論文 / 著書情報
Article / Book Information

題目(和文)	時相論理を用いたタスク表現に対するロボットシステムの軌道最適化と制御
Title(English)	Trajectory Optimization and Control of Robot Systems with Temporal Logic Tasks
著者(和文)	徳田俊平
Author(English)	Shumpei Tokuda
出典(和文)	学位:博士(工学), 学位授与機関:東京工業大学, 報告番号:甲第12692号, 授与年月日:2024年3月26日, 学位の種別:課程博士, 審査員:山北 昌毅,三平 満司,倉林 大輔,田中 正行,畑中 健志
Citation(English)	Degree:Doctor (Engineering), Conferring organization: Tokyo Institute of Technology, Report number:甲第12692号, Conferred date:2024/3/26, Degree Type:Course doctor, Examiner:,,,,
学位種別(和文)	博士論文
Type(English)	Doctoral Thesis

Tokyo Institute of Technology

令和 5 年度 博士論文

時相論理を用いたタスク表現に対する ロボットシステムの軌道最適化と制御

Trajectory Optimization and Control of Robot Systems with Temporal Logic Tasks

2024年3月

東京工業大学
工学院　システム制御系
徳田 俊平
指導教員 山北 昌毅 准教授

Abstract

In recent years, robots play increasingly important roles in society, not only in the industrial field but also in the service industry. With the spread of robots, opportunities for non-professionals are increasing to operate robots and introduce them to services. However, teaching robots to perform desired tasks safely is difficult, and this is one of the obstacles to the introduction of robots. One of the reasons for the difficulty in teaching tasks to robots is the need to simultaneously consider the logical consistency of the task and its feasibility considering the dynamic characteristics of the robot. Therefore, we need to use a control architecture that considers simultaneously both.

The use of modal logic such as linear temporal logic (LTL) and signal temporal logic (STL) has attracted much attention as a way of logically representing tasks in robot systems. These modal logics represent the properties of sequences of state transitions. By describing the rules and constraints of a task as logical expressions, we can determine in advance whether the assumed task is logically sound or not. When generating a sequence of state transitions satisfying the temporal logic on a robot system, we need to generate robot trajectories on an abstracted system, rather than on the actual robot system.

In this case, depending on the state and the definition of the approximated system, the extent to which the abstracted system can emulate the characteristics of the actual robot system differs. In addition, the method of generating robot trajectories differs depending on how the abstracted system is constructed. Therefore, the nonlinearity of the abstracted system and how it is constructed are important depending on the actual robot systems or tasks. Especially, when the trajectory optimization is formulated as a mixed integer programming problem, it is necessary to construct a dynamical linear system that emulates the dynamical characteristics of the actual robot as an approximated system.

To achieve tasks, we need to control the actual system based on trajectories generated by the abstracted system. Control methods for satisfying tasks expressed as temporal logic specifications include model predictive control (MPC) and control barrier functions (CBFs). CBFs are used as constraints in determining control inputs to keep the system trajectory within a certain region, and the constrained optimization problem with CBFs can be formulated as a quadratic programming problem, which requires less computation time than the nonlinear MPC. In designing a controller to achieve tasks represented by temporal logic specifications using CBFs, the issue is how to set up the CBFs. In the case of a task with a time limit expressed in STL, the design method of CBFs proposed in the previous studies requires several processes to design the CBF from the robustness functions of STL. Therefore, we propose a new design method for CBFs. Furthermore, there are cases in which robots

cannot achieve a given task with the task plans. In this case, if there are multiple robots, the robots need to change their motions to achieve the task considering the other robots.

In this paper, we propose trajectory optimization methods and control methods for robot systems for tasks expressed as temporal logic specifications. We focus on the task represented by syntactically co-safe LTL, generate trajectories in response to their tasks, and control robots. In this paper, we assume that the control architecture is divided into multiple levels.

The high-level controller defines an abstracted robot system for the actual robot system. We generate trajectories and motions for this abstracted system to achieve tasks expressed by LTL or STL specifications by optimization. Then, the extent to which the abstracted system can emulate the actual robot depends on the abstraction method of the abstracted system. Therefore, we discuss abstraction methods for robot systems. In addition, we show that we can compute trajectories that satisfy the given temporal logic specifications for the abstracted system. Furthermore, we propose a trajectory optimization method for tasks involving external events represented by event-based STL specifications.

The low-level controller uses the task plans and trajectories generated by the high-level controller to control the actual robot systems. In this paper, we propose a control method that considers time constraints to satisfy a given temporal logic specifications using CBFs. In addition, we also consider methods to represent trajectories as distributions, and to update the distributions based on the observation information obtained during the operation to change the motions.

目 次

図 目 次

表 目 次

第1章　序論

1.1　本論文の背景

近年, ロボットの社会における役割は大きくなっており, 工業的な現場のみならずサービス業などの現場で活躍している [1]. ロボットの波及に伴い, 非専門家がロボットを操作し現場に導入する機会が増加している. しかし, ロボットに安全性を考慮して望みのタスクを実行するように教示することは難しく, そのことがロボットの導入に対する障害の一つとなっている. ロボットにタスクを教示する際の難しさの理由の一つとして, タスクの論理的な整合性とロボットの動的特性を考慮した実現性を同時に考慮する必要があることが挙げられる. そのため, それらを同時に考慮したロボットの制御アーキテクチャを用いることが望ましいと考えられる.

1.1.1　時相論理を用いたタスク表現

ロボットシステムのタスクを論理的に表現する方法として, Linear Temporal Logic (線形時相論理, LTL) や Signal Temporal Logic (信号時相論理, STL) といった様相論理を用いることが注目されている [2,3]. これらの様相論理は状態遷移列の性質を表現する論理である [4]. タスクに関するルールや制約条件を論理式として記述することで, 想定しているタスクに論理的な破綻がないかどうかを事前に確認することが可能である. STL は LTL を拡張した時相論理である [5]. STL を状態遷移モデルに対して定義する際, STL の各命題に対して状態を引数とした関数であるロバストネス関数を導入する. このとき, 各 STL 命題の真偽はロバストネス関数の値によって評価することができる. 加えて, STL はタスクに対する制限時間といった定量的な時間特性を表現することも可能である. 加えて, ロボットの状態に関係なく発生するような外的事象に依存したタスクを表現するための時相論理である Event-based STL が提案されている [6]. Event-base STL を用いることで, ロボットが人間が発信する信号に反応して対応するようなタスクを記述することも可能である.

1.1.2　時相論理を用いた行動計画手法

LTL や STL を用いて論理的な整合性を考慮して定義されたタスクに対してロボットの行動計画を行う手法として様々な手法が提案されている [7–11]. ロボットシステム上で時相論理を満たす状態遷移列を生成する際, 実際に制御を行うロボットシステムではなく, 抽象

化したシステム上でロボットの軌道を生成する. 抽象化したシステムとは, 実際に制御を行うロボットシステムの特性を反映したシステムである. 抽象化システムの構築方法として, Simulation, Bisimulation や マルコフ決定過程としてシステムを表現する手法が提案されている [7, 8, 12]. これらの手法の特徴として, 実際のロボットシステムと抽象化したシステムの状態間での関係を定義することで, 抽象化したシステム上での軌道から対応した実際のシステムの軌道を生成することが可能である. しかし, 実際のロボットシステムは非線形なダイナミクスを持つため, 抽象化システムとの状態間の関係を定義し, これらの手法を用いて抽象化したシステムを構築することは難しい. そのため, 手先座標といったタスクに関連するロボットの状態に対して, 適当な近似システムを定義し抽象化システムとする手法も検討されている [13, 14]. 本論文でも, これらの手法と同様に近似システムを抽象化システムとして用いている. この際, 抽象化したシステムの状態や近似システムの定義によって, 実際のロボットシステムの特性をどの程度反映可能なのかが異なっている.

加えて, 抽象化システムの構築方法によってロボットの軌道を生成する手法も異なる. 抽象化したシステムを離散状態システムとして表現した場合, 与えられた時相論理命題に対応するオートマトンとの並行システムを考えることで, 価値反復法など強化学習の手法によってタスクを達成する状態遷移列を生成することができる [8, 15, 16]. しかし, 離散状態システムとして表現する場合, Simulation, Bisimulation や マルコフ決定過程等を用いて表現することになり, 上記で述べた問題に加えて, 並行システムの状態集合が大規模になるといった問題がある.

連続状態システムとして表現した場合, 軌道最適化問題として定式化し, 混合整数計画問題や非線形計画問題を解くことで軌道を生成することができる [9, 10]. 混合整数計画問題を用いて軌道を生成する場合, 原子論理命題の真偽を離散 (0-1) 変数を用いて表現し, 各時相論理命題を 0-1 変数の時系列に対する四則演算を用いて表現することができる [9]. 加えて, 抽象化したシステムとしてハイブリッドシステムである混合論理動的 (MLD) モデルが用いられている場合にも, タスクを達成する軌道を生成することが可能である. しかし, 混合整数計画問題として定式化する場合, 非線形制約が存在したり非線形システムに適用する際に計算コストが大きいといった問題がある. 一方で, 非線形計画問題を用いて軌道を生成する場合, 原子論理命題の真偽をロバストネス関数用いて表現し, 各時相論理命題のロバストネス関数を softmin, softmax 関数を用いて連続な関数として表現することができる [3]. 加えて, 非線形システムに対して混合整数計画問題を用いる場合と比較して, 計算時間が短いといった利点がある. そのため, 対象とするロボットシステムやタスクによって抽象化したシステムの非線形性やどのクラスの問題として定式化するかが重要であり, 特に, 混合整数計画問題として定式化する場合は, 近似システムとして実際のロボットの動的特性を反映した動的線形システムを構築することが必要であり, 近似システムの構築方法には検討の余地が存在する.

1.1.3 タスク達成を考慮した制御手法

タスクを達成するためには, 抽象化したシステムで生成した軌道を基に実際のシステム上で制御を行い, 行動を実行する必要がある. 時相論理命題として表現されたタスクを達成する

ための制御手法として, Model Predictive Control (モデル予測制御, MPC) [17,18] や Control Barrier Functions (制御バリア関数, CBFs) [19–22] を用いた制御器が提案されている. 特に, CBF はシステムの軌道を特定の領域内に保つための制御入力を決定する際の制約条件として用いられ [23], CBF を用いた制約付き最適化問題は二次計画問題として定式化することができ, これは非線形 MPC と比較して計算時間が短い.

CBF を用いて時相論理によって表現されたタスクを実行するための制御器を設計するうえで, CBF をどのように設定するかが課題となる. 特に, STL で表現された制限時間があるようなタスクの場合, 先行研究で提案されている CBF の設計手法では, 補助的な時間関数の設計といった手順を用いて STL のロバストネス関数から CBF を設計する必要がある [19, 24]. 加えて, CBF 等の制約条件付き最適化問題を解くことによって制御入力を生成する手法では, 最適化問題の解が存在しない場合に, 制御入力が得られず制御不能になるといった問題が存在する.

さらに, 行動を実行している際に想定している行動では与えられたタスクが達成できなくなる場合が存在する. そのような場合の例として, 複数のロボットが行動を実行している際に, ロボットが不具合を起こして行動の実行が不可能になる場合である. この際, タスクを達成するためには残ったロボットが行動の変更を行い, 不具合を起こしたロボットの行動を補う必要がある.

1.2 本論文の目的

本論文では，時相論理命題で表現されたタスクをロボットシステムが達成するための制御手法の性能および拡張性の改善を目的とし, 行軌道最適化手法や制御手法を提案する. また, 本論文では有限時間で命題の真偽が定まる syntactically co-safe LTL (scLTL) 命題 [25] および同様な形式で記述できる STL 命題で表現可能なタスクを対象とし, それらのタスクに対して有限時間のロボットシステムの行動計画や軌道を生成し, それを基に制御を行うことでタスクを達成する.

本論文では, Fig. 1.1 に示す複数の段階に分かれた構造を持つ制御アーキテクチャを仮定する.

高レベル制御器では，制御対象であるロボットシステムに対して, 抽象化されたロボットシステムを定義する. この抽象化したシステムに対して, LTL や STL 命題で与えられたタスクを達成するための行動や軌道を最適化によって生成する. この際, 抽象化したシステムの設計方法によって, 実際の制御対象であるロボットの特性をどの程度反映することができるかが異なっている. そのため, 本論文ではロボットシステムの抽象化手法について検討を行う. 加えて, 抽象化したシステムに対して与えられた時相論理命題を満たす軌道を計算できることを示す. さらに, Event-based STL 命題で表現された外的事象が関与するタスクに対する軌道計画手法を提案する.

低レベル制御器では, 高レベル制御器で生成した行動計画や軌道を用いて, 実際の制御対象であるロボットシステムを制御する. 本論文では, 与えられた時相論理命題を達成するた

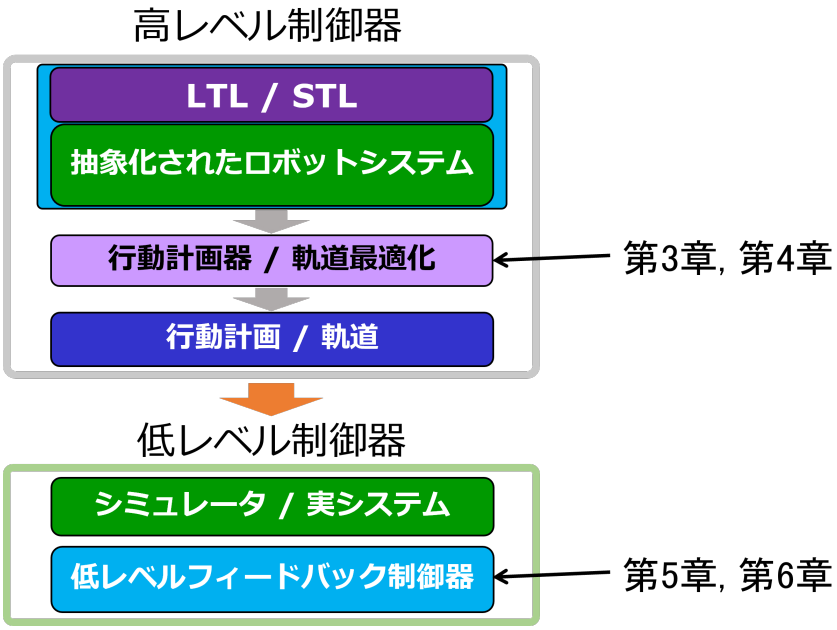

Fig. 1.1: Control architecture in this paper

めに時間的な制約を考慮した CBF を用いた制御手法を提案する. 加えて, 軌道を分布として表現し, 行動中に得られた観測情報から分布を更新し行動を変更する手法についても検討を行う.

