第 36 卷第 11 期
2019年 11 月

控 制 理 论 与 应 用
Control Theory & Applications

Vol. 36 No. 11
Nov. 2019

度量区间时序逻辑下四旋翼的集成任务与运动规划

蓝梦露[1†], 赖叔朋[2], 陈本美[2,3]

(1. 新加坡国立大学 综合科学与工程研究生院, 新加坡 117583;
2. 新加坡国立大学 电子工程系, 新加坡 117573; 3. 香港中文大学 机械及自动化工程系, 中国 香港)

摘要: 本文采用优化方法解决度量区间时序逻辑下进行四旋翼的集成任务与运动规划问题. 传统方法通常将任务约束和运动约束分层处理. 由于不同规划层所使用的抽象模型并不完全匹配, 任务规划层求解出的高层策略往往无法有效地被低层的运动规划层执行, 从而只能找到次优的运动轨迹甚至找不到解. 本文摒弃分层规划的策略, 采用B样条拟合运动轨迹, 将问题转化问一个混合整数线性规划问题, 直接在同一层处理带有时序逻辑的任务约束以及运动约束. 与其他现有方法对比, 本文的方法保证了连续时间下的轨迹也可以完全满足所有约束条件, 而非只有离散的轨迹点才可以满足. 用四旋翼模型进行了一系列仿真验证, 仿真结果表明了方法的有效性.

关键词: 任务与运动规划; 时序逻辑; B样条

引用格式: 蓝梦露, 赖叔朋, 陈本美. 度量区间时序逻辑下四旋翼的集成任务与运动规划. 控制理论与应用, 2019, 36(11): 1952 – 1964

DOI: 10.7641/CTA.2019.90602

Integrated task and motion planning for quadrotors under metric interval temporal logic specifications

LAN Meng-lu[1†], LAI Shu-peng[2], CHEN Ben-mei[2,3]

(1. Graduate School for Integrative Science & Engineering, National University of Singapore, Singapore 117583;
2. Department of Electrical and Computer Engineering, National University of Singapore, Singapore 117583;
3. Department of Mechanical and Automation Engineering, Chinese University of Hong Kong, Hong Kong, China)

Abstract: This paper presents an optimization-based method for the integrated task and motion planning problem under temporal specifications. It is traditional to handle the task specifications and the continuous motion trajectory in separate layers. Due to model mismatching at different layers, it often leads to a sub-optimal or even infeasible trajectory. In this paper, we use the B-spline to formulate a mixed integer programming problem to satisfy the temporal constraints and generates a dynamically feasible trajectory in one run. Compared to existing works, our method guarantees the satisfaction of the state and temporal constraints over the entire trajectory instead of at discrete time points. The proposed method is tested with a simulated quadrotor and validated with various examples.

Key words: task and motion planning; temporal logic; B-spline

Citation: LAN Menglu, LAI Shupeng, CHEN Benmei. Integrated task and motion planning for quadrotors under metric interval temporal logic specifications. *Control Theory & Applications*, 2019, 36(11): 1952 – 1964

1 Introduction

Planning is an essential module in any autonomous system. In the context of robotics, the planning problem is typically decoupled into two layers, i.e, motion planning and task planning. Lower-level motion planning focuses on the continuous domain and mainly handles the dynamics constraints of the vehicle and the geometric constraints presented in the work-space. It aims to find a dynamically feasible and collision-free trajectory from one state to another. On the other hand, higher-level task planning normally works with an abstraction of the original system and focuses on the discrete-domain. It aims to find a sequence of coarse actions (or sometimes a policy) that satisfies task-related requirements, such as “go to rooms P1 or P2 and then go to room P3” and “If the robot find a parcel in room A, it shall then deliver the package to the room B within 5 time units.” These task requirements often require the reasoning over the time domain and thus can be formalized using the temporal logic formulas.

The major problem of two-layered approach is that the discrete model used in task planning layer often totally ignores complex dynamics details and thus generates a sub-optimal or even an infeasible task plan for the

Received 27 July 2019; accepted 17 October 2019.
†Corresponding author. E-mail: lanmenglu@u.nus.edu.sg;
Recommended by Associate Editor: PEI Hai-long.

lower-level motion module to implement[1–2]. An additional interface layer or intermediate procedure[3–7] is often required to communicate between two levels, update the high-level abstraction and perform necessary re-planning procedures.

Instead of using the two-layered approach, one can directly formulate the task requirements as a set of constraints over the underlying detailed dynamics model[8–11]. The integrated task and motion planning problem is then formulated into a single optimization problem. Often, the task requirements are formalized as temporal logics, such as linear temporal logics (LTL) and signal temporal logics (STL). The temporal formulas are then translated into a set of mixed-integer linear constraints. The resulting optimization problem can be handled within a mixed-integer linear or quadratic programming (MILP or MIQP)[12] framework. However, existing works focus only on point-wise constraints instead of interval-wise one. They often either directly assume that a discrete-time model is available, or use sampling-based techniques to abstract the original continuous dynamics into a discrete-time one. These works adopt the *point-wise semantics* of the temporal formulas. In other words, the state trajectory is interpreted as *timed words* (a sequence of timestamp and state pair) instead of a continuous *signal* (a sequence of intervals where the state holds). As a result, constraints are only effective on time points, and there is no guarantee of the satisfaction between two time points.

To address the issue, in this paper, we use smoothing B-spline to parametrize the system trajectory and formalize the task requirements as metric interval temporal logic (MITL). With the B-spline formulation, we can directly work with the continuous dynamics model instead of abstracting it into a discrete-time one. More importantly, the convex hull property of the B-spline allows us to address the interval-wise constraints systematically. Unlike common approaches based on point-wise semantics, we partition the time domain into a sequence of time intervals and translate temporal formulas into mixed-integer linear constraints over time intervals instead of time points. We illustrate our idea with a quadrotor model. Together with the vehicle dynamics and other constraints such as trajectory smoothness, the overall formulation remains an MIQP problem.

In summary, the advantages of our approach includes:

• By satisfying all temporal logical and motion-level constrains in a mixed-integer programming, we avoid the problem of model mismatching due to different levels of abstraction in the task and motion planning layer.

• By utilizing the B-spline to represent the motion trajectory, all constraints can be satisfied on the entire trajectory compared to individually sampled points in existing methods.

The rest of the paper is organized as follows. In Section 2 and Section 3, we present the related-works and the mathematical problem formulation of the integrated task and motion planning under temporal specifications. Section 4 and Section 5 discuss the proposed method, i.e., how to formulate the integrated task and motion planning problem into a MIQP optimization problem using B-spline. Simulations and experiments are provided in Section 6. Finally, Section 7 draw some concluding remarks.

2 Related works

In this section, we present some related works. We first discuss the traditional two-layered planning framework for the integrated task and motion planning problem. Then we proceed to discuss the two major methods for planning under temporal specifications, since the conventional planning techniques only handle the reachability issue (i.e, find a path from one state to another). Specifically, we present automata-based method and the optimization-based method.

2.1 Two-layered planning framework

Task planning and motion planning are normally viewed as two different research domains. The key reason for such an impression is that the former normally deals with abstract discrete dynamics while the latter focuses more on the continuous domain. There are various specialized algorithms and techniques for each domain.

At the task level, one can specify task requirements either in temporal logics or in planning domain definition language (PDDL). One often refers to the planning under temporal logics as a synthesis problem. Since we also use temporal logic as our formal task specification, we will give a more detailed review of the control synthesis under temporal logic in the latter part of this section. Note that it is also possible to express temporal constraints using PDDL[13]. Control synthesis approach and PDDL planning approach are generally different in terms of problem formulation, algorithms complexity, and the expressively of the specification. The two approaches are often studied by two different communities. The synthesis problem is studied more often in the control community and model checking community while the PDDL planning is studied more in the artificial intelligence planning community. Recently, there are more research efforts that aim to close the gap. For instance, [14] handles the non-reactive LTL synthesis problem by compiling the automata generated from LTL into a PDDL action with conditional effects. The resulting problem then can be solved by any standard PDDL classical planner[15]. [16] handles the reactive LTL synthesis problem by compiling the two-player game into a standard fully observable planning problem (FOND). Any PDDL FOND planner thus can be used to solve the problem.

