the selected trajectory is represented by a green dashed line in Figure 1. Then we deploy the flying robot to follow the path under the time-varying wind field. The trajectory of the flying robot disturbed by the wind is represented by the red line and the disturbance is accumulated as the robot moves. The robot successfully performs the mission as shown in the figure. In addition, the robot keeps distance from obstacles and hence we can guarantee the safety of the path under disturbances.

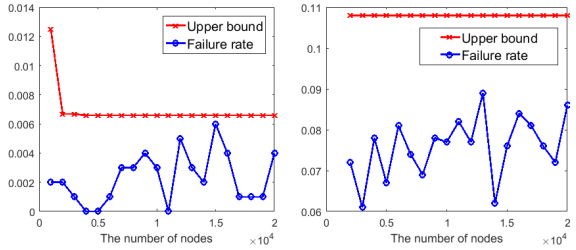
We deploy the robot $5 \times 10^2$ times under the disturbance to evaluate the robustness of the algorithm. In Figure 2, the red line is the upper bound on the mission failure probability of the planned trajectory. The upper bound decreases as tree expands as we show in Theorem 3. The blue line is the failure rate when trajectories are disturbed by wind. The failure rate

[1]National Oceanic and Atmospheric Administration, U.S., https://www.noaa.gov/




(a) $\varphi_1$, proposed method (b) $\varphi_2$, proposed method (c) $\varphi_1$, ML-RRT (d) $\varphi_2$, ML-RRT

Fig. 1. (a), (b) A flying robot accomplishes missions $\varphi_1$ and $\varphi_2$ while moving under disturbances by air flow which are shown as blue arrows. (c), (d) Trajectories for missions $\varphi_1$ and $\varphi_2$ are planned without considering the wind flow. The red lines are trajectories of the robot and the green lines represent the planned trajectories.




(a) $\varphi_1$ (b) $\varphi_2$

Fig. 2. Mission failure rate when the planned trajectory is disturbed by wind. The red line is the upper bound on the falilure probability and the red line represents the failure rate by simulations.

derived by simulations is well bounded by the theoretical upper bound in Theorem 2.

## VI. Conclusion

We introduce a robust and safe path planning so that a flying robot achieves LTL missions under time-varying and location-dependent disturbances. The path planning problem is formulated as an optimization problem with multiple objective functions in order to minimize the failure probability of missions and moving distance while completing the mission. We propose a multi-layer planning algorithm, where a high-level planner searches a discrete path and a low-level planner finds a continuous feasible solution based on the discrete path. The low-level planner has a tree structure which expands to improve Pareto optimality of multiple objective functions. When the flying robot follows the trajectory planned by our algorithm, it succeeds the mission robustly as described in simulations.

## References


[1] 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.

[2] 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.

[3] 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.

[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 (ICRA)*, May 2010.

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

[7] 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.

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

[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] J. Suh, J. Gong, and S. Oh, "Fast sampling-based cost-aware path planning with nonmyopic extensions using cross entropy," *IEEE Transactions on Robotics*, vol. 33, no. 6, pp. 1313–1326, Dec 2017.

[13] 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.

[14] 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.

[15] 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.

[16] T. G. McGee and J. K. Hedrick, "Path planning and control for multiple point surveillance by an unmanned aircraft in wind," in *American Control Conference*, 2006.

[17] Y. Oh, K. Cho, Y. Choi, and S. Oh, "Robust multi-layered sampling-based path planning for temporal logic-based missions," in *Proc. of the IEEE Conference on Decision and Control (CDC)*, Dec. 2017.

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

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

[20] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[21] D. Yi, M. A. Goodrich, and K. D. Seppi, "MORRF*: Sampling-based multi-objective motion planning," in *Proc. of the International Joint Conference on Artificial Intelligence*, 2015.