1.3 論文構成

本論文の構成を以下に示す.

1 章 序論

論文の全体としての研究背景と目的について, 先行研究との対比を行いながら紹介を行った.

2 章 事前知識

本章では, 本論文に関する事前知識として, まず時相論理の基礎的な内容について説明する. 続いて, 時相論理に基づくロボットシステムの制御に関する基礎として, ロボットシステム上での時相論理を用いたタスク表現について説明する.

3 章 混合整数計画問題による軌道最適化

本章では, 線形時相論理で表現されたタスクに関する軌道最適化を混合整数計画問題として定式化し, ハイブリッドシステムまたは非線形システムに適用する. 非線形システムに適用する際, 混合整数凸計画や Koopman 作用素を用いた近似線形システムを導入することで, 混合整数線形計画問題として定式化する. 最後に, 数値シミュレーション

にて, いくつかのロボットシステムに提案手法を適用し, 各ロボットシステム上で与えられた時相論理が達成されることを示す.

4 章 不確かな外的事象に関する命題への軌道最適化

本章では, 発生時刻が不確かな外的事象に関する命題を達成するための軌道最適化手法を提案する. 外的事象はロボットの状態に依存せず発生する事象であり, 本章では発生時刻に関する分布が既知であると仮定する. その分布に基づき, 複数の発生時刻を想定して与えられた命題を達成する軌道を生成することが可能な軌道最適化を非線形計画問題として定式化する. 最後に, 不確かな外的事象に関する命題が与えられた移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

5 章 時間制約をもつ信号時相論理に対する制御

本章では, 時間制約をもつ様相作用素を含む信号時相論理を達成するための制御バリア関数を用いた制御手法を提案する. 与えられた時相論理命題に対応する時間オートマトンと, 3,4 章で述べた軌道最適化等の手法によって生成された軌道に対応するパスを考える. このパスに含まれる時間オートマトン上の各ロケーションの遷移に対応する命題を達成するために, CBF を拡張した制御器を提案する. この制御器を用いてロケーション間の遷移を逐次達成することで, 最終的に与えられた本来の命題が達成されることを示す. 最後に, 移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

6 章 信号時相論理に対する確率的動作プリミティブの再配合

本章では, 複数のシステムでタスクを行う際に, 他のロボットの行動に合わせて自身の行動を修正するための軌道最適化手法を提案する. まず 3,4 章で述べた軌道最適化によって得られた軌道から, 確率的動作プリミティブを生成する. 生成された確率的動作プリミティブを用いて, 他のシステムの行動が属している動作プリミティブをカルマンフィルターを用いて推定する手法を提案する. その後, 推定された動作プリミティブを基に, 自身の動作プリミティブを更新することで, 他のシステムの行動を考慮して与えられた命題を達成する軌道を生成する手法を提案する. 最後に, 移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

7 章 結論

本章では，本論文で得られた成果の総括と，我々の研究の今後の発展と課題などについて述べる.

数学記号

Mathmatical symbols

$\mathbb{R}$　実数全体の集合 ($\mathbb{R}_+$ は正の実数全体の集合, $\mathbb{R}_{0+}$ は 0 を含む正の実数全体の集合)

$\mathbb{R}^n$　n 次元実数ベクトル全体の集合

$\mathbb{R}^{m\times n}$　$m \times n$ 次元実数行列全体の集合

$\mathbb{C}$　複素数全体の集合

Int　集合の内部

$0_{m\times n}$　m 行 n 列の零行列

I_n　n 次元単位行列

A^T　行列 $A \in \mathbb{R}^{m\times n}$ の転置

A^{-1}　正方行列 $A \in \mathbb{R}^{n\times n}$ の逆行列

$L_f h$　関数 $h(x) : \mathbb{R}^n \to \mathbb{R}^p$ の関数 $f(x) : \mathbb{R}^n \to \mathbb{R}^n$ に沿った微分　$L_f h = \frac{\partial h}{\partial x} f(x)$

$\mathrm{diag}(a_1, \ldots, a_n)$ スカラー $a_i \in \mathbb{R},\ (i = 1, ...n)$ を対角要素にもつ対角行列

$\mathcal{N}(\mu_x, \Sigma_x)$ 平均値 μ_x，分散 Σ_x を持つガウス分布

Logical symbols

::=　Backus-Naur 記法, BNF

$\neg$, $\wedge$, $\vee$, $\Rightarrow$　論理記号 (否定, 論理積, 論理和, 実質含意)

$\models$　Double-turnstile

Other symbols

$\mathbf{x}(t_0 : t_N)$ $\mathbf{x}$ の時系列データ, 無限長の場合, $\mathbf{x}(t_0 :)$

$\mathbf{x}_{t_0:t_N}$ $\mathbf{x}(t_0 : t_N)$ を省略した表現

第2章　事前知識

本章では, 本論文に関する事前知識として, まず線形時相論理の基礎的な内容について説明する. 続いて, 時相論理に基づくロボットシステムの制御に関する基礎として, ロボットシステム上での時相論理を用いたタスク表現について説明する. その後, 線形時相論理を拡張した信号時相論理について説明する.

2.1　線形時相論理

2.1.1　クリプキモデル

クリプキモデルとは, 原子論理命題の集合 AP 上で定義されたクリプキ構造 (S, R) を持つ組 (S, R, V) である [26].

- (S, R) はクリプキ構造
- $V : S \times \mathsf{AP} \to \{\top, \bot\}$ は付加関数

ここで, クリプキ構造は次のような組 (S, R) である.

- S は空でない集合, S の各元 $s \in S$ を状態または可能世界と呼ぶ
- R は S 上の二項関係

2.1.2　線形時相論理

線形時相論理 (LTL) は様相論理の１つであり, 状態遷移列の性質を表現する論理である. LTL 論理式の集合は, 以下の BNF によって定義される.

$$\varphi ::= \pi \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\, \bigcirc \varphi \,|\, \varphi_1 \mathbf{U} \varphi_2, \tag{2.1}$$

ここで, π は原子論理命題, φ は時相論理式, $\bigcirc$ は *next* 作用素, $\mathbf{U}$ は *until* 作用素とする. $\vee$ 等その他の論理演算子や $\mathbf{G}$ (always) や $\mathbf{F}$ (eventually) といった様相作用素は上記の作用素によって定義できる:

$$\mathbf{G}\varphi = \top \mathbf{U} \varphi, \ \mathbf{F}\varphi = \neg\mathbf{G}\neg\varphi$$

LTL はクリプキモデルの各パスにおいて真偽が定まる. ここで, パスとはクリプキ構造上の状態がなす無限の列 $\mathbf{s} = \mathbf{s}_0, \mathbf{s}_1, \ldots$ であり, パス上の各状態に対して $(\mathbf{s}_i, \mathbf{s}_{i+1}) \in R$ となるもののことである.

本論文における LTL の意味論は次のように再帰的に定義される.

$$
\begin{aligned}
(\mathbf{s}, t) &\models \pi && \iff \mathbf{s}_t \models \pi \\
(\mathbf{s}, t) &\models \neg\varphi && \iff \neg((\mathbf{s}, t) \models \varphi) \\
(\mathbf{s}, t) &\models \varphi_1 \wedge \varphi_2 && \iff (\mathbf{s}, t) \models \varphi_1 \wedge (\mathbf{s}, t) \models \varphi_2 \\
(\mathbf{s}, t) &\models \bigcirc\varphi && \iff (\mathbf{s}, t+1) \models \varphi \\
(\mathbf{s}, t) &\models \varphi_1 \,\mathbf{U}\, \varphi_2 && \iff \\
& && \quad \exists t' \in \mathcal{T} \; s.t. \; (\mathbf{s}, t') \models \varphi_2 \\
& && \quad \wedge \forall t'' \in [t,\, t'] \,, (\mathbf{s}, t'') \models \varphi_1 .
\end{aligned}
$$

2.2 ロボットシステム上での時相論理

ここでは, 本論文で扱うロボットシステムについて説明する. 加えて, ロボットシステムをクリプキモデルとして扱うことができることを示し, 本論文で扱う時相論理について述べる.

本論文で扱うロボットシステムは, 以下の式で表されるダイナミクスを持つシステムである:

$$\mathbf{x}^+ = f(\mathbf{x}(t), \mathbf{u}(t)), \tag{2.2}$$

ここで, $t \in \mathcal{T}$ は時刻, $\mathbf{x} \in \mathcal{X} \subset \mathbb{R}^{n_\mathbf{x}}$ は状態, $\mathbf{u} \in \mathcal{U} \subset \mathbb{R}^{n_\mathbf{u}}$ は入力, $f : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}^{n_\mathbf{x}}$ は遷移関数, $\mathbf{x}^+$ は $\mathbf{x}$ の時間微分, または時刻 $t+1$ での $\mathbf{x}$ である.

遷移関数 f がロボットシステム上のすべての状態に対して定義されているとする. この際, ロボットシステム上の軌道 $\mathbf{x}(t_0 :) = \mathbf{x}(t_0), \mathbf{x}(t_1), \ldots$ が得られた際, 状態 $\mathbf{x}(t_i), \mathbf{x}(t_{i+1})$ に対して $\mathbf{x}(i+1) = F(\mathbf{x}(i+1), u(i))$ を満たす関数 $F : \mathbb{R}^{n_\mathbf{x}} \times \mathbb{R}^{n_\mathbf{u}} \to \mathbb{R}^{n_\mathbf{x}}$, 入力関数 $u(t) : \mathbb{R} \to \mathbb{R}^\mathbf{u}$ が存在する. そのため, 状態 $\mathbf{x}(t_i), \mathbf{x}(t_{i+1})$ は二項関係を持つ. 従って, 本論文で扱うロボットシステムに対してクリプキ構造を考えることができる. 加えて, 付加関数 V を以下のように定義することでクリプキモデルとして扱うことができ, ロボットシステムの振る舞いを LTL を用いて表現することができる.

定義 1. ロボットシステム上の軌道 $\mathbf{x}$ が与えられたとき, 付加関数 V は各時刻の状態 $\mathbf{x}(t)$ に対して定義される, $V : \mathbb{R}^{n_\mathbf{x}} \times \mathsf{AP} \to \{\top, \bot\}$.

本論文では以後, パスとしてロボットの軌道 $\mathbf{x}$ を扱うことで, 命題の意味論を $(\mathbf{x}, t) \models \varphi$ で表わすこととする.

加えて, LTL 命題として表現されたタスクが無矛盾であることをチェックすることは一般に計算量が非常に大きくなる [27]. 従って, 実際の応用にあたっては, 計算が容易に抑えられ

るような LTL のサブクラスである GR(1) や scLTL などに限定して用いるなどの工夫が必要になる [25, 28]. 特に, 有限時間で命題の真偽が定まる scLTL で表現されるタスクに対して本論文で提案する手法を適用することで, 時相論理命題を満たす有限時間のシステムの軌道を生成することが可能である.

2.3 信号時相論理

Signal Temporal Logic (STL) は LTL を拡張した様相論理である. STL の原子論理命題 AP, π の充足関係は以下の様にロバストネス関数 $\rho_\pi : \mathbb{R}^{n_x} \to \mathbb{R}$ によって評価される [5]:

$$\pi = \begin{cases} \top & \text{if } \rho_\pi(\mathbf{x}) \geq 0 \\ \bot & \text{if } \rho_\pi(\mathbf{x}) < 0 \end{cases}. \tag{2.3}$$

本論文では, ロバストネス関数は微分可能な連続関数と仮定する.

STL 論理式の集合は LTL と同様に以下の BNF で定義される.

$$\varphi ::= \pi \,|\, \neg\varphi \,|\, \varphi_1 \wedge \varphi_2 \,|\, \varphi_1 \mathbf{U}_I \varphi_2, \tag{2.4}$$

ここで $I = [t_1, t_2]$, $t_1 < t_2$ [29], $\mathbf{U}_I$ は時間区間 I に対する *until* 作用素とする. $\mathbf{G}_I$ (always) や $\mathbf{F}_I$ (eventually) といった様相作用素は LTL と同様に上記の作用素によって定義できる. STL の意味論は以下の様に再帰的に定義される:

$$\begin{aligned}
&(\mathbf{x},\, t) \models \pi && \iff \rho_\pi(\mathbf{x}(t)) \geq 0 \\
&(\mathbf{x},\, t) \models \neg\varphi && \iff \neg((\mathbf{x},\, t) \models \varphi) \\
&(\mathbf{x},\, t) \models \varphi_1 \wedge \varphi_2 && \iff (\mathbf{x},\, t) \models \varphi_1 \wedge (\mathbf{x},\, t) \models \varphi_2 \\
&(\mathbf{x},\, t) \models \varphi_1 \,\mathbf{U}_{[t_1,t_2]}\, \varphi_2 && \iff \\
& && \quad \exists t' \in [t + t_1,\, t + t_2] \; s.t. \; (\mathbf{x}, t') \models \varphi_2 \\
& && \quad \wedge \forall t'' \in [t,\, t'] \,, (\mathbf{x}, t'') \models \varphi_1.
\end{aligned}$$

このとき, $\varphi_\wedge = \bigwedge_i \varphi_i$ や $\varphi_\vee = \bigvee_i \varphi_i$ といった命題に対するロバストネス関数は以下の様に計算できる:

$$\rho_{\varphi_\wedge}(\mathbf{x}) = \min_i \rho_{\varphi_i}(\mathbf{x}), \tag{2.5}$$

$$\rho_{\varphi_\vee}(\mathbf{x}) = \max_i \rho_{\varphi_i}(\mathbf{x}). \tag{2.6}$$

また, 上記のロバストネス関数は min / max を近似することで, 下記の様になめらかな関数として表現できる [19]:

$$\min_i \rho_{\varphi_i}(\mathbf{x}) \geq -\frac{1}{\kappa} \ln \left(\sum_{i=1} \exp(-\kappa \rho_{\varphi_i}(\mathbf{x})) \right), \tag{2.7}$$

$$\max_i \rho_{\varphi_i}(\mathbf{x}) \leq \frac{1}{\kappa} \ln \left(\sum_{i=1} \exp(\kappa \rho_{\varphi_i}(\mathbf{x})) \right), \tag{2.8}$$

ここで $\kappa > 0$ は定数とする.