The integration of task and motion planning is not a trivial issue. The high-level task plan may be infeasible for lower-level motion planner to implement. This is due to the fact that, in a common hierarchical layered planning framework, task planning model is essentially an abstraction of the motion-level model; however, it is very difficult to obtain a correct abstraction such that every high-level plan can be implemented by the lower-level planner. In practice, one often adopts the iterative planning strategy. Specifically, one can start with a guessed task-level abstraction, and then plan with the guessed model. If the lower-level motion planner cannot implement the resulting task plan, the planner should figure out the reasons and update the high-level abstraction accordingly. A new planning cycle then begins with the updated high-level task model. For instance, [3] use an independent interface to communicate the geometric details from the motion level to task-level in terms of the logical predicate. [6, 17–18] partition the state space into several regions to form an abstraction. The discrete plan helps the sampling-based motion tree to identify the best possible region to explore. The motion tree also feedbacks utility information to the discrete planner, i.e., updates the discrete abstraction by adjusting region weights. [19] unifies the sampling-based planning for integration task and motion planning. It abstracts the motion planner into a PDDL action. By performing conditional sampling in task-motion space, it constructs a sampling-based abstraction. A discrete planner is then performed on this abstraction. If the planner fails to find a solution, more samples will be drawn to provide a denser abstraction. Though iterative planning provides a solution to the integrated task and motion planning problem, it should be noted that it remains challenging to identify the lower-level planning failure reasons and figure out how to update the high-level abstraction.

2.2 Automata-based methods for planning under temporal specifications

One popular approach to synthesize a controller under temporal specification is based on the automata theory[6, 20–22]. The basic procedure can be summarized as follows. First, the temporal logic is translated into an equivalent automaton in the sense that the language satisfying the temporal formula is the same language accepted by the automaton. For instance, it has been proven that any linear temporal logic (LTL) formula can be fully mapped to an Büchi automaton and there are various tools available such as LTL2BA[23] and Spot[24]. After obtaining the specification automaton, one can create a product automaton between the system abstraction (which is represented as a labelled transition system) and the specification automaton. Finally, one can search for a satisfying path or policy over the product automaton. The automata-based synthesis approach is frequently combined with the iterative planning technique to handle partially unknown environments[6, 22]. Sampling-based techniques can also be integrated with automata-based control synthesis[25–26]. There are also research efforts focusing on partial specification satisfaction based on automata-based synthesis approach[6, 27–28]. Reactive synthesis with state uncertainty is also studied[29]. To synthesize a control strategy under MITL specifications, timed automata can be used. The system abstraction and specification automaton are all expressed in timed automaton[30]. A timed path (a sequence of the time-stamped waypoints) is obtained for a lower-level planner to follow. Algorithms of translating MITL specifications into timed automaton can be found in [31].

2.3 Optimization-based method for planning under temporal specifications

The temporal specification can also be translated directly into a set of constraints over the original dynamical system. The overall formulation is then solved in an MILP/MIQP framework, assuming the predicates are linear. Such an approach is generally referred as control synthesis from an optimization perspective[32]. Pioneer work of such an approach can be found in [8], where the author encodes the LTL formulas into a set of mixed-integer constraints. There is also various work focusing on the encoding of STL[9]. Instead of Boolean satisfaction, STL can provide quantitative satisfaction. For each STL formula, one can assign a robustness s-core over it to evaluate the robustness of the satisfaction. Such a robustness score can be seamlessly integrated into an optimization problem. [9] extends the synthesis problem into a receding horizon control framework. The major problem of MILP approach is the high computational burden, due to a large set of binary variables. Work in[33] adopts an iterative planning approach and introduce fewer binary variables for each iteration. The idea is to guess a trajectory first and identify the violated points. Then they add new constraints over these violation points and solve again, which shares a similar idea of incremental constraints solving[5]. [34] provides a differentiable approximation of robustness function, and thus formulates an SQP problem for higher computational speed. [35] goes one step further by defining a new robustness score (Arithmetic-Geometric Mean Robustness) which directly has gradient information available. As mentioned, existing works adopt point-wise semantics of the temporal formulas and there is no guarantee that the constraints will be satisfied between t-wo discrete time points. The major difference of our proposed method is that we can guarantee interval-wise constraints satisfaction.

3 Problem formulation

In this section, we present the problem formulation for the integrated task and motion planning problem. Since we formalize the task specifications using the metric interval temporal logic (MITL), we will first provide some preliminaries on MITL.

3.1 Signal and MITL

Let T denote the time domain, which is the set of non-negative real number $R \geqslant 0$. A time interval I is a non-empty convex subset of the time domain T. An interval I that sup(I) exists is called a bounded interval. We use $|I|$ to represent sup(I)-inf(I). Let I and J be two intervals over T, the Minkowski sum of two intervals $I \oplus J$ is the set $\{r + s \in T | r \in I, s \in J\}$. Similarly, the Minkowski difference of two intervals $I \ominus J$ is the set $\{r - s \in T | r \in I, s \in J\}$.

Table 1 Minkowski sum of two intervals

$\oplus$	$[c$	$(c$	$\oplus$	$d)$	$d]$
$[a$	$[a+c$	$(a+c$	$b)$	$b+d)$	$b+d)$
$(a$	$(a+c$	$(a+c$	$b]$	$b+d)$	$b+d]$

Let AP denote a set of atomic propositions. A state σ over AP is the subset of AP. A signal γ over 2^{AP} is a function $\gamma : T \to 2^{AP}$. In this work, we assume the signal has only finitely many discontinuities over a bounded time interval, in other words, we exclude the Zeno signal.

We consider a time interval sequence $\bar{I} = I_0 I_1 I_2 \cdots$, which partitions the time domain T (i.e, 1) $\forall i \geqslant 0,\ \sup(I_i) = \inf(I_{i+1})$ and the intersection of I_i and I_{i+1} is the empty set; 2) $\forall t \in T,\ t \in I_i$ for some i). The corresponding *state sequence* over the proposition set 2^{AP} is $\bar{\sigma} = \sigma_0, \sigma_1, \sigma_2, \cdots$. Then the timed state sequence over 2^{AP} is a pair $\kappa = (\bar{\sigma}, \bar{I})$ where $\bar{\sigma}$ and $\bar{I}$ are state sequence and interval sequence over 2^{AP} respectively. We let $\kappa(t) = \sigma_i$ if $t \in I_i$ for some $i \geqslant 0$, and $k(t)$ is essentially a signal over 2^{AP} in the time domain T.

We now introduce metric interval temporal logic (MITL). Formally, MITL formula over a set of propositions AP are defined according to the grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \varphi\, \mathcal{U}_I\, \varphi, \tag{1}$$

where $p \in AP$ and I is a non-singular interval of T with rational end-points. $\neg$ and $\wedge$ are standard Boolean operator negation and conjunction. Other boolean connectives, disjunction ($\vee$), implication ($\Rightarrow$) and equivalence ($\Leftrightarrow$), are defined as standard. $\mathcal{U}_I$ denotes temporal operator *until*. Other temporal operators *eventually* $\Diamond_I$ and *always* $\Box_I$ can be derived:

$$\Diamond_I\varphi := \text{True}\, \mathcal{U}_I\varphi,\ \Box_I\varphi := \neg\Diamond_I\neg\varphi. \tag{2}$$

There are two types of semantics for MITL, one is *point-wise semantics* where the MITL formula is interpreted over timed words; the other is continuous semantics where the formula is interpreted over signals. We focus on the continuous semantics. Given a signal γ over 2^{AP}, $t \in T$, and a formula φ, the satisfaction relation $\gamma, i \models \varphi$ (whether the signal γ satisfies the formula φ at time t) is recursively defined as

- $\gamma, t \models p$ iff $p \in \gamma(t)$;
- $\gamma, t \models \neg\varphi$, iff $\sigma, i \not\models \varphi$;
- $\gamma, t \models \varphi_1 \wedge \varphi_2$, iff $\gamma, i \models \varphi_1$ and $\gamma, t \models \varphi_2$;
- $\gamma, t \models \varphi_1 \mathcal{U}_I \varphi_2$, iff $\exists\, t' \in t \oplus I, \gamma, t' \models \varphi_2$ and $\forall\, t'' \in (t, t'), \gamma, t'' \models \varphi_1$.

Remark Metric temporal logic (MTL) can be seen as a timed version of linear temporal logic (LTL). LTL considers only the ordering of the events but MTL extends LTL with quantitative timed information. However, it has been shown that the model checking of MTL is non decidable over the infinite continuous semantics[36]. To address the issue, metric interval temporal logic (MITL), as a fragment of MTL is proposed. Unlike MTL, MITL does not allow punctuality. In other words, singular time interval for temporal operator is not allowed in MITL. Signal temporal logic (STL)[37] can be seen as a special extension (real-valued signals) of the bounded subset of MITL, i.e, MITL$_{[a,b]}$.

3.2 Integrated task and motion planning

We consider a system with the state space $\mathcal{X} \subseteq \mathbb{R}^n$. The state evolves according to a constrained differential equation:

$$\dot{x}(t) = f(x(t), u(t)),\ x \in X_{\mathrm{C}}, \tag{3}$$

where $X_{\mathrm{C}} \subseteq X \subseteq \mathbb{R}^n$ describe the state constraints. U is a set of admissible control inputs, and $u(t) \in U$ is the control signal.

The semantics of a proposition $p \in AP$ is defined over the state space X by a function HOLDS(p, x) : $AP \times X \to \{\top, \bot\}$, which indicates whether or not $x \in X$ satisfies p. A mapping $\mathcal{M} : V \to 2^{AP}$ defines all the propositions satisfied by the state x:

$$\mathcal{M}(x) = \{p |\ p \in AP \text{ and HOLDS}(p, x) = \top\}. \tag{4}$$

In this paper, we assume that p maps to a union of a finite number of convex polytopes. The integrated task and motion planning problem is to find a control signal $u : [0, T] \to U$ such that the dynamically feasible trajectory $x(t)$ obtained by applying u to initial state i.e, $x(t) = x_{\text{init}} + \int_{h=0}^{t} (f(x(h), u(h)))\mathrm{d}h$ is also collision free and satisfy the formula φ.

4 Motion parametrization

Since a continuous trajectory is of infinite dimension, which cannot be manipulated with numerical methods, it is necessary to describe the trajectory with finite parameters, such as using a fixed ordered polynomial to describe a trajectory. In this paper, we adopt the B-spline formulation to parameterize the trajectory. By using the convex hull property of the B-spline, it is possible to guarantee the satisfaction of temporal logic constraints over a continuous time-segment.

4.1 B-spline formulation

We use a clamped uniform B-spline S_k to parametrize the trajectory, where k is the order of the spline. Let N_i^k denote the i-th basis function, c_i be the corresponding control point and s be the path parameter. The k-th order spline S_k is then defined as

$$S_k(s) = \sum_{i=0}^{M-1} c_i N_i^k(s),\ c_i, S \in \mathcal{X},\ N_i^k \in \mathbb{R}, \tag{5}$$

where $\mathcal{X} \in \mathbb{R}^{d\times 1}$ is the d dimensional *work space* and $d \in \mathbb{N}$.

The matrix $C = [c_0\ c_1\ \cdots\ c_{M-1}] \in \mathbb{R}^{d\times M}$ is referred as the control point matrix and $\hat{C} = C^{\mathrm{T}}$ is the vectorized version of C. We can also create a matrix Λ_i to represent the relationship between the i-th control point c_i and $\hat{C}$, such that:

$$c_i = \Lambda_i \hat{C},\ i \in \{0, 1, \cdots, M-1\}. \tag{6}$$

Let $\mathcal{K} = [s_0\ s_1\ \cdots\ s_{M+k}]$ denote a knot vector, the basis functions are defined recursively over the knot vector as

$$N_i^0(s) = \begin{cases} 1, & \text{if } s_i \leqslant s < s_{i+1}, \\ 0, & \text{otherwise,} \end{cases}$$
$$N_i^j(s) = N_{\mathrm{A}}(s,i,j) N_i^{j-1}(s) + N_{\mathrm{B}}(s,i,j) N_{i+1}^{j-1}(s), \tag{7}$$

where

$$N_{\mathrm{A}}(s,i,j) = \begin{cases} 0, & \text{if } s_i = s_{i+j}, \\ \dfrac{s - s_i}{s_{i+j} - s_i}, & \text{otherwise,} \end{cases}$$
$$N_{\mathrm{B}}(s,i,j) = \begin{cases} 0, & \text{if } s_{i+1} = s_{i+j+1}, \\ \dfrac{s_{i+j+1} - s}{s_{i+j+1} - s_{i+1}}, & \text{otherwise.} \end{cases} \tag{8}$$

For a uniform clamped spline, we have

$$\mathcal{K} = [s_0\ s_1\ \cdots\ s_{M+k}] = [\underbrace{0\ \cdots\ 0}_{k+1 \text{ times}}\ 1\ 2\ \cdots\ \underbrace{M-k\ \cdots\ M-k}_{k+1 \text{ times}}]. \tag{9}$$

There are two important properties of the B-spline. Specifically, they are 1) non-negativity:

$$N_i^k(s) \geqslant 0,\ \forall i \in \{0, 1, \cdots, M-1\},\ \forall s \in [s_0, s_{M+k}], \tag{10}$$

and 2) partition-of-unity:

$$\sum_{i=0}^{M-1} N_i^k(s) = 1,\ \forall s \in [s_0, s_{M+k}]. \tag{11}$$

4.2 Mapping between the path parameter and the time parameter

The mapping between the path parameter s and the time t can be assigned linearly, following the work in [38]:

$$\frac{s}{t} = \alpha. \tag{12}$$

Hence, we have $S_k(s) = S_k(\alpha t)$ where $t \in [0, T_{\mathrm{end}}]$ and $T_{\mathrm{end}} = \dfrac{M-k}{\alpha}$.

From Eq.(12) above, given a time point τ_j, the path variable for τ_j can be obtained as $\alpha\tau_j$. Let $\eta(\tau_j) = \lfloor \alpha\tau_j \rfloor + k$. Then $\alpha\tau_j$ would appear in a knot vector $[s_{\eta(\tau_j)}, s_{\eta(\tau_j)+1})$.

According to the definition of the basis functions (Eq.(8)), for any knot span $[u_i, u_{i+1})$, there are at most $k+1$ non-zero basis functions, which are N_{i-k}^k, $N_{i-k+1}^k, \cdots, N_i^k$. Hence for a time interval $[\tau_j, \rho_j)$, only basis functions $N_i^k(s)$, $i \in \{\eta(\tau_j)-k, \cdots, \eta(\rho_j)\}$ are possibly non-zero. We can then safely ignore all other basis functions since they are zeros and have no effects on the spline.

4.3 Convex hull property of B-spline

Assume we have a linear constraint on state x that can be represented by a convex polytope. Let the polytope be the set of states

$$\{x \in \mathcal{X} | Ax \leqslant b\}. \tag{13}$$

For a B-spline formulation, the above constraint can be satisfied by constraining the control points c_i inside the polytope. Specifically, we have

$$Ac_i \leqslant b,\ i \in \{0, 1, \cdots, M-1\}. \tag{14}$$

We present a simple proof here by applying the two important properties of the B-spline mentioned in Eq.(10) and Eq.(11).

Proof We slightly abuse the notation and use N_i to denote the basis function $N_i^k(t_s)$ in Eq.(5). Let x denote a point on the trajectory at time t_s. We have

$$Ax = A\sum_i N_i c_i = \sum_i N_i (Ac_i). \tag{15}$$

Since basis functions are all non-negative (Eq.(11)) and the summation of all basis functions at a particular point is always unity (Eq.(11)), we then have

$$\sum_i N_i(Ac_i) \leqslant \sum_i N_i b = b \Rightarrow Ax \leqslant b. \tag{16}$$

QED.

We now consider the above linear constraint over a specific time interval $I = [\tau_j, \rho_j)$. Following the Section 4.2, we have

$$Ac_i \leqslant b,\ i \in \{\eta(\tau_j) - k, \cdots, \eta(\rho_j)\}. \tag{17}$$

By replacing c_i with Eq.(6), we have

$$A\Lambda_i\hat{C} \leqslant b,\ i \in \{\eta(\tau_j) - k, \cdots, \eta(\rho_j)\}. \tag{18}$$

4.4 Optimization objective: smoothness

It is desired to generate a smooth trajectory for the vehicle to track. We define the smoothness cost as the following and would like to minimize this cost during the optimization process:

$$E_{\mathrm{smooth}} = \sum_{n=1}^{k} \rho_{(n)} \int_{-\infty}^{\infty} \|\frac{\mathrm{d}^n S_k}{\mathrm{d}t^n}\|^2 \mathrm{d}t = \sum_{n=1}^{k}\sum_{i=1}^{d} \rho_{(n)} \int_{-\infty}^{\infty} (\frac{\mathrm{d}^n S_{k(i)}}{\mathrm{d}t^n})^2 \mathrm{d}t, \tag{19}$$

where $\rho_{(n)} \geqslant 0, \forall n \in \{1, \cdots, k\}$. We can express the above term as a quadratic form, following the work in [39],

$$\hat{C}^{\mathrm{T}}[\sum_{n=1}^{k}(\rho_{(n)}V_n) \otimes I_d]\hat{C}, \tag{20}$$

where

$$V_{n(i,j)} = \alpha^{2n-1}\int_{-\infty}^{\infty}\frac{\mathrm{d}^n N_i^k(s)}{\mathrm{d}s^n}\frac{\mathrm{d}^n N_j^k(s)}{\mathrm{d}s^n}\mathrm{d}s, \tag{21}$$

and $\otimes$ is the Kronecker product operator, and I_d is an identity matrix of dimension $\mathbb{R}^{d\times d}$. We identify that the above cost term is actually obtained by integrating/summing together the square of trajectory's derivatives. For quadrotors, we can take the derivative up to the snap since the snap can be mapped to the angular acceleration of the quadrotor[40]. In practice, minimum jerk trajectory is also popular.