第3章　混合整数計画問題による軌道最適化

3.1　はじめに

混合整数計画問題による定式化では, 時相論理に関する離散変数とロボットシステムに関する連続変数を同時に扱うことが可能である. しかし, 混合整数非線形計画問題の場合では計算コストが大きい傾向にあり, 非線形システムや非線形制約に対して適用することが難しい.

これを踏まえ, 本章では非線形システムや非線形制約を線形システムで近似し, 混合整数計画問題として定式化することを検討する. まず, 混合整数計画問題による定式化の準備として, 本章で扱う混合論理動的 (MLD) モデル, 混合整数凸計画, 拡大線形化について説明する. 次に, 混合整数計画問題による定式化ついて説明する. その後, 混合整数凸計画を用いることで, クォータニオンベースの $SO(3)$ システムに対して混合整数線形計画問題の定式化が可能であることを示す. 加えて, 拡大線形化を用いることで, 非線形システムに対しても混合整数線形計画問題の定式化が可能であることを示す. 最後に, 数値シミュレーションにて, いくつかのロボットシステムに提案手法を適用し, 各ロボットシステム上で与えられた時相論理命題が達成されることを示す

3.2　準備

ここでは, MLD モデル, 混合整数凸計画, 拡大線形化について説明する. 特に, MLD モデルとして, ロボットのタスクに現れるハイブリッドシステムの例であるアーム型ロボットのピックアンドプレイスタスクにおける手先座標の近似システムについて説明する.

3.2.1　混合論理動的モデル

ハイブリッドシステムは連続的な状態と離散的な状態の両方を含むシステムである.

混合論理動的 (MLD) モデルはハイブリッドシステムの表現の 1 つであり, 本論文では MLD モデルを下記の様に記述する.

$$\begin{cases} \mathbf{x}^+ = A\mathbf{x} + B_1\mathbf{u} + B_2\mathbf{z} + B_3\varpi + B_4\vartheta + B_5\mathbf{p} \\ C\mathbf{x} + D_1\mathbf{u} + D_2\mathbf{z} + D_3\varpi + D_4\vartheta + D_5\mathbf{p} \leq E, \end{cases} \tag{3.1}$$

ここで $\mathbf{z}(t) \in \mathbb{R}^{n_\mathbf{z}}$ は連続値の補助変数, $\varpi(t) \in \{0,1\}^{n_\varpi}$ は MLD モデルの離散モードに関する離散値の補助変数, A, B_i, C, D_i, E, $(i = 1,2,3,4,5)$ はシステム行列とする. 加えて, $\vartheta(t) \in \{0,\,1\}^{n_\vartheta}$, $\mathbf{p}(t) \in [0,\,1]^{n_\mathbf{P}}$ はそれぞれ LTL に関係する離散値と連続値の変数である. $B_2 = 0_{n_\mathbf{x} \times n_\mathbf{z}}, D_2 = 0_{n_E \times n_\mathbf{z}}$ とすることで, 複数のモードを持たない線形システムを表現することができる.

ピッキングモデルのハイブリッドシステム表現

ピックアンドプレイスタスクに対して, アーム型ロボットの手先座標とオブジェクトの中心座標を状態とする近似システムを考える. 近似システムの状態として, 以下を定義する.

$$\mathbf{x} = \begin{bmatrix} X_{\mathrm{a},1}^\mathsf{T} & X_{\mathrm{o},1}^\mathsf{T} & \dots & X_{\mathrm{o},n_o}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{3.2}$$

$$X_{\mathrm{a}i} = \begin{bmatrix} x_{\mathrm{arm},i}^\mathsf{T} & \dot{x}_{\mathrm{arm},i}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{3.3}$$

$$X_{\mathrm{o}i} = \begin{bmatrix} x_{\mathrm{obj},i}^\mathsf{T} & \dot{x}_{\mathrm{obj},i}^\mathsf{T} \end{bmatrix}^\mathsf{T}, \tag{3.4}$$

ここで $X_{\mathrm{a},i}$ は手先 i の状態, $X_{\mathrm{obj},i}$ はオブジェクト i の状態, n_o はオブジェクトの数とする. また, x_*, $\dot{x}_*$ は各手先とオブジェクトの位置と速度とする.

ピッキングのモデル化

ピッキングモデルの挙動を Fig.3.1 に示す. 近似システム上では, アームがオブジェクトをつかんでいるとき, オブジェクトの速度は手先の速度と一致すると仮定する. ここで, $\varpi_{ij} \in \{0,1\}$ を命題 “アーム i がオブジェクト j を掴んでいる” の論理値を示す変数とする. このとき, ピッキング動作による状態遷移は以下の様に記述できる:

$$\varpi_{ij} = 1 \Rightarrow \dot{x}_{\mathrm{arm},i} = \dot{x}_{\mathrm{obj},j}. \tag{3.5}$$

MLD システムを用いて上記の状態遷移を含むシステムは次のように記述できる:

$$\mathbf{x}^+ = A^{\neg\varpi}\mathbf{x} + B^{\neg\varpi}u + \varpi((A^\varpi - A^{\neg\varpi})x + (B^\varpi - B^{\neg\varpi})u), \tag{3.6}$$

ここで $A^{\neg\varpi}, B^{\neg\varpi}$ は $\varpi = 0$ のときのシステム行列であり, A^ϖ, B^ϖ は $\varpi = 1$ のときのシステム行列とする. 例として, MLD システムは次のように表わすことができる (Fig. 3.1).

$$\begin{bmatrix} x_{\mathrm{obj},1} \\ \dot{x}_{\mathrm{obj},1} \end{bmatrix} = \begin{bmatrix} x_{\mathrm{obj},1} + \Delta T \dot{x}_{\mathrm{obj},1} \\ \sum_i \varpi_{i1} \dot{x}_{\mathrm{arm},i} \end{bmatrix} \tag{3.7}$$

ここで ΔT は時間ステップとする.

また, ϖ_{ij} は次の命題を満たす:

$$\varpi_{ij} = 1 \Rightarrow \varphi_{\vartheta_{ij}}, \tag{3.8}$$

ここで命題 $\varphi_{\vartheta_{ij}}$ は "$x_{\mathrm{arm},i} \in H_{ij}$"を示す命題, H_{ij} はアーム i とオブジェクト j の状態に関係するポリトープである:

$$H_{ij} = \{x_{\mathrm{arm},i} \in \mathbb{R}^{2 \text{ or } 3} : C_{ij}(x_{\mathrm{arm},i} - x_{\mathrm{obj},j}) \leq e_{ij}\}, \tag{3.9}$$

ここで C_{ij} は定数行列, e_{ij} は定数ベクトルとする. この命題は, アームがオブジェクトに関するポリトープの内側にあるときに限り, ピッキング動作を実行することが可能であることを意味している. 式 (3.7) で表現されるダイナミクスと, 命題 (3.8) とポリトープ (3.9) に関する関係式を後述する方法によって線形制約で表現することで, ピッキングモデルを MLD システムとして式 (3.1) の形式で表現できる.

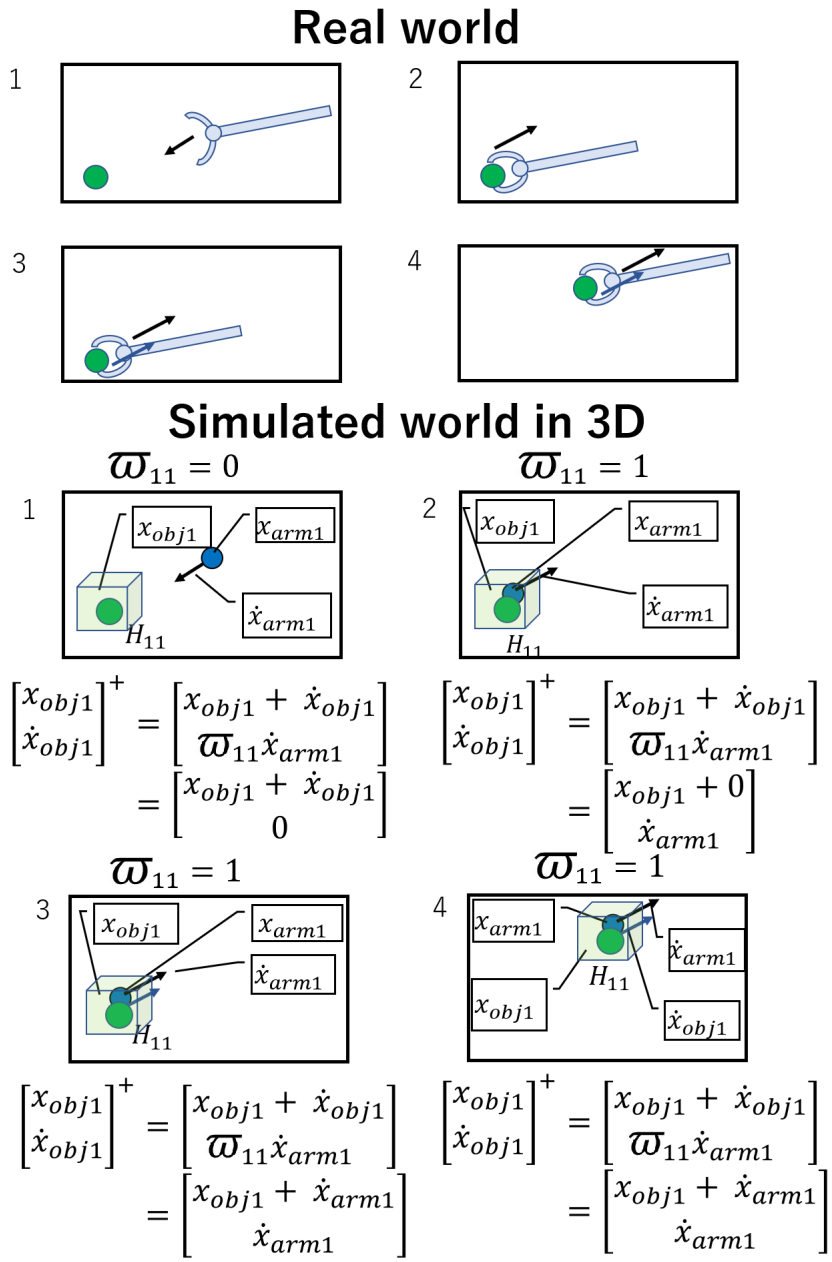

Fig. 3.1: Modeling of picking. Top: the robot behavior in the real world. Bottom: the behavior of MLD model in the simulated world. $x_{\mathrm{arm},1}$ and $x_{\mathrm{obj},1}$ are the position of the arm 1 and the object 1, respectively. H_{11} is the bounding box of the object and proposition $\varphi_{\vartheta 11}$ is $x_{\mathrm{arm},1} \in H_{11}$. When the arm enters the bounding box H_{11}, $\varphi_{\vartheta 11}$ becomes 1. We can change ϖ_{11} to 1 and when $\varpi_{11} = 1$, the velocity constraint of the object is activated, and it follows the arm. This is the modeling of picking. Since the proposition $\varpi_{11} = 1$ is a sufficient condition of (3.8), the arm can release objects after picking the object.

3.2.2 混合整数凸計画

双線形項 xy を混合整数凸計画 (MICP) で近似する手法について説明する [30]. まず, スカラ変数 $x \in [\psi_0^x, \psi_{N_w}^x]$, $y \in [\psi_0^y, \psi_{N_w}^y]$ に対して, x, y の定義域を N_w 個の区間 $[\psi_i^*, \psi_{i+1}^*]$ に分割する. その後, いくつかの補助変数を導入し, 次の制約を用いて変数 w_{xy} によって双線形項 xy を近似する (Fig.3.2):

$$x = \sum_i \lambda_{x,i}\psi_i^x,\ y = \sum_j \lambda_{y,j}\psi_j^y, \tag{3.10}$$

$$w_{xy} = \sum_i \sum_j \gamma_{ij}\psi_i^x\psi_j^y, \tag{3.11}$$

$$\sum_j \gamma_{ij} = \lambda_{x,i},\ \sum_i \gamma_{ij} = \lambda_{y,j}, \tag{3.12}$$

ここで $\lambda_{x,i}$, $\lambda_{y,j}$, $\gamma_{ij} \in [0, 1]$ $(i, j = 0, \ldots, N)$ は連続値の補助変数であり, 特に $\lambda_{x,i}$, $\lambda_{y,j}$ は以下のように定義される special ordered sets of type 2 (SOS2) 制約を満たす変数とする.

定義 2. (SOS2 制約) もし変数 λ_i $(i = 0, \ldots, N)$ が SOS2 制約を満たすならば, λ_i は次の条件を満たす;

- 最大で 2 つの λ_i は非ゼロである.
- もし 2 つの λ_i が非ゼロであれば, それらの添え字は隣り合っている.
- λ_i の合計は 1 である.