4.5 Dynamics constraints

The trajectory has to satisfy the dynamics constraints of the vehicle. In [41], it has shown that, for a quadrotor, the constraints on the thrust and body rate can be translated into a set of constraints on the accelerations and jerks. In other words, we can satisfy the dynamics constraints of the quadrotor by assigning limits on the derivatives of the trajectory, i.e., velocity, acceleration and jerk. We now show how can we apply the convex hull property of B-spline to constrain these derivatives of the trajectory. Let $S_k^{(1)}$ denote the first derivative of a kth order B-spline S_k, which is a $k-1$ th order B-spline itself:

$$S_k^{(1)} = \frac{\mathrm{d}S^k(s)}{\mathrm{d}t} = \sum_{i=0}^{M-2} c_i^{(1)} N_{i+1}^{k-1}(s), \tag{22}$$

where $c_i^{(1)}$ is given as

$$c_i^{(1)} = \alpha\frac{k(c_{i+1}-c_i)}{s_{i+k+1}-s_{i+1}}. \tag{23}$$

The knot vector $\mathcal{K}^{(1)}$ for $S_k^{(1)}$ is

$$\begin{aligned}\mathcal{K}^{(1)} &= [s_0^{(1)}\ s_1^{(1)}\ \cdots\ s_{M+k-1}^{(1)}] \\ &= [\underbrace{0\ \cdots\ 0}_{k \text{ times}}\ 1\ 2\ \cdots\ \underbrace{M-k\ \cdots\ M-k}_{k \text{ times}}],\end{aligned} \tag{24}$$

which simply removes the first and last elements in the knot vector $\mathcal{K}$ of S_k. Let $C^{(1)} = [c_0^{(1)}\ c_1^{(1)}\ \cdots\ c_{(1)}^{(M-2)}]$ denote the control point matrix of $S_k^{(1)}$, we can express Eq.(23) in a matrix form

$$\hat{C}^{(1)} = \alpha\varGamma_1\hat{C}, \tag{25}$$

where $\hat{C}$ is the vectorized control point matrix of S_k. Similarly, for the nth order derivative $S_k^{(n)}$, the vectorized control point matrix $\hat{C}^{(n)}$ is

$$\hat{C}^{(n)} = \alpha^n\varGamma_n\hat{C}. \tag{26}$$

Following the same reasoning procedure for Eq.(6), we define the mapping matrix for the control points of $S_k^{(n)}$ as

$$c_i^{(n)} = \varLambda_i^{(n)}\hat{C}^{(n)}. \tag{27}$$

Note that $S_k^{(n)}$ is a $k-n$ th order B-spline and we can still apply the convex hull property in Section 4.3. To constrain the n-th derivative $S_k^{(n)}$ inside a convex polytope $\{p \in \mathcal{X} | A_n p \leqslant b_n\}$, it is sufficient to constrain its control point:

$$\begin{aligned}A_n c_i^{(n)} = A_n\varLambda_i^{(n)}\hat{C}^{(n)} = \alpha^n A_n\varLambda_i^{(n)}\varGamma_n\hat{C} < b_n, \\ \forall i \in \{0, \cdots, M-n-1\}.\end{aligned} \tag{28}$$

4.6 Boundary conditions

For a clamped B-spline S_k , the boundary condition can be directly imposed on its control points, because

$$\begin{aligned}&S_k(0) = c_0, \\ &S_k(\alpha T_{\text{end}}) = c_{M-1},\end{aligned} \tag{29}$$

where the c_0 is the first control point and c_{M-1} is the last control point. Similarly, for its nth derivative $S_k^{(n)}$, we have

$$\begin{aligned}&S_k^{(n)}(0) = c_0^{(n)}, \\ &S_k^{(n)}(\alpha T_{\text{end}}) = c_{M-n-1}^{(n)},\end{aligned} \tag{30}$$

where $c_0^{(n)}$ and $c_{M-n-1}^{(n)}$ are first and last control points.

Following the derivative relation (Eq.(26)) and control points mapping (Eq.(6)), we have boundary condition as equality constraints over vectorized control points matrix $\hat{C}$:

$$\begin{cases}\varLambda_0\hat{C} = S_{\text{ini}}, \\ \varLambda_{M-1}\hat{C} = S_{\text{end}}, \\ \alpha^n\varLambda_0^{(n)}\varGamma_n\hat{C} = S_{\text{ini}}^{(n)}, \\ \alpha^n\varLambda_{M-n-1}^{(n)}\varGamma_n\hat{C} = S_{\text{end}}^{(n)},\end{cases} \tag{31}$$

where S_{ini}, S_{end}, $S_{\text{ini}}^{(n)}$, $S_{\text{end}}^{(n)}$ denote the desired initial and end condition of the trajectory S_k and its nth derivative $S_k^{(n)}$.

5 Mixed-integer formulation for MITL

We have discussed how to use B-spline to parametrize the trajectory and formulate basic dynamics constraints in Section 4. In this section, we will discuss how to formally formulate the task requirements specified in MITL. Based on the framework developed by [8] for LTL and [9] for STL, we translate the MITL formula into a set of mixed-integer linear constraints. However, instead of introducing binary variables over time points like[8–9], we define our boolean variables over time intervals. With the capability of B-spline to handle interval-wise constraints (Section 4.3), we can now interpret our trajectory as a continuous signal instead of timed words.

5.1 Encoding

5.1.1 Proposition

A proposition p is essentially mapped to a subset of the underlying state space. In this paper, we assume that such a subset can be described by a union of a finite number of closed convex polytopes. Each polytope then can be expressed as a linear constraint. Assume there are in total N_{poly} polytopes, the j-th polytope is the set of states

$$\{x \in \mathcal{X} | H^{p_j} x \leqslant K^{p_j}\},\ \forall j \in \{0, 1, \cdots, N_{\text{poly}} - 1\}. \tag{32}$$

From the analysis in Section 4.3, the above constraint can be satisfied by constraining the control points c_i instead. Assume that during the time interval $I = [\tau_j, \rho_j)$, the system satisfies the constraint described by the j-th polytope. Following the Section 4.2, we have

$$H^{p_j} c_i \leqslant K^{p_j},\ i \in \{\eta(\tau_j) - k, \cdots, \eta(\rho_j)\}. \tag{33}$$

We define a boolean variable $z_I^{p_j} \in \{0, 1\}$ such that during the time interval $I = [\tau_j, \rho_j)$, if j-th polytope constraint p_j is true, and then $z_I^{p_j} = 1$. For each polytope, we can use standard big-M formulation to enforce the above constraint. By replacing c_i with Eq.(6), we have

$$H^{p_j} \Lambda_i \hat{C} \leqslant K^{p_j} + M_{\text{c}}(1 - z_I^{p_j}),\ i \in \{0, 1, \cdots, M-1\}, \tag{34}$$

where M_{c} is a sufficiently large constant. We further define $z_I^p := \bigvee z_I^{p_j},\ j \in \{0, 1, \cdots, N_{\text{poly}} - 1\}$. If z_I^p is true, then x satisfies the proposition p during the time interval I.

$$\begin{aligned} &z_I^p \geqslant z_I^{p_j},\ \forall j \in \{0, 1, \cdots, N_{\text{poly}} - 1\}, \\ &z_I^p \leqslant \sum_{j=0}^{N_{\text{poly}}-1} z_I^{p_j}. \end{aligned} \tag{35}$$

5.1.2 Boolean operators

For a formula φ, we define a boolean variable z_I^φ to indicate whether the formula φ hold or not during the time interval I. We then define new binary variable z_I^ψ to represent logic operators.

Negation $z_I^\psi = \neg z_I^\varphi$:

$$z_I^\psi = 1 - z_I^\varphi. \tag{36}$$

Conjunction of n sub-formulas $z_I^\psi = \bigwedge_{i=1}^{n} z_I^{\varphi_i}$:

$$\begin{aligned} &z_I^\psi \leqslant z_I^{\varphi_i},\ i = 1, \cdots, n, \\ &z_t^\psi \geqslant 1 - n + \sum_{i=1}^{n} z_I^{\varphi_i}. \end{aligned} \tag{37}$$

Disjunction of n sub-formulas $z_I^\psi = \bigvee_{i=1}^{n} z_I^{\varphi_i}$, we have

$$\begin{aligned} &z_I^\psi \geqslant z_I^{\varphi_i},\ i = 1, \cdots, n, \\ &z_I^\psi \leqslant \sum_{i=1}^{n} z_I^{\varphi_i}. \end{aligned} \tag{38}$$

5.1.3 Temporal operators

Temporal operators can be viewed as the conjunction or disjunction over time intervals. Before we start encoding, we first need to calculate the correct time interval with respect to current time interval. Assume that the temporal operator is operating over interval I with respect to current time interval I_{c}. The new time interval is the Minkowski sum of two intervals $I_{\text{c}} \oplus I$, but within the total time horizon T_{end}. So the final interval is $I_{\text{c}} \oplus I \cap [0, T_{\text{end}}]$.

Always: $\psi = \Box_I \varphi$,

$$z_{I_{\text{c}}}^\psi := \bigwedge_{I_i} z_{I_i}^\varphi,\ \forall I_i \in I_{\text{c}} \oplus I \cap [0, T_{\text{end}}]. \tag{39}$$

Basically, the temporal operator *always* encodes that the during the time interval $I_{\text{c}} \oplus I \cap [0, T_N]$, the formula has to be satisfied for every sub time intervals. The encoding of conjunction is similar to Eq.(37), but with one φ for n intervals. Note that if the formula φ can be represented as a set of convex constraints (e.g., an simple convex proposition), it is possible to merge the multiple sub time interval into a big one.

Eventually: $\psi = \Diamond_I \varphi$,

$$z_{I_{\text{c}}}^\psi := \bigvee_{I_i} z_{I_i}^\varphi, \forall I_i \in I_{\text{c}} \oplus I \cap [0, T_{\text{end}}]. \tag{40}$$

The temporal operator *always* basically encodes that the during time interval $[a', b']$, the formula has to be satisfied at one of sub intervals. The encoding of disjunction is similar to Eq.(38).

Until: $\psi = \varphi_1 \mathcal{U}_I \varphi_2$,

$$z_{I_{\text{c}}}^\psi = \bigvee_{I' \in I_{\text{c}} \oplus I \cap [0, T_{\text{end}}]} (z_{I'}^{\varphi_2} \wedge \bigwedge_{I'' \in [I_{\text{c}}, I']} z_{I''}^{\varphi_1}). \tag{41}$$

5.2 Prefix-suffix structure

The MITL is interpreted over an infinite sequence. However, we have to encode this infinite behaviour in a finite number of time intervals. The core idea is to parametrize the infinite trajectory into a prefix-suffix form. The prefix is a finite path starting from the initial state and the suffix part is a periodic signal that forms a loop. For an MITL specification, at least one of the satisfying run is in the prefix-suffix form. The result is directly derived from the LTL. The LTL formula can be translated equivalently into a Büchi automata. The loop structure comes from the fact that acceptance conditions of the Büchi automata require a run visits the same accepting state infinite often. Hence, the satisfying run can be decomposed into two segments. One segment (prefix) starts from the initial state, and ends in one of accepting state. Another segment (suffix) starts from the reached accepting state and comes back to the same accepting state (possibly after visiting some other states), and thus looping around the accepting state and satisfying the Büchi acceptance condition.

Formally, we consider the trajectory in the form of $x = x_{\text{pre}}(x_{\text{suf}})^\omega$, where x_{pre} is the prefix and x_{suf} is the suffix. Let $x_{\text{cat}} := x_{\text{pre}} x_{\text{suf}}$ denote the concatenation of prefix and suffix. Assume there are in total $N + 1$ time intervals. We assign interval indices to x_{cat}

as $T_{\text{cat}} := \{0,1,\cdots,L,\cdots,N\}$, let $T_{\text{pre}} := \{0,1,\cdots,L-1\}$ and $T_{\text{suf}} := \{L,\cdots,N\}$. To encode the prefix-suffix structure, for each time interval i, we introduce a boolean variable l_i to indicate whether the loop starts in this particular time interval. If a loop occurs, then it means the start of the suffix part of the signal should be the same as the end of the suffix part of signal. Let t_L be the $\inf(I_L)$ of the time interval I_L, and t_{end} be the total end time used for the optimization. We can enforce the prefix-suffix structure by imposing a constraint of $x(t_L) = x(t_{\text{end}})$.

$$\begin{cases} \sum\limits_{i=0}^{N} l_i = 1, \\ x(t_{\text{end}}) \leqslant x(t_L) + M_i(1-l_i),\ i=0,\cdots,N, \\ x(t_{\text{end}}) \geqslant x(t_L) + M_i(1-l_i),\ i=0,\cdots,N. \end{cases} \tag{42}$$

6 Simulations and results

This section presents the simulation results for the proposed method as well as the comparison study with other methods. By using the B-spline to parametrize the motion trajectory, our method can directly handle the continuous-time dynamics, without the need to discretize it into a discrete-time one. Also, the B-spline formulation allows us to handle the interval wise constraints directly. As a result, unlike the previous methods relying on pointwise semantics, our generated trajectory is not a sequence of points, but a true smooth signal over continuous time.

We first present two examples to illustrate the ability of our method. Then we present the comparison study between our method and other existing methods, including the traditional two-layered method and the method based on point-wise semantics of temporal logic. For all simulations, we use a quadrotor model presented above. The total planning horizon is set as 10 seconds and we partition it into 10 time intervals. The implementation is done in the Matlab. The encoding of MITL is obtained by modifying the existing toolbox BluSTL[9]. The B-spline formulation is based on our previous works [42] and [43]. We use the Gurobi as the main optimization solver for our simulations.

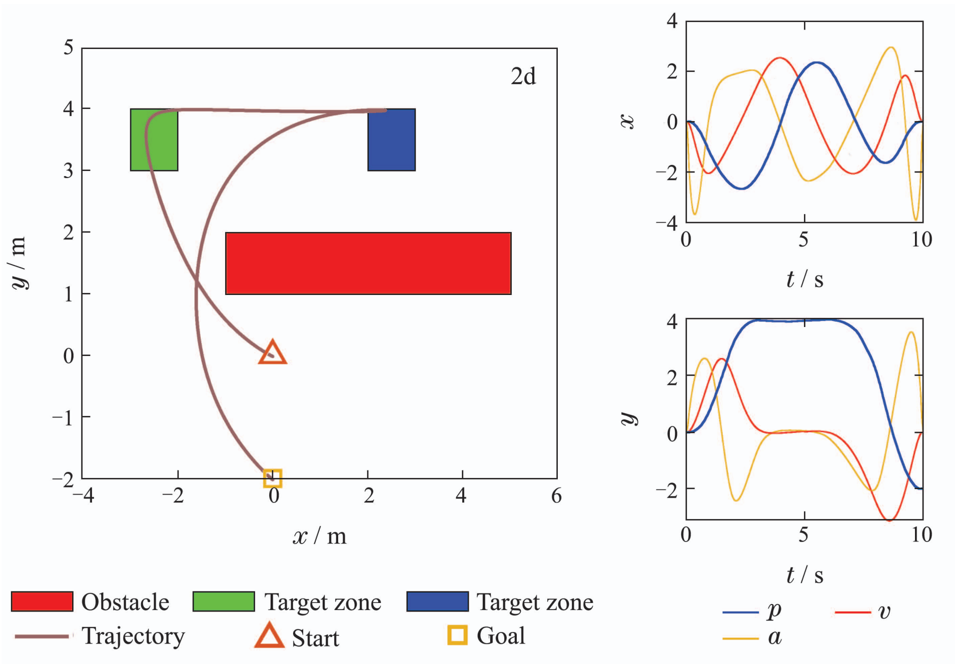
6.1 Simulation results

Example 1 First of all, we consider a simple temporal formula, where the time interval for the temporal operators is $[0,\infty)$. Specifically, the drone is always required to avoid the red obstacle. Also, it has to eventually reach the blue region and green region for some tasks. Finally, it should stop at an end target. Such a task specification can describe a wide range of tasks. For instance, it can model the task that the drone need to take some pictures of both blue region and green region. The order of visiting the green region and the blue region does not matter as long as two regions have been both visited. Formally, if we omit the time interval notation, the specification of Example 1 can be written as

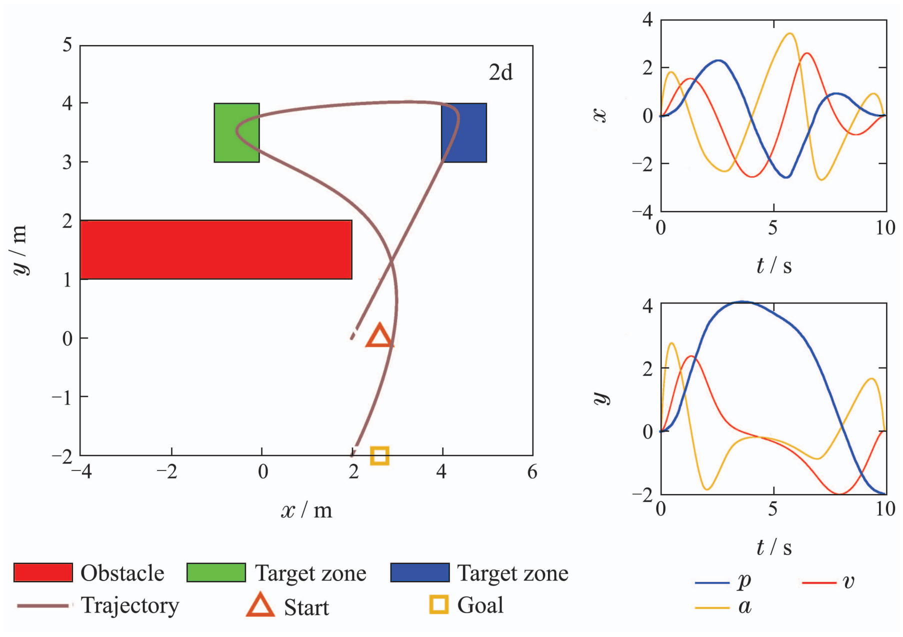
$$\varphi_{\text{eg1}} := \Box(\neg p_{\text{red}}) \wedge \Diamond p_{\text{green}} \wedge \Diamond p_{\text{blue}} \wedge \Diamond\Box p_{\text{goal}}, \tag{43}$$

where p_{red}, p_{green} and p_{blue} are propositions to indicate whether the vehicle position is inside the red, green or blue region. Where red, green, blue indicate the target zones.