SOS2 制約は $\log_2 N_w$ 個のバイナリ変数を用いて混合整数線形制約として実装することができる [31].

同様に二乗項についても下記の様に近似することができる: $w_{x^2} \approx x^2$,

$$x = \sum_i \lambda_{x,i}\psi_i^x, \tag{3.13}$$

$$w_{x^2} = \sum_i \sum_j \gamma_{ij}\psi_i^x\psi_j^x, \tag{3.14}$$

$$\sum_j \gamma_{ij} = \lambda_{x,i},\ \sum_i \gamma_{ij} = \lambda_{x,j}. \tag{3.15}$$

本章の数値実験では, 特に記載がない場合, これらの制約について分割区間を $N_w = 4$ として実装している.

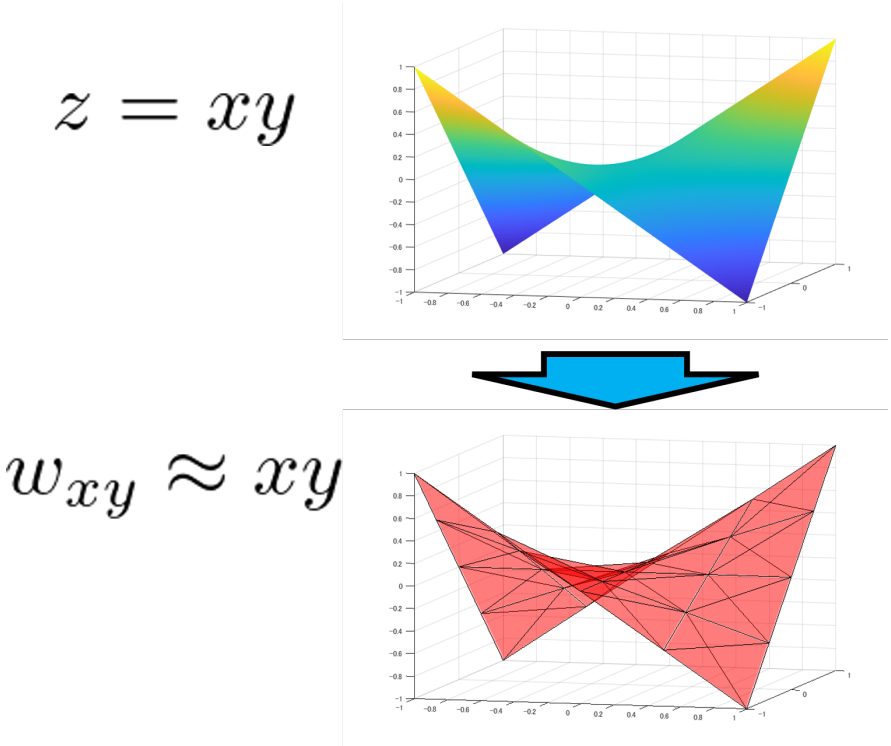

Fig. 3.2: An approximation of the quadratic term xy by MICP

3.2.3 拡大線形化

次の微分方程式で表現される連続時間非線形システムを考える

$$\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t)). \tag{3.16}$$

ここで $\Gamma(t, \mathbf{x})$ をシステム (3.16) の flow map, $\mathcal{F}$ を無限次元の観測関数 $g : \mathcal{X} \to \mathbb{C}$ の空間とする. 連続時間の Koopman 作用素 $\mathcal{K}^\tau : \mathcal{F} \to \mathcal{F}$ は次の関係を満たす:

$$(\mathcal{K}^\tau g)(\cdot) = g \circ \Gamma(\tau, \cdot), \tag{3.17}$$

ここで $\circ$ は写像の合成を表わす:

$$(\mathcal{K}^\tau g)(\mathbf{x}(t), \bar{\mathbf{u}}(t)) = g(\mathbf{x}(t+\tau), \bar{\mathbf{u}}(t)). \tag{3.18}$$

$\bar{\mathbf{u}}$ は時間区間 $[t, t+\tau]$ で一定値の入力とする.

無限小生成作用素 $\mathcal{K}^\tau$ を用いてシステムの軌道に沿った観測関数の時間微分は以下の様に書ける;

$$\mathcal{K}g = \frac{d}{dt}g, \tag{3.19}$$

すなはち,

$$(\mathcal{K}g)(\mathbf{x}(t),\mathbf{u}(t)) = \frac{\partial g}{\partial \mathbf{x}} f(\mathbf{x}(t),\mathbf{u}(t)). \tag{3.20}$$

そのため, Koopman 作用素を用いることで非線形システムを線形システムとして表現することができる. 本来の Koopman 作用素は無限次元であるが, 実際に制御などで用いる場合は, Koopman 作用素を有限次元の観測関数を用いた拡大状態空間で近似する. この際, 近似するシステムに元々含まれていた状態に対して, 新たに追加する状態を拡大状態と呼ぶことにする. 加えて, 本来の Koopman 理論においては入力のない自励系について扱っているが, 入力のあるシステムへの理論の拡張についても研究されている [32] [33].

Koopman 作用素の近似を用いることで, 非線形システムを以下のように線形システムとして表現できる.

$$\dot{\xi} = A_\xi \xi + B_\xi \mathbf{u}, \tag{3.21}$$

$$\mathbf{x} = C_\xi \xi, \tag{3.22}$$

ここで $\xi \in \mathbb{R}^{n_\xi}$ は拡大状態空間上で定義される高次 $n_\xi \geq n_x$ の状態変数である:

$$\xi = \begin{bmatrix} g_1(\mathbf{x}) \\ g_2(\mathbf{x}) \\ \vdots \\ g_{n_\xi}(\mathbf{x}) \end{bmatrix}. \tag{3.23}$$

特に, もし $\xi = [x^\mathsf{T}, \mathbf{g}^\mathsf{T}]^\mathsf{T}$ ならば, $C_\xi = [\mathbb{I}_{n_\mathbf{x},n_\mathbf{x}}, 0_{n_\mathbf{x},n_\xi - n_\mathbf{x}}]$ となる. 行列 A_ξ および B_ξ は定数行列であり, この拡大線形システムは非線形システムを大域的に近似したシステムになっている.

しかし, ロボットなどのシステムの多くは, 線形な入力伝達項をもつ線形システムとして表現することが難しいことが知られている. そのため, Koopman 作用素を用いて表現可能なシステムのクラスを拡大するために, 状態と入力の双線形項を持つようなモデルが提案されている [34]:

$$\dot{\xi} = A_\xi \xi + B_\xi \mathbf{u} + \sum_i^{n_\mathbf{u}} H_{\xi,i} \xi u_i, \tag{3.24}$$

$$\mathbf{x} = C_\xi \xi, \tag{3.25}$$

ここで, $H_{\xi,i}\,(i = 1, \ldots, n_u)$ は定数行列である.

観測関数の選定方法や及び定数行列 A_ξ, B_ξ, C_ξ, and $H_{\xi,i}$ の計算方法については多くの手法が提案されており, 例としては Extended Dynamic Mode Decomposition (EDMD) [35] や deep learning を用いた手法 [36] が挙げられる. 本章で提案する手法は, 上記の述べた計算手法に依存せず適用可能である. 加えて, Koopman 作用素および本章で提案する手法は離散時間システムに対しても適用することができ, 本章内でも離散時間システムについて考えるものとする.

3.3 混合整数計画問題による定式化

LTL 命題で表現されたタスク φ を達成する MLD システムの軌道を生成する問題を混合整数線形計画問題として定式化する. まず, LTL 命題 φ を線形制約として表現する. その後, 前節で定義した MLD システムを用いることで混合整数線形計画問題または二次計画問題として定式化できることを示す.

3.3.1 線形時相論理命題の線形制約

LTL 命題 φ に対して, サブ命題 φ_i $(i = 1, 2, , \dots)$ を定義する.

Example 1. LTL 命題:

$$\varphi = \mathbf{G}\neg(\pi_1 \wedge \pi_2) \wedge \mathbf{F}\left((\pi_3 \vee \pi_4) \wedge \bigcirc\pi_5\right),$$

に関するサブ命題は,

$$\begin{aligned}
&\varphi_1 = \mathbf{G}\varphi_3,\ \varphi_2 = \mathbf{F}\varphi_4,\\
&\varphi_3 = \neg\varphi_5,\ \varphi_4 = \varphi_6 \wedge \varphi_7,\\
&\varphi_5 = \varphi_8 \wedge \varphi_9, \varphi_6 = \varphi_{10} \vee \varphi_{11}, \varphi_7 = \bigcirc\varphi_{12},\\
&\varphi_8 = \pi_1,\ \varphi_9 = \pi_2,\ \varphi_{10} = \pi_3,\ \varphi_{11} = \pi_4,\ \varphi_{12} = \pi_5.
\end{aligned}$$

各サブ命題に対して, 原子論理の真偽を示す変数 $\vartheta_j(t) \in \{0, 1\}$, それ以外の命題の真偽を示す変数 $p_j(t) \in [0, 1]$ を定義する. ここで, p_0 は φ の真偽を示す変数とする. このとき, 有限な時間区間 $[0, T]$ におけるサブ命題の論理的関係は, 以下の LTL 命題 φ の真偽を示す変数 $p(t) = [0, 1]$ に関する制約条件を用いて逐次的に表現することができる [2].

$$\begin{aligned}
&\varphi = \neg\varphi_i \quad : p(t) = 1 - p_i(t)\\
&\varphi = \bigvee_{i=1}^{k} \varphi_i : p(t) \le p_i(t)\,(i = 1, \dots, k),\ p(t) \ge \sum_{i=1}^{k} p_i(t) - (k-1)\\
&\varphi = \bigwedge_{i=1}^{k} \varphi_i : p(t) \ge p_i(t)\,(i = 1, \dots, k),\ p(t) \le \sum_{i=1}^{k} p_i(t)\\
&\varphi = \bigcirc\varphi_i \ : p(t) = p_i(t+1)\\
&\varphi = \mathbf{G}\varphi_i \ : p(t) \le p_i(\tau)\,(i = 1, \dots, T),\ p(t) \ge \sum_{\tau=t}^{T} p_i(\tau) - (T-t)\\
&\varphi = \mathbf{F}\varphi_i \ : p(t) \ge p_i(\tau)\,(i = 1, \dots, T),\ p(t) \le \sum_{\tau=t}^{T} p_i(\tau)
\end{aligned}$$

until 演算子を用いた命題 $\varphi = \varphi_i \mathbf{U} \varphi_j$ に関しても新たな補助変数を導入することで制約条件として表現することができる [2]. 原子論理命題以外の命題の真偽を示す変数の値は, 原理論

理命題の真偽を示す離散変数に依存して決定されるため, 連続変数である命題の真偽を示す変数 p_j も, 制約条件を満たすならば 0-1 変数のように振る舞う. このため, 実装において, 真偽を示す変数は原子論理命題に関する変数 ϑ_j のみを離散変数とすれば十分である.

3.3.2 付加関数との対応関係

原子論理命題の真偽を返す付加関数は, ロボットシステム上での制約条件と真偽を示す変数との制約条件で表現することができる [2] [37]. 本章では, 各原子論理命題はロボットシステムの状態に関する線形制約の真偽, または, 離散入力に対応すると仮定し, 以下に示す big-M 線形拘束を用いて真偽を示す変数と制約条件との対応関係を実装している. big-M 線形拘束では, 十分に大きな定数 $M > 0$ を用いることで離散変数の振る舞いを拘束することができる. さらに, 目的に合わせて以下の 2 種類の制約条件を用いている.

1. $\vartheta = 1 \iff C_\vartheta \mathbf{x} \leq 1$ を満たす場合 [2]:

$$C_\vartheta \mathbf{x} \leq 1 + M(1-\vartheta),\ C_\vartheta \mathbf{x} \geq 1 + M(\epsilon - \vartheta)$$

2. $\vartheta = 1 \Rightarrow C_\vartheta \mathbf{x} \leq 1$ を満たす場合 [37]:

$$C_\vartheta \mathbf{x} \leq 1 + M(1-\vartheta)$$

ここで, $C_\vartheta^\mathsf{T} \in \mathbb{R}^{n_\mathbf{x}}$ は定数ベクトル, $\epsilon > 0$ は十分小さな定数である. 2 番目の big-M 線形拘束は, 線形制約が満たされている場合に $\vartheta = 1$ となることが可能であり, 前節で述べたピッキングモデルのピッキング動作を離散入力を用いて表現する場合などに用いられる.

3.3.3 混合整数線形 (二次) 計画問題による定式化

LTL 命題 φ と, ロボットシステムとして式 (3.1) で表わされる MLD モデルが与えられているとする. このとき, 最適化変数 $Z = [Z_\mathbf{x}^\mathsf{T}, Z_\mathbf{u}^\mathsf{T}, Z_\mathbf{z}^\mathsf{T}, Z_\varpi^\mathsf{T}, Z_\vartheta^\mathsf{T}, Z_\mathbf{p}^\mathsf{T}]^\mathsf{T}$ を定義する. Z_* は各変数 $\mathbf{x}$, $\mathbf{u}$, $\mathbf{z}$, ϖ, ϑ, $\mathbf{p}$ の時刻 $t = 0, \ldots, T$ での値をまとめたものとする.

このとき, 前小節から LTL 命題に関する制約条件は以下のように記述できる:

$$D_{\varphi,\vartheta} Z_\vartheta + D_{\varphi,\mathbf{p}} Z_\mathbf{p} \leq E_\varphi, \tag{3.26}$$

ここで $D_{\varphi,\vartheta}$, $D_{\varphi,\mathbf{p}}$ は定数行列, E_φ は定数ベクトルとする. 制約条件 (3.26) には, 与えられた LTL 命題に関する制約 $-p_0 \leq -1$ が含まれている. また, 式 (3.1) から MLD モデルに関する制約条件は以下の様に記述できる:

$$C_\mathbf{x} Z_\mathbf{x} + D_{\mathbf{x},\mathbf{u}} Z_\mathbf{u} + D_{\mathbf{x},\mathbf{z}} Z_\mathbf{z} + D_{\mathbf{x},\varpi} Z_\varpi + D_{\mathbf{x},\vartheta} Z_\vartheta + D_{\mathbf{x},\mathbf{p}} Z_\mathbf{p} \leq E_\mathbf{x}, \tag{3.27}$$

式 (3.26)(3.27) から, LTL 制約を満たす軌道を生成するための混合整数線形 (二次) 計画問題は以下のように定式化できる:

$$\begin{aligned} &\min_{Z} \ f_Z Z(+Z^{\mathsf{T}} Q_Z Z), \\ &\text{s.t. } D_Z Z \leq E_Z, \end{aligned} \tag{3.28}$$

ここで, D_Z は定数行列, f_Z, E_Z は定数ベクトル (Q_Z は定数行列) である.

$f_Z, (Q_z)$ は所望のシステムの振る舞いに応じて設定することができる. 例として, システムの状態や入力の $l_1(l_2)$-ノルムの最小化が挙げられる [38]. また, 命題の真偽を示す変数に対して重みづけを行うことで, システムの振る舞いを表現することも可能である. 例として, 特定の領域内に存在することを示す命題の真偽を示す変数に重みづけすることによって, システムのその領域での滞在時間を調節するといったことが可能である.

3.4 クォータニオンベースの $SO(3)$ の MICP を用いた行動計画

本節では, MICP による双線形項, 二乗項の近似を用いたクォータニオンベースの $SO(3)$ システムのダイナミクスを近似する手法を提案する. まず, オイラーパラメーター $SO(3)$ のダイナミクスを近似する手法について説明する. その後, 近似したダイナミクスをピックアンドプレイスモデルと組み合わせる方法について説明する.

3.4.1 オイラーパラメーターの近似

オイラーパラメーターは表現上の特異姿勢を持たない回転を表現する手法の一つであり, 次式のように定義される [39]:

$$\varepsilon = \mathbf{a} \sin \frac{\theta_{\mathbf{a}}}{2}, \ \eta = \cos \frac{\theta_{\mathbf{a}}}{2}, \tag{3.29}$$

ここで $\mathbf{a} \in [-1, 1]^3$ は回転軸を示す 3 次元ベクトル, $\theta_{\mathbf{a}}$ は回転軸 $\mathbf{a}$ 周りの回転量である. また, オイラーパラメーターはユニットクォータニオンと等価である. オイラーパラメーターのダイナミクスは次式で計算できる,

$$\begin{aligned} \frac{d}{dt}\begin{bmatrix}\varepsilon \\ \eta\end{bmatrix} &= \frac{1}{2}\begin{bmatrix}\eta I_{3\times 3} - \varepsilon\times \\ -\varepsilon^{\mathsf{T}}\end{bmatrix}\omega \\ &= \frac{1}{2}\begin{bmatrix}\eta\omega_1 + \varepsilon_3\omega_2 - \varepsilon_2\omega_3 \\ -\varepsilon_3\omega_1 + \eta\omega_2 + \varepsilon_1\omega_3 \\ \varepsilon_2\omega_1 - \varepsilon_1\omega_2 + \eta\omega_3 \\ -\varepsilon_1\omega_1 - \varepsilon_2\omega_2 - \varepsilon_3\omega_3\end{bmatrix}, \end{aligned} \tag{3.30}$$

ここで $\varepsilon = [\varepsilon_1, \varepsilon_2, \varepsilon_3]^\mathsf{T} \in [-1, 1]^3$, $\omega = [\omega_1, \omega_2, \omega_3]^\mathsf{T} \in \mathbb{R}^3$ は角速度, $\varepsilon\times$ は ε の外積の行列表現である. オイラーパラメーターを用いて回転行列は以下のように計算できる:

$$\begin{aligned} R(\eta, \varepsilon) &= (\eta^2 - ||\varepsilon||^2) I_{3\times 3} + 2\varepsilon\varepsilon^\mathsf{T} + 2\eta\varepsilon\times \\ &= \begin{bmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ r_{31} & r_{32} & r_{33} \end{bmatrix}, \end{aligned} \tag{3.31}$$

ここで $r_{11} = \varepsilon_1^2 - \varepsilon_2^2 - \varepsilon_3^2 + \eta^2$, $r_{12} = 2(\varepsilon_1\varepsilon_2 - \varepsilon_3\eta)$, $r_{13} = 2(\varepsilon_1\varepsilon_3 + \varepsilon_2\eta)$, $r_{21} = 2(\varepsilon_1\varepsilon_2 + \varepsilon_3\eta)$, $r_{22} = -\varepsilon_1^2 + \varepsilon_2^2 - \varepsilon_3^2 + \eta^2$, $r_{23} = 2(\varepsilon_2\varepsilon_3 - \varepsilon_1\eta)$, $r_{31} = 2(\varepsilon_1\varepsilon_3 - \varepsilon_2\eta)$, $r_{32} = 2(\varepsilon_2\varepsilon_3 + \varepsilon_1\eta)$, $r_{33} = -\varepsilon_1^2 - \varepsilon_2^2 + \varepsilon_3^2 + \eta^2$. オイラーパラメーターのダイナミクスおよび回転行列に含まれる非線形項は η, ε, ω の二乗項または双線形項のみである. そのため, 二乗項および双線形項を MICP によって近似することで, ダイナミクスと回転行列を線形制約として表現することができる. この際, ダイナミクスに 12 個, 回転行列に 10 個の二乗項および双線形項が含まれるため, それぞれに対して近似変数が必要になる.

このとき, オイラー法を用いて $SO(3)$ のダイナミクスは以下のように表現できる:

$$\begin{aligned} \begin{bmatrix} \varepsilon \\ \eta \end{bmatrix}^+ &= \begin{bmatrix} \varepsilon \\ \eta \end{bmatrix} + \frac{\Delta T}{2} \begin{bmatrix} w_{\eta\omega_1} + w_{\varepsilon_3\omega_2} - w_{\varepsilon_2\omega_3} \\ -w_{\varepsilon_3\omega_1} + w_{\eta\omega_2} + w_{\varepsilon_1\omega_3} \\ w_{\varepsilon_2\omega_1} - w_{\varepsilon_1\omega_2} + w_{\eta\omega_3} \\ -w_{\varepsilon_1\omega_1} - w_{\varepsilon_2\omega_2} - w_{\varepsilon_3\omega_3} \end{bmatrix} \\ &= \begin{bmatrix} \varepsilon \\ \eta \end{bmatrix} + B_w w, \end{aligned} \tag{3.32}$$

$$\begin{aligned} R_w &= \begin{bmatrix} r_{w11} & r_{w12} & r_{w13} \\ r_{w21} & r_{w22} & r_{w23} \\ r_{w31} & r_{w32} & r_{w33} \end{bmatrix} \\ &= J_{R_w} w, \end{aligned} \tag{3.33}$$

ここで $w = [w_{\eta\omega_1}, w_{\varepsilon_3\omega_2}, \ldots] \in \mathbb{R}^{22}$ は近似変数をまとめたベクトル, B_w, J_w は定数行列, $r_{w11} = w_{\varepsilon_1^2} - w_{\varepsilon_2^2} - w_{\varepsilon_3^2} + w_{\eta^2}$, $r_{w12} = 2(w_{\varepsilon_1\varepsilon_2} - w_{\varepsilon_3\eta})$, $r_{w13} = 2(w_{\varepsilon_1\varepsilon_3} + w_{\varepsilon_2\eta})$, $r_{w21} = 2(w_{\varepsilon_1\varepsilon_2} + w_{\varepsilon_3\eta})$, $r_{w22} = -w_{\varepsilon_1^2} + w_{\varepsilon_2^2} - w_{\varepsilon_3^2} + w_{\eta^2}$, $r_{w23} = 2(w_{\varepsilon_2\varepsilon_3} - w_{\varepsilon_1\eta})$, $r_{w31} = 2(w_{\varepsilon_1\varepsilon_3} - w_{\varepsilon_2\eta})$, $r_{w32} = 2(w_{\varepsilon_2\varepsilon_3} + w_{\varepsilon_1\eta})$,, $r_{w33} = -w_{\varepsilon_1^2} - w_{\varepsilon_2^2} + w_{\varepsilon_3^2} + w_{\eta^2}$.

この際, 拡張オイラー法やルンゲクッタ法といった他の数値積分手法を用いることもできる [40]. 一方で, 近似精度と計算コストはトレードオフの関係にあることを考慮する必要がある.

3.4.2 姿勢を考慮した MLD システム

オイラーパラメーターをピッキングモデルに関する MLD モデル (3.4) に組み込むことを考える. このとき, ロボットシステムの状態を以下のように定義する:

$$X_{\mathrm{a}i} = \begin{bmatrix} x_{\mathrm{arm}i}^{\mathsf{T}} & \dot{x}_{\mathrm{arm}i}^{\mathsf{T}} & \varepsilon_{\mathrm{arm}i}^{\mathsf{T}} & \eta_{\mathrm{arm}i} \end{bmatrix}^{\mathsf{T}}, \tag{3.34}$$

$$X_{\mathrm{o}i} = \begin{bmatrix} x_{\mathrm{obj}i}^{\mathsf{T}} & \dot{x}_{\mathrm{obj}i}^{\mathsf{T}} & \varepsilon_{\mathrm{obj}i}^{\mathsf{T}} & \eta_{\mathrm{obj}i} \end{bmatrix}^{\mathsf{T}}, \tag{3.35}$$

ここでは, オイラーパラメーターに関する入力は角速度とする. ピッキング動作に伴うオイラーパラメーターのダイナミクスは以下のように仮定する:

$$\begin{bmatrix} \varepsilon_{\mathrm{arm}i} \\ \eta_{\mathrm{arm}i} \end{bmatrix}^{+} = \begin{bmatrix} \varepsilon_{\mathrm{arm}i} \\ \eta_{\mathrm{arm}i} \end{bmatrix} + B_w w_{\mathrm{arm}i}, \tag{3.36}$$

$$\begin{bmatrix} \varepsilon_{\mathrm{obj}j} \\ \eta_{\mathrm{obj}j} \end{bmatrix}^{+} = \begin{bmatrix} \varepsilon_{\mathrm{obj}j} \\ \eta_{\mathrm{obj}j} \end{bmatrix} + \varpi_{ij} B_w w_{\mathrm{obj}j}. \tag{3.37}$$

この MLD システムを用いることで, アームとオブジェクトの姿勢を考慮したピッキングモデルを構築することができる. このロボットシステムを用いてピッキングモデルに対する混合整数線形計画問題の定式化の際の離散変数の数は, 計画問題のステップ数 N に対して $(n_{xy} \log_2 N + n_\vartheta + n_\varpi)N$ となる. このとき, n_{xy} は双線形項または二乗項に含まれる状態の種類の数である.

3.4.3 2 次元モデルの姿勢近似

2 次元平面上での姿勢 η_z も同様に, 拡大変数 $\sin(\eta_z)$, $\cos(\eta_z)$ を用いて近似できる. このとき, 姿勢に関するダイナミクスは次式で表現できる:

$$\begin{bmatrix} \eta_z \\ \sin(\eta_z) \\ \cos(\eta_z) \end{bmatrix}^{+} = \begin{bmatrix} \eta_z \\ \sin(\eta_z) \\ \cos(\eta_z) \end{bmatrix} + \Delta T \begin{bmatrix} \omega_3 \\ w_{\omega_3 \cos(\eta_z)} \\ -w_{\omega_3 \sin(\eta_z)} \end{bmatrix}. \tag{3.38}$$

オイラーパラメーターと同様に MLD モデルに組み込むことも可能であり, 2 次元平面上でののアームとオブジェクトの姿勢の制約を考慮したピックアンドプレイスタスクの行動計画を計算することが可能である.

3.5 拡大線形システム, 拡大双線形システムを用いた行動計画

本節では, 拡大線形, 双線形システムに対する行動計画手法を提案する. まず, 拡大線形, 双線形システムに対して MLD システムを定義し, 混合整数計画問題として定式化できることを示す. 次に, 非弾性衝突のような状態のジャンプがあるシステムに対しても, 拡大線形, 双線形システムを用いて MLD システムとして表現できることを示す.

3.5.1 拡大線形システムのための MLD システム

本章では, システムのダイナミクスに複数のモードがある場合, それぞれのモードにおいて同一の拡大状態空間を用いてシステムを表現することが可能であると仮定する. このとき, 拡大線形システムに対する MLD システムは以下のように定義できる:

$$\begin{cases} \xi^+ = A\xi + B_1\mathbf{u} + B_2\mathbf{z} + B_3\varpi + B_4\vartheta + B_5\mathbf{p} \\ C\xi + D_1\mathbf{u} + D_2\mathbf{z} + D_3\varpi + D_4\vartheta + D_5\mathbf{p} \leq E. \end{cases} \tag{3.39}$$

特に, モードの数が 2 つの場合, 各行列は次のようになる; $A = A_{\xi,0}$, $B_1 = B_{\xi_0}$, $B_2 = [A_{\xi_1} - A_{\xi_0}, B_{\xi_1} - B_{\xi_0}]$, $z = \varpi \cdot [\xi^\mathsf{T}, \mathbf{u}^\mathsf{T}]^\mathsf{T}$. ここで $A_{\xi_0}, A_{\xi_1}, B_{\xi_0}, B_{\xi_1}$ は各モードのダイナミクスに関する定数行列である. この MLD システムは前述した MLD システム (3.1) と同等である. そのため, この MLD システムを用いることで, 拡大線形システムに対しても混合整数計画問題による定式化が可能である.