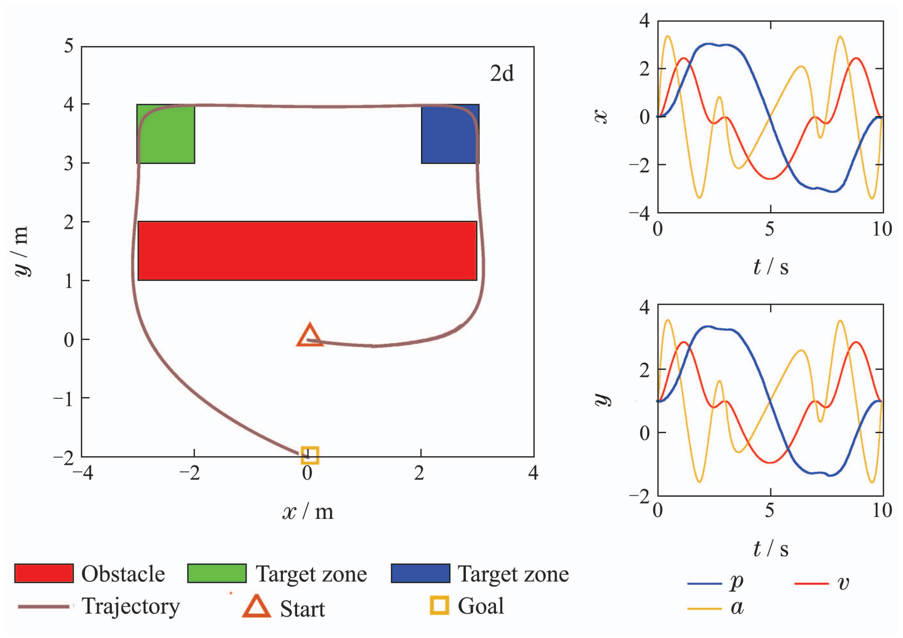
We create three different settings by varying the locations of red, blue, and green regions, as well as the initial target and goal target, as shown in Fig. 1.



(a) Case 1



(b) Case 2



(c) Case 3

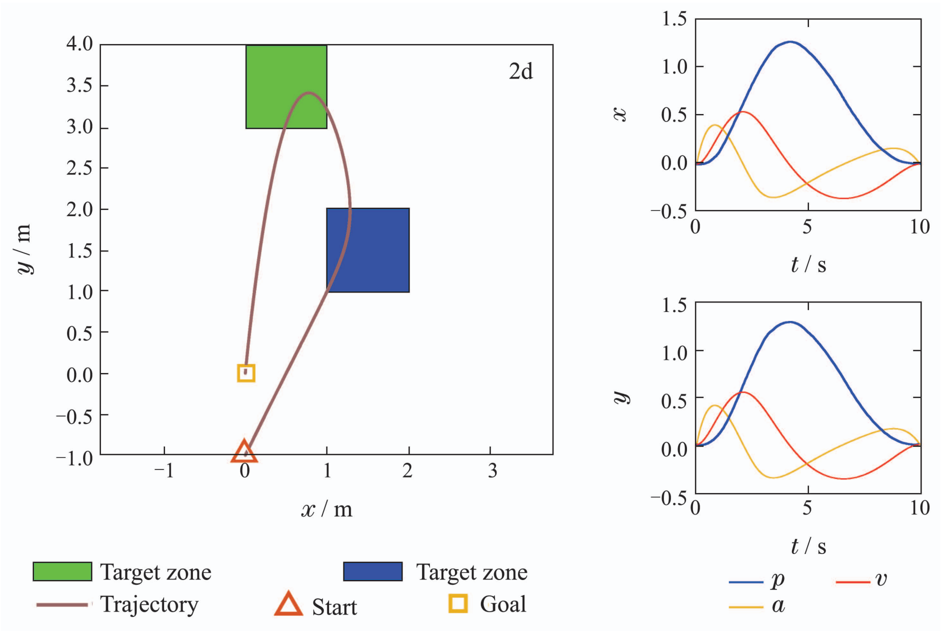
Fig. 1 Example 1. The quadrotor is required to always avoid the red region and eventually visit the green region and the blue region. Also, it is required that the vehicle finally stops at the goal location. We consider three different settings by varying the locations of interested regions. The left sub-figures show the workspace. The sub-figures on the middle and right columns show the position, velocity and acceleration profiles for x axis and y axis respectively

For different settings, the generated trajectories are also different. For instance, in Case 1 (Fig. 1), the drone reaches green region first and then reaches the blue region. In Case 2 (Fig. 1), the drone reaches the blue region first, then it reaches the green region. However, all generated trajectories satisfy the task requirements, i.e., avoiding the red regions, reaching both green and blue regions, stopping at goal location. The position, velocity, and acceleration profiles for x axis and y axis are shown in the sub-figures of each case. As shown, we have continuous smooth signals for all position, velocity, and acceleration. More importantly, unlike the point-wise semantics, the constraints are satisfied over the entire trajectory instead of individual points.

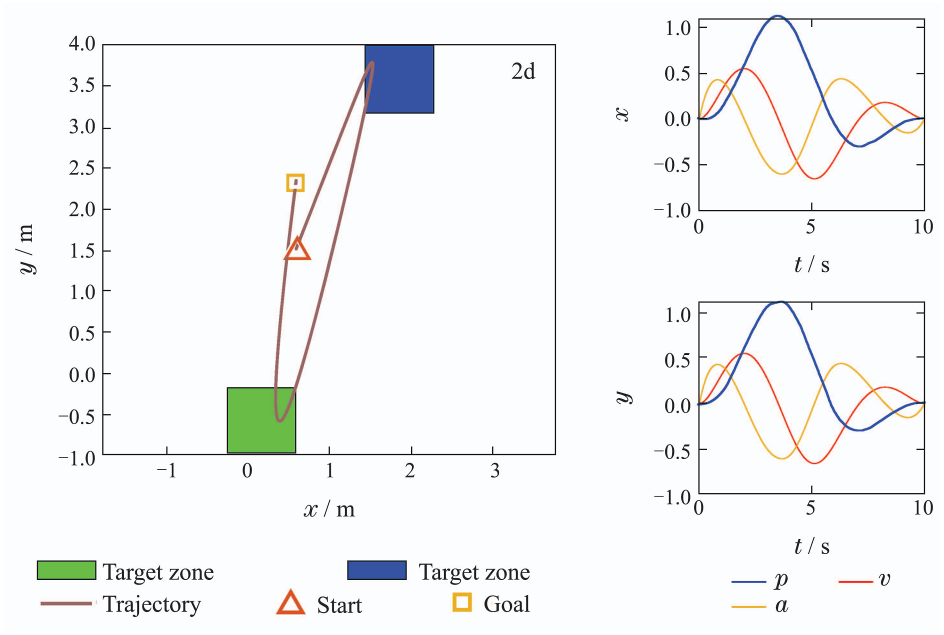
Example 2 In the second example, the quadrotor is required first to reach the blue region within 5 s, and after that, it should reach the green region within the next 4 s. Similar to Example 1, the drone has to finally stop at the goal target. The specification can be formally written as

$$\varphi_{\mathrm{eg2}} := \Diamond_{[0,5]}(p_{\mathrm{blue}} \wedge \Diamond_{[0,4]}p_{\mathrm{green}}) \wedge \Diamond\Box p_{\mathrm{goal}}. \tag{44}$$

We also vary the locations of interested regions and present two different settings for the second example, as shown in Fig. 2. As expected, the generated trajectories for different workspace settings are also different, but all satisfy the requirements.