3.5.2 拡大双線形システムのための MLD システム

拡大線形システムと同様に, システムのダイナミクスに複数のモードがある場合, それぞれのモードにおいて同一の拡大状態空間を用いてシステムを表現することが可能であると仮定する. 加えて, ダイナミクスの中に現れる双線形項を混合整数凸計画を用いて近似し, 以下のように新しい変数に置き換える.

$$w_{\xi\mathbf{u}} \approx \begin{bmatrix} \xi u_1 \\ \xi u_2 \\ \vdots \\ \xi u_{n_\mathbf{u}} \end{bmatrix}. \tag{3.40}$$

このとき, 拡大双線形システムに対する MLD システムは以下のように定義できる:

$$\begin{cases} \xi^+ = A\xi + B_1\mathbf{u} + B_2\mathbf{z} + B_3\varpi + B_4\vartheta + B_5\mathbf{p} + B_6 w_{\xi\mathbf{u}} \\ C\xi + D_1\mathbf{u} + D_2\mathbf{z} + D_3\varpi + D_4\vartheta + D_5\mathbf{p} + D_6 w_{\xi\mathbf{u}} \leq E, \end{cases} \tag{3.41}$$

ここで, B_6, D_6 は定数行列である. 特に, モードの数が 2 つの場合, 各行列は次のようになる; $A = A_{\xi,0}$, $B_1 = B_{\xi_0}$, $B_2 = [A_{\xi_1} - A_{\xi_0}, B_{\xi_1} - B_{\xi_0}, H_{\xi_1,1} - H_{\xi_0,1}, H_{\xi_1,2} - H_{\xi_0,2}, \ldots, H_{\xi_1,n_\mathbf{u}} - H_{\xi_0,n_\mathbf{u}}]$, $B_6 = [H_{\xi_0,1}, H_{\xi_0,2}, \ldots, H_{\xi_0,n_\mathbf{u}}]$, $\mathbf{z} = \varpi \cdot [\xi^\mathsf{T}, \mathbf{u}^\mathsf{T}, w_{\xi\mathbf{u}}^\mathsf{T}]^\mathsf{T}$ $(i = 1, 2, \ldots, n_\mathbf{u})$. ここで $A_{\xi_0}, A_{\xi_1}, B_{\xi_0}, B_{\xi_1}$ は各モードのダイナミクスに関する定数行列である. この MLD システムは $w_{\xi\mathbf{u}}$ を z に含めることで, 前述した MLD システム (3.1) と同等となる. そのため, この MLD システムを用いることで, 拡大双線形システムに対しても混合整数計画問題による定式化が可能である.

3.5.3 拡大システム上での状態のジャンプ

機械システムには非弾性衝突のような状態のジャンプをもつものがある. このシステムを拡大システムとして表現する場合, そのジャンプに関する特性も拡大システムに反映する必要がある. ここでは, ジャンプの振る舞いが既知であると仮定し, 以下に示すような MLD システムの離散状態 ϖ が 0 から 1 へ変化した際に発生するジャンプについて考える:

$$\varpi = (0 \nearrow 1) \;\Rightarrow\; \dot{\mathbf{q}}_+ = P(\mathbf{q})\dot{\mathbf{q}}_-. \tag{3.42}$$

ここで $\nearrow$ は ϖ の変化を表わし, $\mathbf{q} \in \mathbb{R}^{n_\mathbf{q}}$ は一般化座標系, $\dot{\mathbf{q}}_-$ は ϖ が 1 になる直前の速度, $\dot{\mathbf{q}}_+$ は ϖ が 1 になったことで状態のジャンプが発生した後の速度, $P(\mathbf{q})$ はジャンプに関する射影行列とする. 以降では, この速度の射影として表現可能なジャンプをもつシステムについて考える.

このとき, 状態のジャンプは拡大状態空間上での状態変数同士の双線形項に対する制約として記述できる.

定理 1. 拡大システム (3.39) または (3.41) に対する状態のジャンプ (3.42) を考える. もし拡大システムの拡大状態が速度の多項式: $\dot{\mathbf{q}}_1^{m_1}\dot{\mathbf{q}}_2^{m_2}\cdots\dot{\mathbf{q}}_{n_\mathbf{q}}^{m_{n_\mathbf{q}}}$ $(m_i \in [0, 1, \ldots, M], \sum_i m_i \leq M)$, とそれらと一般化座標のみを引数とする関数の積のみで構成されているとき, 状態のジャンプは拡大システムの状態の射影として表現できる,

$$\varpi = (0 \nearrow 1) \;\Rightarrow\; \xi_+ = P_\xi(\xi_-)\xi_-. \tag{3.43}$$

ここで, ξ_- は ϖ が 1 になる直前の状態, ξ_+ は ϖ が 1 になり状態のジャンプが発生した後の状態, $P_\xi(\xi_-)$ は射影行列である.

Proof. 一般化座標はジャンプによって変化しないため, 一般化座標を引数とする関数 ($m_1 = m_2 = \cdots = m_{n_q} = 0$ の場合) も変化しない. $\otimes$ を Kronecker 積とし, 速度ベクトル同士の m 回の Kronecker 積をベクトル $\dot{\mathbf{q}}^m \in \mathbb{R}^{(n_q)^m}$, Kronecker 積のベクトルの i 個目の要素を $\dot{\mathbf{q}}_i^m \in \mathbb{R}$ とする. ベクトル $\dot{\mathbf{q}}^m \in \mathbb{R}^{(n_q)^m}$ のジャンプは射影行列 $\mathbf{P}^m$ を用いて, $\dot{\mathbf{q}}_+^m = \mathbf{P}^m(q)\dot{\mathbf{q}}_-^m$ と表現できると仮定する. このとき, 速度ベクトル同士の m 回の Kronecker 積のベクトルの

ジャンプは,

$$\begin{aligned}\dot{\mathbf{q}}_+^{m+1} =&(\dot{q}_+ \otimes \dot{\mathbf{q}}_+^m)\\ =&\begin{bmatrix}\dot{q}_{1+}\dot{\mathbf{q}}_+^m\\ \dot{q}_{2+}\dot{\mathbf{q}}_+^m\\ \vdots\\ \dot{q}_{n_q+}\dot{\mathbf{q}}_+^m\end{bmatrix}\\ =&\begin{bmatrix}Q_1 P\dot{q}_-\mathbf{P}^m\dot{\mathbf{q}}_-^m\\ Q_2 P\dot{q}_-\mathbf{P}^m\dot{\mathbf{q}}_-^m\\ \vdots\\ Q_{n_q} P\dot{q}_-\mathbf{P}^m\dot{\mathbf{q}}_-^m\end{bmatrix}\\ =&\begin{bmatrix}\mathbf{P}_1^{m+1}\dot{\mathbf{q}}_-^{m+1}\\ \mathbf{P}_2^{m+1}\dot{\mathbf{q}}_-^{m+1}\\ \vdots\\ \mathbf{P}_{n_q}^{m+1}\dot{\mathbf{q}}_-^{m+1}\end{bmatrix}\\ =&\mathbf{P}^{m+1}\dot{\mathbf{q}}_-^{m+1},\end{aligned}$$

ここで $\mathbf{P}_i^{m+1}(q)(i=1,2,\ldots,n_q)$, $\mathbf{P}^{m+1}(q)$ は射影行列, Q_i は i 番目の要素が 1, それ以外が 0 の行ベクトルとする. そのため, $\dot{\mathbf{q}}^{m+1}$ のジャンプは $\dot{\mathbf{q}}^{m+1}$ の射影となっている. $\dot{\mathbf{q}}$ の場合と速度ベクトル同士の m 回の Kronecker 積のベクトルの場合に射影であることが示されたため, 数学的帰納法により, 速度ベクトル同士の m 回の Kronecker 積のベクトルのジャンプ挙動は, あらゆる自然数 m に対して, $\dot{\mathbf{q}}^m$ の射影であることが示された.

さらに, 速度の多項式と一般化座標を引数とする関数との積のジャンプは次式で記述できる:

$$\begin{aligned}(\dot{\mathbf{q}}_i^m g(q))_+ =&\dot{\mathbf{q}}_{i+}^m g(q)\\ =&Q_i\mathbf{P}^m(q)\dot{\mathbf{q}}_-^m g(q).\end{aligned}$$

このとき, ベクトル $\dot{\mathbf{q}}_i^{m+1}g(q)$ の要素のジャンプは $\dot{\mathbf{q}}_i^{m+1}g(q)$ の射影である. そのため, もし拡大状態が速度の多項式とそれらと一般化座標のみを引数とする関数の積のみで構成されているとき, 状態のジャンプは拡大システムの状態の射影として表現できる. □

定理 1 に従って, 拡大状態を決定することで式 (3.43) で表現されるジャンプを線形または双線形な拘束条件として近似できる.

3.6 $SO(3)$ に対する数値実験

本節では, MICP を用いた提案手法を 2 つのタスクに適用し, 有効性を確認する. まず, 3 次元空間上でのオブジェクトの姿勢を考慮したピックアンドプレイスタスクに提案手法を適用する. 次に, 劣駆動システムであるドローンモデルの到達タスクに対して, 提案手法を適用

する. 本節の全ての計算は Intel Core i7 1.8GHz で行われ, 全ての混合整数線形計画問題は MATLAB によって実装され, ソルバーとして Gurobi [41] を用いた. また, 混合整数計画問題の最適性ギャップの上限を 1.0×10^{-4} と設定した.

3.6.1 3 次元のピックアンドプレイスタスク

1 つのアーム, 1 つのオブジェクト, および 1 つのゴールであるボックス領域で構成される 3 次元空間でのピックアンドプレイスタスクについて考える. オブジェクトに対して入力はアームから与えられ, コスト関数は次のように定義する.

$$J = \sum_{k \in \mathcal{T}} ||\mathbf{u}(t)||_1, \tag{3.44}$$

ここで $\mathbf{u} \in \mathbb{R}^3$ は並進方向の力, $||\mathbf{u}||_1$ は l_1-ノルムとする. $SO(3)$ への入力は角速度 $\omega \in [-\pi/2, \pi/2]^3$ とし, アームの並進運動に関するダイナミクスは 2 重積分器のシステムと仮定する. また, 問題を単純化するため, アームの姿勢は考慮せず, オブジェクトの姿勢のみを考慮することとする. システムのダイナミクスはオイラー法を用いて実装した.

本実験では, 直方体状のオブジェクトを考え, オブジェクトの各頂点の座標は次のように計算する:

$$\mathcal{V}_{\mathrm{obj}i} = R_{w\mathrm{obj}} L_{\mathrm{obj}i} + x_{\mathrm{obj}} \ (i = 1, \ldots, 8), \tag{3.45}$$

ここで $R_{w\mathrm{obj}}$ はオブジェクトの姿勢を示す回転行列, $L_{\mathrm{obj}i}$ は x_{obj} に対する頂点 i の相対位置とする. また, 命題 “オブジェクトがボックス内に収まっている” を考え, この命題はオブジェクトの全ての頂点がボックス領域内に収まっているとき満たされるとする. 本実験ではオブジェクトのある 1 面に対してピッキング動作に関するポリトープを設定した (Fig.3.3 中, “picking polytope”).

ピックアンドプレイスタスクの制約条件に関して以下の LTL 命題を定義する:

- アームはボックス領域以外でオブジェクトを離さない:

$$(\varpi_{11} \wedge \bigcirc \neg \varpi_{11}) \ \Rightarrow \ \varphi_{\theta 1}, \tag{3.46}$$

  ここで ϖ_{11} はアームがオブジェクトを掴むことを示す論理入力, $\varphi_{\theta 1}$ はオブジェクト 1 がボックス内に収まっていることを示す命題である.

このタスクに対して, オブジェクトの姿勢と位置が異なる 20 のケースにおいて命題を達成する軌道を提案手法を用いて生成した. 近似回転行列 R_w の近似精度を向上するために, 全てのケースで, オイラーパラメーターの初期状態は $[0, 0, 0, 1]$ とし, オブジェクトの初期姿勢に合わせて $L_{\mathrm{obj}i}$ を変更した. 時間ステップは $\Delta T = 0.3$, $N = 8$ と設定した. このときの混合整数線形計画問題は 207 個の離散変数を含み, 87 個の変数は LTL 命題に関係し, 残りの 120 個は $SO(3)$ の双線形項に関係している.

結果として, 実行可能解は平均 7.63 秒, 標準偏差 ±9.00 秒以内に計算された. 最適解に関しては, 計算したケースの内, 1 つのケースのみ最適解が 6000 秒以内に収束せず, 他のケースにおいては平均 $1.93 \times 10^{+3}$ 秒, 標準偏差 $\pm 1.79 \times 10^{+3}$ 秒以内に準最適な軌道に収束した. Fig.3.4 は, 2 つのケースと得られた準最適な軌道を示している.

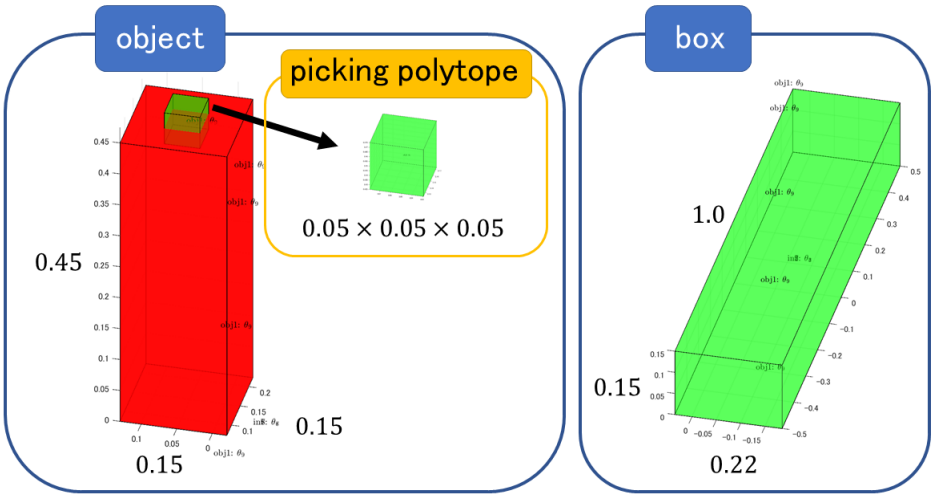

Fig. 3.3: Setting of the object and box. x_{obj1} exists at the center of the picking polytope. The arm needs to lay the object down to put it in the box.

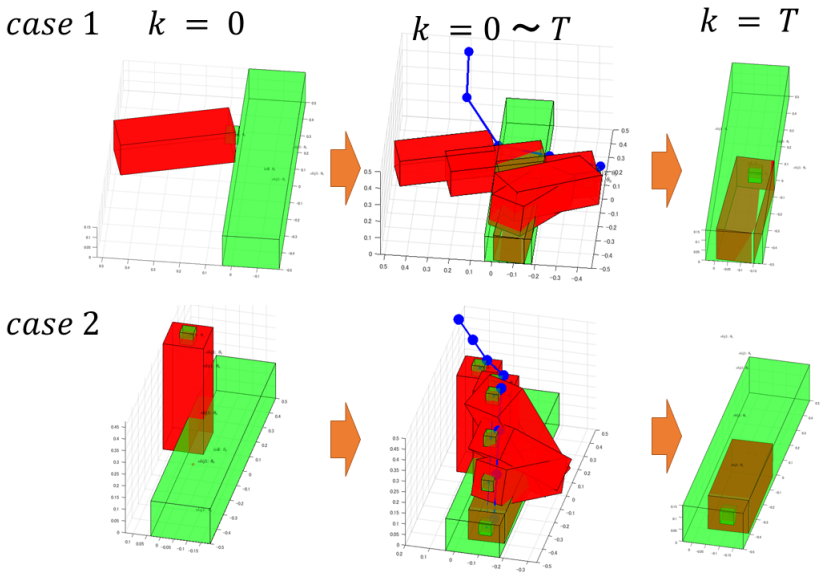

Fig. 3.4: Behavior in semi-optimal solutions. Upper: a situation in which the object is lying down. Lower: a situation in which the object is upright. The blue line with circles is the trajectory of the arm. In each situation, the arm completes the pick-and-place task by laying the object down.

3.6.2 ドローンモデルにおけるプランニング

下記のドローンモデルを考える.

$$\frac{d}{dt}\begin{bmatrix}\varepsilon_{\mathrm{d}}\\ \eta_{\mathrm{d}}\end{bmatrix} = \frac{1}{2}\begin{bmatrix}\eta_{\mathrm{d}} I_{3\times 3} - \varepsilon_{\mathrm{d}}\times\\ -\varepsilon_{\mathrm{d}}{}^{\mathsf{T}}\end{bmatrix}\omega, \tag{3.47}$$

$$\begin{aligned}\frac{d^2}{dt^2}\begin{bmatrix}x_{\mathrm{d}}\\ y_{\mathrm{d}}\\ z_{\mathrm{d}}\end{bmatrix} &= \frac{1}{m}R(\eta,\varepsilon)\begin{bmatrix}0\\0\\f_z\end{bmatrix} + \begin{bmatrix}0\\0\\g\end{bmatrix}\\ &= \frac{1}{m}\begin{bmatrix}r_{13}f_z\\ r_{23}f_z\\ r_{33}f_z\end{bmatrix} + \begin{bmatrix}0\\0\\g\end{bmatrix},\end{aligned} \tag{3.48}$$

ここで f_z は機体座標系の z 軸方向の力, $X_{\mathrm{d}} = [x_{\mathrm{d}}, y_{\mathrm{d}}, z_{\mathrm{d}}, \dot{x}_{\mathrm{d}}, \dot{y}_{\mathrm{d}}, \dot{z}_{\mathrm{d}}, \varepsilon_{\mathrm{d}}^{\mathsf{T}}, \eta_{\mathrm{d}}]^{\mathsf{T}} \in \mathbb{R}^{10}$ はドローンの状態, $[\omega^{\mathsf{T}}, f_z] \in \mathbb{R}^4$ をドローンモデルへの入力とする.

このモデルの計画問題は, $SO(3)$ に加えて $w_{r_{13}f_z} \approx r_{13}f_z$, $w_{r_{23}f_z} \approx r_{23}f_z$, $w_{r_{33}f_z} \approx r_{33}f_z$ の項を MICP を用いて近似することで, ダイナミクス (3.48) を線形制約として表現できる. 加えて, ドローンの幾何的な特性として機体は厚みのない長方形と見なし, $[x_{\mathrm{d}}, y_{\mathrm{d}}, z_{\mathrm{d}}]$ をドローンの機体の中心の座標とする. 近似したダイナミクス (3.47)(3.48) に対して, 積分手法は拡張オイラー法 [40] を用いた. コスト関数は以下のように設定した:

$$J = \sum_{k\in\mathcal{T}} |f_z(k)|. \tag{3.49}$$

ドローンがスリットを通ってゴール領域に到達するタスクを考える. Fig.3.5 に, ドローンがタスクを行う環境を示す. ドローンは, スタート領域からスリットを通ってゴール領域に到達する必要があり, スタート, スリット, ゴールの 3 つの領域に対して, LTL 命題 $\varphi_{\mathrm{exist},0}$, $\varphi_{\mathrm{exist},s}$, $\varphi_{\mathrm{exist},g}$ を考える. 各命題はドローンが各エリアに存在する場合に満たされるとする. また, ドローンは機体の各頂点が領域から外れないように移動することとする. このとき, 到達タスクの LTL 命題を次のように定義する.

- ドローンはゴール領域に到達する:

$$\mathbf{F}\,\varphi_{\mathrm{exist},g} \tag{3.50}$$

- ドローンはスリットを通る:

$$\varphi_{\mathrm{exist},0} \Rightarrow \bigcirc\neg\varphi_{\mathrm{exist},g} \tag{3.51}$$

- ドローンの機体が領域外に出ない:

$$\mathbf{G}\,(\varphi_{\mathrm{exist},0} \vee \varphi_{\mathrm{exist},s} \vee \varphi_{\mathrm{exist},g}) \tag{3.52}$$

各パラメーターは $\Delta T = 0.3$, $T = 6$, $m = 1$, $g = -1$, $f_z \in [0, 2]$, および $\omega \in [-\pi/2, \pi/2]^3$ とする. このときの混合整数線形計画問題には 259 個の離散変数を含み, 105 個の変数は LTL 命題に関係し, 残りの 154 個の変数は $SO(3)$ とダイナミクスの双線形項に関係している. ドローンの初期姿勢と位置を変更し, 10 個のケースに対して計画問題を計算した.

このとき, 実行可能解は平均 $28.2s$, 標準偏差 ± 22.7 秒で計算された. 最適解は平均 $6000s$ 以内に収束しなかったが, MIP 最適性ギャップの平均 $4.93 \times 10^{-1}\%$, 標準偏差 $\pm 2.08 \times 10^{-1}\%$ の準最適な解が得られた. そのため, 提案手法を他の最適化手法と組み合わせ, 提案手法を実現可能性チェックまたは初期解として使用することにより, より短時間で (準) 最適解が得られることが推測できる. Fig.3.5 は, 準最適な解の軌道を示している

シミュレーターでの挙動との比較

混合整数線形計画問題を解くことによって得られたシステムへの入力を, 連続時間システム (3.47)(3.48) のシミュレーターに開ループ方式で適用する. ドローンモデルの連続時間システムのシミュレーションには, Simulink を使用して実装した. Fig.3.6 は, シミュレーションに適用した入力を示している. 混合整数線形計画問題を解くことによって取得した入力に対して 1 次ホールドを適用し, シミュレーターへの入力を生成した. Fig.3.6 は, 上記の入力に対応するドローンの動作を示している. シミュレーターによって得られたドローンの軌道は, 近似誤差のためにタスクを完了できていない. その原因の一つとして, 計画時に用いた近似モデルの近似精度が挙げられる. Fig.3.7-3.9 に, 混合整数線形計画問題の解として得られた軌道とシミュレーションによって得られた軌道の比較を示す.

また, Fig.3.10 に, 混合整数線形計画問題の解であるオイラーパラメーターの l_2-ノルムと, 混合整数線形計画問題の解とシミュレーターでの回転行列の差分 $\Delta R = R_{milp}^{\mathsf{T}} R_{sim}$ をオイラーパラメーターで表現した際の回転量 $\Delta\theta_{\mathbf{a}}$, $\Delta\bar{\theta}_{\mathbf{a}}$ の軌道を示す. $\Delta\theta_{\mathbf{a}}$ は混合整数線形計画問題の解として得られた双線形項を近似する変数から計算された回転行列との差分に対応し, $\Delta\bar{\theta}_{\mathbf{a}}$ は混合整数線形計画問題の解であるオイラーパラメーターを $SO(3)$ に射影した値を用いて式 (3.31) より計算した回転行列との差分に対応している. $\Delta\theta_{\mathbf{a}}$, の軌道から, 回転行列については長期的な近似が難しく, これは多くの双線形の近似項を含むためであると考えられる. 一方で, $\Delta\bar{\theta}_{\mathbf{a}}$ の軌道から, 混合整数線形計画問題を行う際に $SO(3)$ への射影の構造を組み込むことができれば, より近似精度の高い計画を行うことができると考えられる. また, 目標状態をより精度よく追従するには, フィードバック制御器等の機構を導入する必要が考えられる.

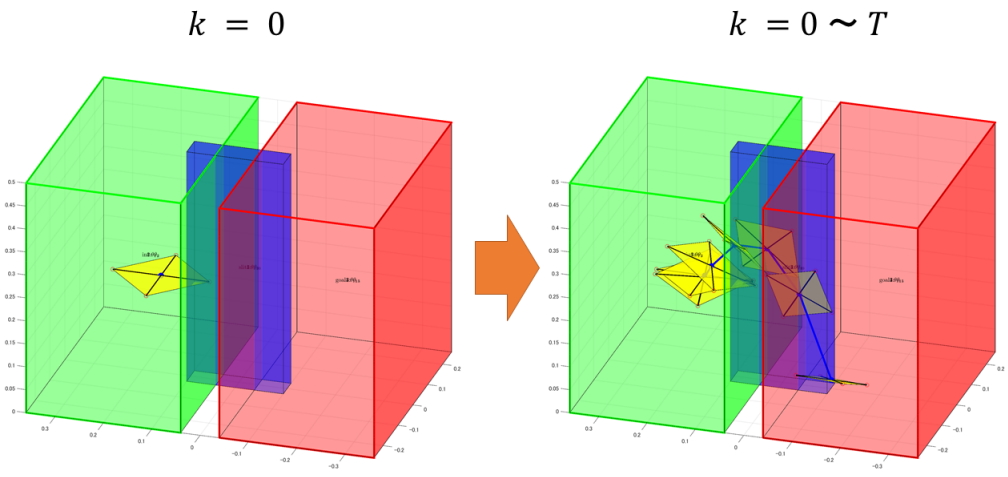

Fig. 3.5: Overview of reaching task of the drone. Left: initial states and workspace. The yellow rectangle represents initial states of the drone. The green, blue and red areas are the initial, slit, and goal areas, respectively. Right: the trajectory of a semi-optimal solution. The drone passes through the slit area with the body in a vertical position.

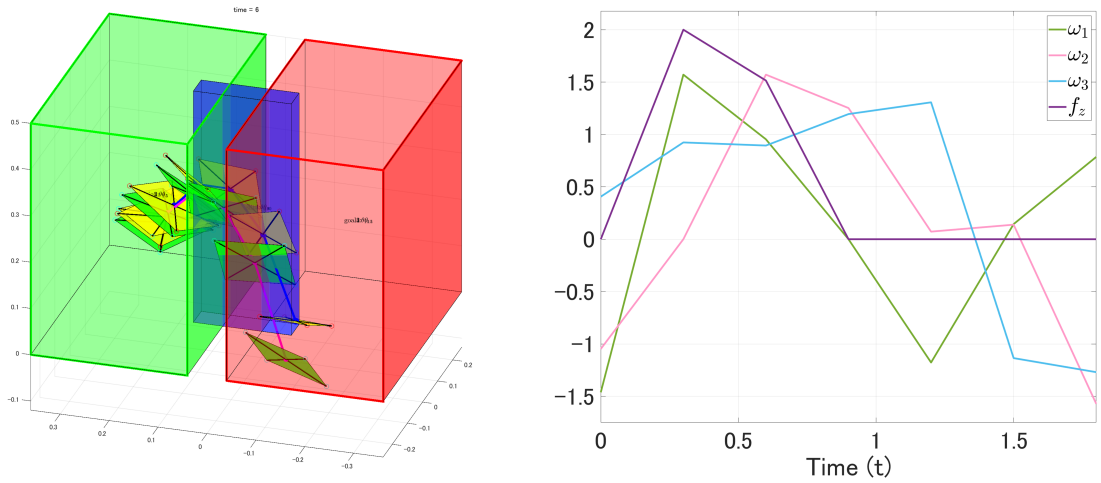

Fig. 3.6: Left: Comparison of the behavior of a solution and simulator. The green rectangle represents the simulation of the drone to which inputs obtained by the Mixed-Integer Linear Problem planning are applied. Right: Input sequences used in the simulator. We applied the First-order hold to the input obtained by the Mixed-Integer Linear Problem planning.