(a) Case 1



(b) Case 2

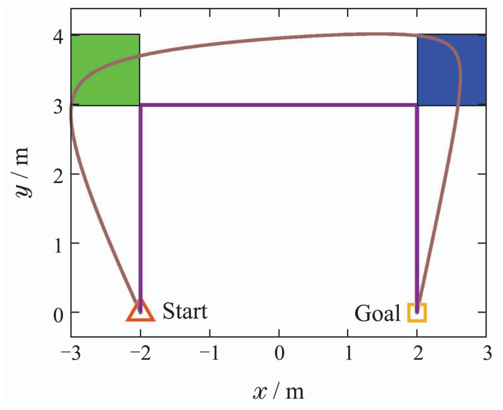
Fig. 2 Example 2. The quadrotor is required first to reach the blue region within 5 s. After that, it should reach the green region within the next 4 s. Also, it is required that the vehicle finally stop at the goal location. We consider two different settings by varying the locations of interested regions. The left sub-figures show the workspace. The sub-figures on the middle and right columns show the position, velocity and acceleration profiles for x axis and y axis respectively

First of all, we compare our method with the traditional two layered approach. For a mobile robot such as the quadrotor, it is traditional to decouple the task and motion planning into two layers. The complex system is first abstracted into a simpler one, which often totally ignores the high order dynamics such as velocity, acceleration and jerks. Then a high-level planner such as a pure geometric planner or discrete task planner is involved to generate a high-level plan based on the abstracted model. The resulted high-level plan is then fed into the lower-level planner to implement, where the detailed dynamics are considered. The major drawback of the two-layered approach is that it requires a simulation (or bi-simulation) relation between the high level planning model and low level planning model. In other words, it works with an underlying assumption that every high-level task plan can be implemented or simulated by the lower layer planer. Clearly, such an assumption does not always hold in real applications. Even the high-level plan is implementable, the resulted trajectory may be highly sub-optimal. It remains an open problem to obtain a correct abstract model for general dynamical system such that the abstraction has a simulation relation with the original system.

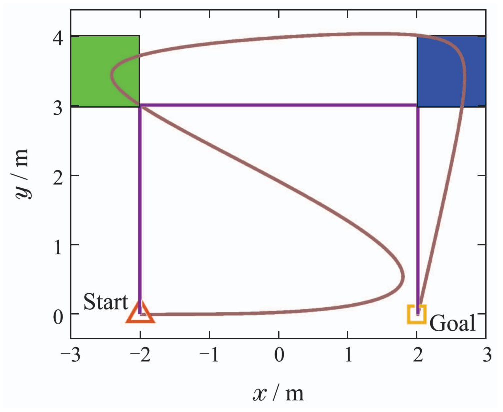
Figure 3 shows the comparison result with the two-layered approach. The vehicle is tasked to visit two regions (green and blue regions) in any order from a start position and stop at a goal position. The high-level planner considers only position-level dynamics of the system and computes a pure geometric path that fulfils the given temporal logic. For this particular task, the high-level plan is calculated as visiting green region first and then visiting blue region (see the purple straight line in Fig. 3(a). Note that this geometric path is actually optimal with respect to the abstracted model (i.e, it is the shortest geometric path that fulfils the task-level specification). We use the same B-spline based motion algorithm discussed in Section 4 as our lower-level trajectory generation algorithm. The optimization target is the same (i.e, to generate minimal snap trajectory).

As shown in Fig. 3(a), if the initial velocity of the vehicle is zero, the resulted lower-level smooth trajectory is sufficiently satisfying. However, if the vehicle has a non-trivial amount of initial velocity, it is no longer suitable to stick to the high-level plan.

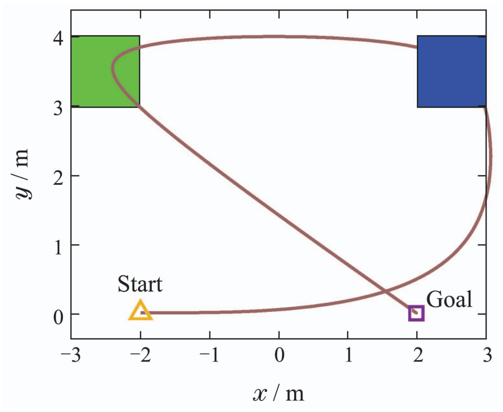
As shown in Fig. 3(b), due to the non-zero initial velocity of the positive x-direction, the generated trajectory curves intensively in order to visit the green region first, which is highly sub-optimal with respect to the full system dynamics. On the other hand, our method is able to find the optimal solution, which visits the blue region first and then visits the green region, as shown in Fig. 3(c). Compared to our method, the two layered planner shown in Fig. 3(b) has a cost value that is 26% higher.



(a) Two-layered method: zero initial velocity



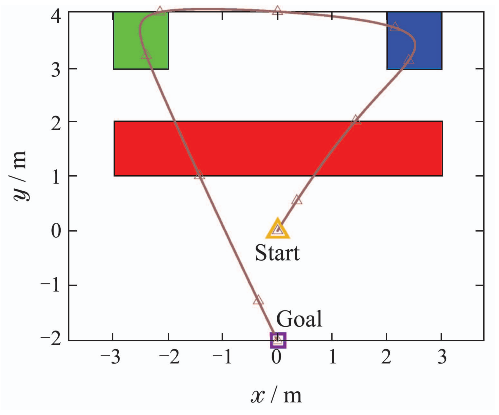
(b) Two-layered method: non-zero initial velocity



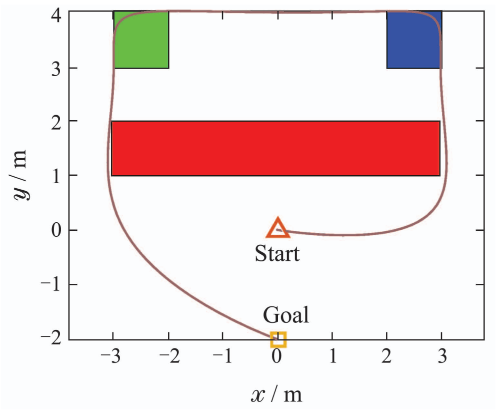
(c) The proposed method: non-zero initial velocity

Fig. 3 Comparison with two-layered planning method. The vehicle is tasked to visit two regions (green and blue regions) in any order from a start position and stops at a goal position

We also compare our method to the existing methods of adopting the point-wise semantics of the temporal logics. The point-wise semantics interpreted the state trajectory as timed words. As a result, temporal constraints are only enforced on the individual points of the trajectory. There is a risk that the constraints will be violated when the sampling rate is not high enough. In Fig. 4(a), The vehicle is tasked to reach green and blue regions in any order while avoiding the red obstacle area. If we only enforce the obstacle avoidance constraints on the sampled points (the triangles) of the trajectory, it is possible that while all the sampled points are collision free, but the trajectory is still not-safe.



(a) The method using point-wise semantics



(b) The proposed method

Fig. 4 Comparison with methods adopting the point-wise semantics. The vehicle is tasked to reach green and blue regions in any order while avoiding the red obstacle area.

7 Conclusion

This paper presented an MIQP method for the integrated task and motion planning problem for a quadrotor. The task requirements were formalized by metric interval temporal logic and were translated into a set of mixed-integer constraints. The most crucial component of our method is the B-Spline based motion parameterization, which allows us to handle continuous-time dynamics and interval wise constraints directly. We conducted various simulations to verify the effectiveness of our system. The simulation results showed that our method could successfully generate a smooth trajectory over continuous domain while satisfying the temporal specification. Our work can be naturally extended into a receding horizon control framework or be extended to handle the robustness satisfaction instead of Boolean satisfaction. Another future direction is to study how to generate a domain-dependent time-domain partition.

References:

[1] SRIVASTAVA S, RUSSELL S, PINTO A. Metaphysics of planning domain descriptions. *Proceedings of the 30th AAAI Conference on Artificial Intelligence*. Phoenix, USA: AAAI Press, 2016: 1074 – 1080.

[2] HAUSER K, LATOMBE J C. Integrating task and PRM motion planning: Dealing with many infeasible motion planning queries. *ICAPS 2009 Workshop on Bridging the Gap between Task and Motion Planning*. Thessaloniki, Greece: AAAI Press, 2009.

[3] SRIVASTAVA S, FANG E, RIANO L, et al. Combined task and motion planning through an extensible planner-independent interface layer. *Proceedings of the 2014 IEEE International Conference on Robotics and Automation*. Hong Kong, China: IEEE, 2014: 639 – 646.

[4] PLAKU E, LE D. Interactive search for action and motion planning with dynamics. *Journal of Experimental & Theoretical Artificial Intelligence*, 2016, 28(5): 849 – 869.

[5] DANTAM N T, KINGSTON Z K, CHAUDHURI S, et al. An incremental constraint-based framework for task and motion planning. *The International Journal of Robotics Research*, 2018, 37(10): 1134 – 1151.

[6] LAHIJANIAN M, MALY M R, FRIED D, et al. Iterative temporal planning in uncertain environments with partial satisfaction guarantees. *IEEE Transactions on Robotics*, 2016, 32(3): 583 – 599.

[7] DE SILVA L, PANDEY A K, ALAMI R. An interface for interleaved symbolic-geometric planning and backtracking. *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan: IEEE, 2013: 232 – 239.

[8] KARAMAN S, SANFELICE R G, FRAZZOLI E. Optimal control of mixed logical dynamical systems with linear temporal logic specifications. *Proceedings of the 47th IEEE Conference on Decision and Control*. Cancun, Mexico : IEEE, 2008: 2117 – 2122.

[9] RAMAN V, DONZÉ A, MAASOUMY M, et al. Model predictive control with signal temporal logic specifications. *Proceedings of the 53rd IEEE Conference on Decision and Control*. Los Angeles, USA: IEEE, 2014: 81 – 87.

[10] SADRADDINI S, BELTA C. Formal synthesis of control strategies for positive monotone systems. *IEEE Transactions on Automatic Control*, 2019, 64(2): 480 – 495.

[11] WOLFF E M, MURRAY R M. Optimal control of nonlinear systems with temporal logic specifications. INABA M, CORKE P. *Robotics Research: Springer Tracts in Advanced Robotics*. Cham: Springer, 2016, 114: 21 – 37.

[12] VIELMA J P. Mixed integer linear programming formulation techniques. *SIAM Review*, 2015, 57(1): 3 – 57.

[13] FOX M, LONG D. PDDL2.1: An extension to PDDL for expressing temporal planning domains. *Journal of Artificial Intelligence Research*, 2003, 20: 61 – 124.

[14] PATRIZI F, LIPOVEZTKY N, DE GIACOMO G, et al. Computing infinite plans for LTL goals using a classical planner. *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*. Barcelona: AAAI Press, 2011.

[15] HOFFMANN J. FF: The fast-forward planning system. *AI Magazine*, 2001, 22(3): 57 – 62.

[16] CAMACHO A, BAIER J A, MUISE C, et al. Finite LTL synthesis as planning. *Proceedings of the 28th International Conference on Automated Planning and Scheduling*. Delft, Netherlands: AAAI Press, 2018.

[17] PLAKU E, KAVRAKI L E, VARDI M Y. Discrete search leading continuous exploration for kinodynamic motion planning. *Proceedings of Robotics: Science and Systems*. Atlanta, USA: MIT Press, 2007: 326 – 333.

[18] PLAKU E, HAGER G D. Sampling-based motion and symbolic action planning with geometric and differential constraints. *Proceedings of the 2010 IEEE International Conference on Robotics and Automation*. Anchorage, Alaska: IEEE, 2010: 5002 – 5008.

[19] GARRETT C R, LOZANO-PÉREZ T, KAELBLING L P. Sampling-based methods for factored task and motion planning. *The International Journal of Robotics Research*, 2018, 37(13/14): 1796 – 1825.

[20] SMITH S L, TUMOVáJ, BELTA C, et al. Optimal path planning under temporal logic constraints. *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Taipei, China: IEEE, 2010: 3288 – 3293.

[21] SMITH S L, TUMOVáJ, BELTA C, et al. Optimal path planning for surveillance with temporal-logic constraints. *The International Journal of Robotics Research*, 2011, 30(14): 1695 – 1708.

[22] MALY M R, LAHIJANIAN M, KAVRAKI L E, et al. Iterative temporal motion planning for hybrid systems in partially unknown environments. *Proceedings of the 16th International Conference on Hybrid Systems: Computation and Control*. Philadelphia, Pennsylvania, USA: ACM, 2013: 353 – 362.

[23] GASTIN P, ODDOUX D. Fast LTL to Büchi automata translation. BERRY G, COMON H, FINKEL A. *Computer Aided Verification (CAV 2001): Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 2001, 2102: 53 – 65.

[24] DURET-LUTZ A, LEWKOWICZ A, FAUCHILLE A, et al. Spot2.0-–a framework for LTL and ω-automata manipulation //ARTHO C, LEGAY A, PELED D. *Automated Technology for Verification and Analysis (ATVA 2016): Lecture Notes in Computer Science*. Cham: Springer, 2016, 9938: 122 – 129.

[25] KANTAROS Y, ZAVLANOS M M. Sampling-based optimal control synthesis for multirobot systems under global temporal tasks. *IEEE Transactions on Automatic Control*, 2018, 64(5): 1916 – 1931.

[26] VASILE C I, BELTA C. Sampling-based temporal logic path planning. *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*. Tokyo, Japan: IEEE, 2013: 4817 – 4822.

[27] LAHIJANIAN M, ALMAGOR S, FRIED D, et al. This time the robot settles for a cost: A quantitative approach to temporal logic planning with partial satisfaction. *Proceedings of the 29th AAAI Conference on Artificial Intelligence*. Austin, USA: AAAI, 2015.

[28] KIM K, FAINEKOS G, SANKARANARAYANAN S. On the minimal revision problem of specification automata. *The International Journal of Robotics Research*, 2016, 34(12): 1515 – 1535.

[29] MONTANA F J, LIU J, DODD T J. Sampling-based reactive motion planning with temporal logic constraints and imperfect state information. PETRUCCI L, SECELEANU C, CAVALCANTI A. *Critical Systems: Formal Methods and Automated Verification (AVoCS 2017, FMICS 2017): Lecture Notes in Computer Science*. Cham: Springer, 2017, 10471: 134 – 149.

[30] ZHOU Y, MAITY D, BARAS J S. Timed automata approach for motion planning using metric interval temporal logic. *Proceedings of the 2016 European Control Conference (ECC)*. Aalborg, Denmark: IEEE, 2016: 690 – 695.

[31] BRIHAYE T, GEERAERTS G, HO HM, et al. MightyL: A compositional translation from mitl to timed automata. MAJUMDAR R, KUNČAK V. *International Conference on Computer Aided Verification: Lecture Notes in Computer Science*. Cham: Springer, 2017, 10426: 421 – 440.

[32] BELTA C, SADRADDINI S. Formal methods for control synthesis: An optimization perspective. *Annual Review of Control, Robotics and Autonomous Systems*, 2019, 2(1): 115 – 140.

[33] SAHA S, JULIUS A A. An MILP approach for real-time optimal controller synthesis with metric temporal logic specifications. *Proceedings of the 2016 American Control Conference (ACC)*. Boston, USA: IEEE, 2016: 1105 – 1110.

[34] PANT Y V, ABBAS H, MANGHARAM R. Smooth operator: Control using the smooth robustness of temporal logic. *Proceedings of the 2017 IEEE Conference on Control Technology and Applications (CCTA)*. Mauna Lani, USA: IEEE, 2017: 1235 – 1240.

[35] MEHDIPOUR N, VASILE C I, BELTA C. Arithmetic-geometric mean robustness for control from signal temporal logic specifications. *Proceedings of the 2019 American Control Conference (ACC)*. Philadelphia, PA, USA: IEEE, 2019: 1690 – 1695.

[36] ALUR R, FEDER T, HENZINGER T A. The benefits of relaxing punctuality. *Journal of the Association for Computing Machinery*, 1996, 43(1): 116 – 146.

[37] MALER O, NICKOVIC D. Monitoring temporal properties of continuous signals. LAKHNECH Y, YOVINE S. *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems (FTRTFT 2004, FORMATS 2004): Lecture Notes in Computer Science*. Berlin, Heidelberg: Springer, 2004, 3253: 152 – 166.

[38] KANO H, FUJIOKA H, MARTIN C F. Optimal smoothing and interpolating splines with constraints. *Applied Mathematics and Computation*, 2011, 218(5): 1831 – 1844.

[39] KANO H, FUJIOKA H. Velocity and acceleration constrained trajectory planning by smoothing splines. *Proceedings of the IEEE 26th International Symposium on Industrial Electronics (ISIE)*. Edinburgh, UK: IEEE, 2017: 1167 – 1172.

[40] MELLINGER D, KUMAR V. Minimum snap trajectory generation and control for quadrotors. *Proceedings of the 2011 IEEE International Conference on Robotics and Automation*. Shanghai, China: IEEE, 2011: 2520 – 2525.

[41] HEHN M, D' ANDREA R. Real-time trajectory generation for quadrocopters. *IEEE Transactions on Robotics*, 2015, 31(4): 877 – 892.

[42] LAI S, LAN M, CHEN B M. Optimal constrained trajectory generation for quadrotors through smoothing splines. *Proceedings of the 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Madrid, Spain: IEEE, 2018: 4743 – 4750.

[43] LAI S, LAN M, GONG K, et al. Axis coupled trajectory generation for chains of integrators through smoothing splines. *Control Theory and Technology*, 2019, 17(1): 48 – 61.

作者简介:

蓝梦露 博士研究生, 目前主要研究方向为无人机运动及任务规划, E-mail: lanmenglu@u.nus.edu.sg;

赖叔朋 博士, 目前主要研究方向为无人系统、运动规划等, E-mail: elelais@nus.edu.sg;

陈本美 教授, 目前研究方向为无人系统及控制应用等, E-mail: bmchen@ieee.org.