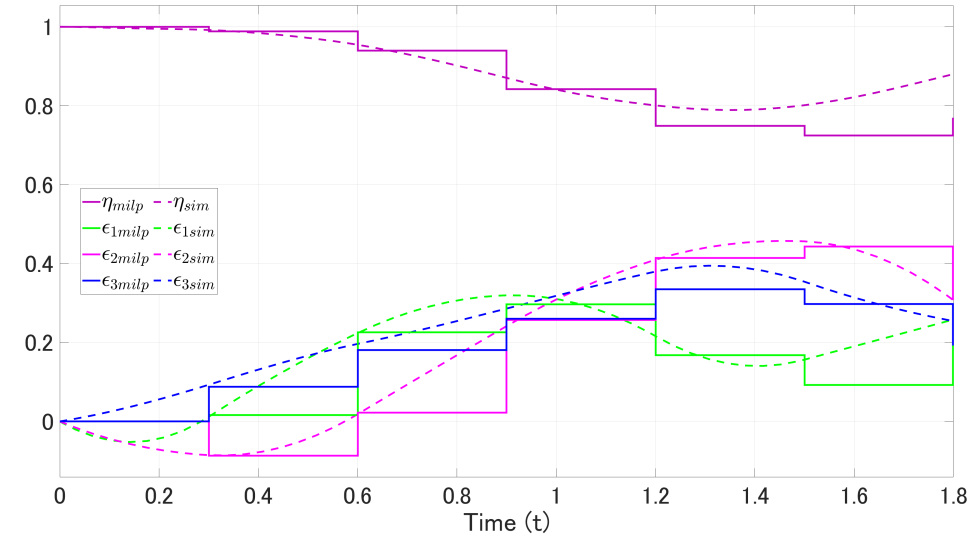

Fig. 3.7: Comparison of the Euler parameter states of the solution and simulator. Solid: the solution. Dashed: simulator.

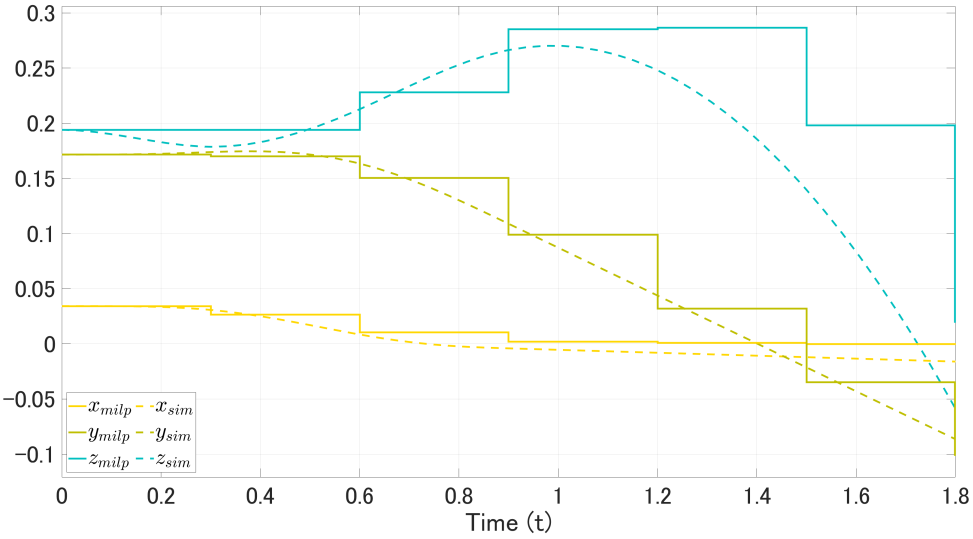

Fig. 3.8: Comparison of position states of the solution and simulator. Solid: the solution. Dashed: simulator.

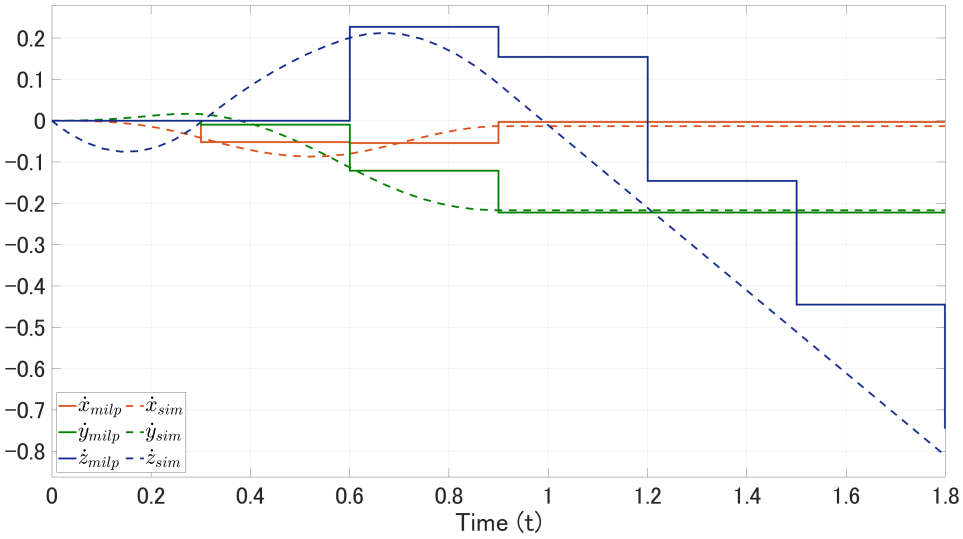

Fig. 3.9: Comparison of velocity states of the solution and simulator. Solid: the solution. Dashed: simulator.

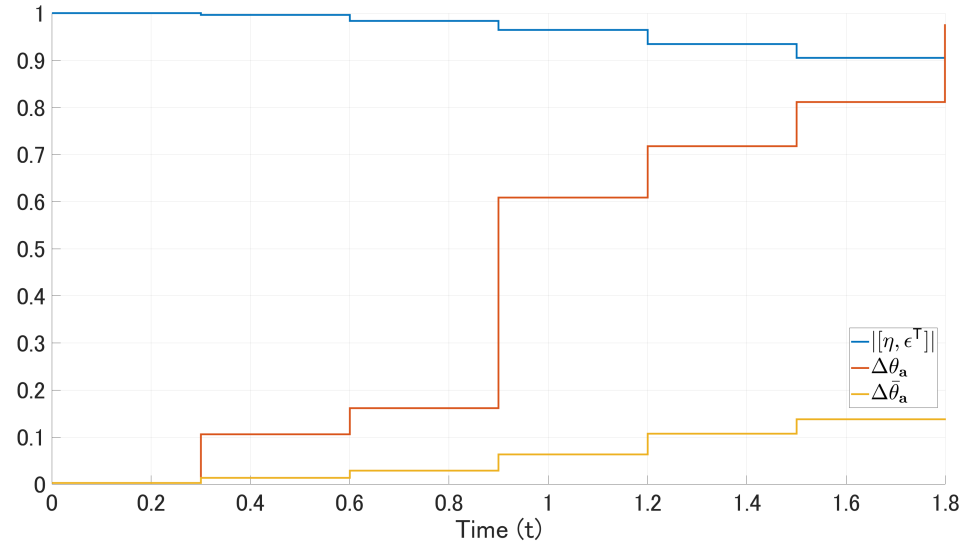

Fig. 3.10: The trajectory of l_2-norm of the Euler parameter states of simulation and rotation angles of difference of rotation matrics: $\Delta\theta_{\mathbf{a}}$ and $\Delta\bar{\theta}_{\mathbf{a}}$

3.7 拡大システムに対する数値実験

本節では, 拡大システムに対して提案手法を適用し, 有効性を確認する. 平面移動ロボットシステムに対して, 拡大線形システムを構築し, 与えられた LTL 命題を満たす軌道を混合整数線形計画問題を解くことによって生成する. 本節での計算は, 3.6GHz Intel Core i9 で実行した. 加えて, 全ての混合整数線形計画問題は MATLAB によって実装され, ソルバーとして Gurobi [41] を用いた. また, 全ての混合整数計画問題の最適性ギャップの上限を 1.0×10^{-4} と設定した.

3.7.1 2 次元平面上でのピックアンドプレイスタスク

2 次元平面上でのピックアンドプレイスタスクを考える. ロボットシステムとして, 以下のシステムのダイナミクスを考える:

$$\frac{d^2}{dt^2}\begin{bmatrix} x \\ y \\ \theta \end{bmatrix} = \begin{bmatrix} u_1 \cos\theta \\ u_1 \sin\theta \\ u_2 \end{bmatrix}. \tag{3.53}$$

ここで $u_1 \in [-1, 1]$ は並進方向の入力加速度, $u_2 \in [-\frac{1}{2}\pi, \frac{1}{2}\pi]$ は回転方向の入力加速度, x, y はロボットの位置, θ はロボットの進行方向とする. ロボットの初期状態は $[x, y, \theta]^\mathsf{T} = [0, 0, 0]^\mathsf{T}$ とする. このシステムは初期状態のまわりでテイラー展開した線形システムを考えた場合, 状態 y が制御不可能な状態になる. そのため, ここでは拡大システムの状態を次のようにとる: $\xi = [x, y, \theta, \dot{x}, \dot{y}, \dot{\theta}, \sin\theta, \cos\theta, \dot{\theta}\sin\theta, \dot{\theta}\cos\theta]^\mathsf{T} \in \mathbb{R}^{10}$. このとき, 離散時間拡大双線形システムは次式で表わされる:

$$\begin{aligned} \xi^+ &= \xi + \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \\ O_{3,1} \\ \dot{\theta}\cos\theta \\ -\dot{\theta}\sin\theta \\ O_{2,1} \end{bmatrix} \Delta_t + \begin{bmatrix} O_{5,1} \\ u_2 \\ O_{4,1} \end{bmatrix} \Delta_t + \begin{bmatrix} O_{3,1} \\ u_1\cos\theta \\ u_1\sin\theta \\ O_{3,1} \\ u_2\sin\theta \\ u_2\cos\theta \end{bmatrix} \Delta_t \\ &= A_\xi \xi + B_\xi \mathbf{u} + \sum_{i=1}^{2} H_{\xi,i}\xi u_i. \end{aligned} \tag{3.54}$$

ここで $\Delta_t = 0.1$ はサンプル時間とする.

ピックアンドプレイスタスクの LTL 命題は, 前述した数値実験と同様に定義する. ここでは, “いつかオブジェクトが目的領域に存在する”という命題を考える. 物体の状態は, 並進位置と速度を考え, ピッキング動作は, ダイナミクスのモードの変化として表現する. ここでは, ピッキング動作の際に状態のジャンプは発生しないと考える. 加えて, オブジェクトをピッキングした場合, 物体の速度は移動ロボットと同期するとする. また, 行動計画を行うステップ数を 30 とする.

開ループ系での評価

まず, 提案手法によって得られた入力を開ループ系に適用し, タスクを達成できるか評価した. 15 パターンの物体の位置を考え, それぞれの位置について行動計画を行う. 15 個の位置のパターンを領域 $[-0.1, 0.1] \times [-0.1, 0.1]$ 上に格子状に設定する. 混合整数凸計画の双線形項に現れる変数の近似に用いる分割数は, 4 と 8 に設定した.

行動計画問題を混合整数計画問題として解いた結果, 4 分割の場合は, 平均 1.31 秒, 標準偏差 ±0.88 秒で実行可能解が得られた. 8 分割の場合は, 平均 11.30 秒, 標準偏差 ±9.39 秒で実現可能解が得られた. 4 分割, 8 分割ともに, 実現可能な解は得られたが, ほとんどの位置のパターンで 200 秒以内に最適解は得られなかった.

Fig.3.11 左に, 計画によって得られた軌道と開ループ系の x と y の軌道の平均二乗誤差 (RMSE) を示す. このとき, 計画によって得られた軌道は最適な軌道ではないため, オブジェクトの位置が同じでも 4 分割と 8 分割の場合で軌道が大きく異なる場合がある. Fig.3.11 から, 8 分割では計画によって得られた軌道と開ループ系での軌道の差が小さくなっていることがわかる. この結果は, 分割数を多くすることで混合整数凸計画の精度が向上するためであると考えられる. Fig.3.11 右に計画によって得られた軌道と開ループ系の軌道を示す. 計画によって得られた軌道は, 8 分割の場合に得られた軌道である. Fig.3.11 で示された軌道と同様に, ほぼ全てのオブジェクトの位置で, 計画によって得られた軌道がタスクを完了させているにもかかわらず開ループ系の軌道はタスクを完了することができなかった.

そこで, 計画によって得られた軌道を参照軌道とみなして, フィードバック制御を行った.

閉ループ系での評価

閉ループ系に対して, 4 分割と 8 分割で行動計画した軌道を参照軌道として用いて制御した際の軌道について評価する. 参照軌道に追従するためのフィードバック制御としてモデル予測制御 (MPC) を用いた.

MPC を拡大双線形系を用いた非線形問題として実装する. この際, MPC の目的関数は, x, y, θ の参照軌道との誤差の重み付け l_2-norm である. そして, 予測ステップ数を 10, MPC におけるサンプリング時間を 0.02, コントローラのサンプリング時間を 0.02 とする. 計画によって得られた軌道 x, y, θ の軌道を参照軌道とする. この非線形問題を CasADi [42] で実装し, ソルバーとして Ipopt を用いた.

Fig.3.11 左に, 計画によって得られた軌道と閉ループ系の x と y の軌道間の RMSE を示す. 開ループ系と比較して軌道間の誤差が減少しており, 特に 8 分割の軌道に対しては追従性が高いことが確認できる. このことから, 提案手法は拡大システムを用いることで, 本来のシステムが実行可能な軌道を生成できていると考えられる. しかし, 場合によっては Fig.3.11 右のように, 軌道追従用の制御器のみではシステムがタスクを達成できない場合がある. しかし, より保守的な計画手法や LTL の情報を利用した制御器を用いることで, より効果的な結果が得られる可能性が考えられる.

保守的な計画手法として, 計画時にオブジェクトや目的領域を実際より小さく設定することが挙げられる. また, 入力の上限といったシステムのダイナミクスの保守性も考慮する必要がある. また, 先行研究 [43] で提案されているような LTL の情報を活用した制御手法と組み合わせることで, よりタスクを完遂しやすくなることも期待できる.

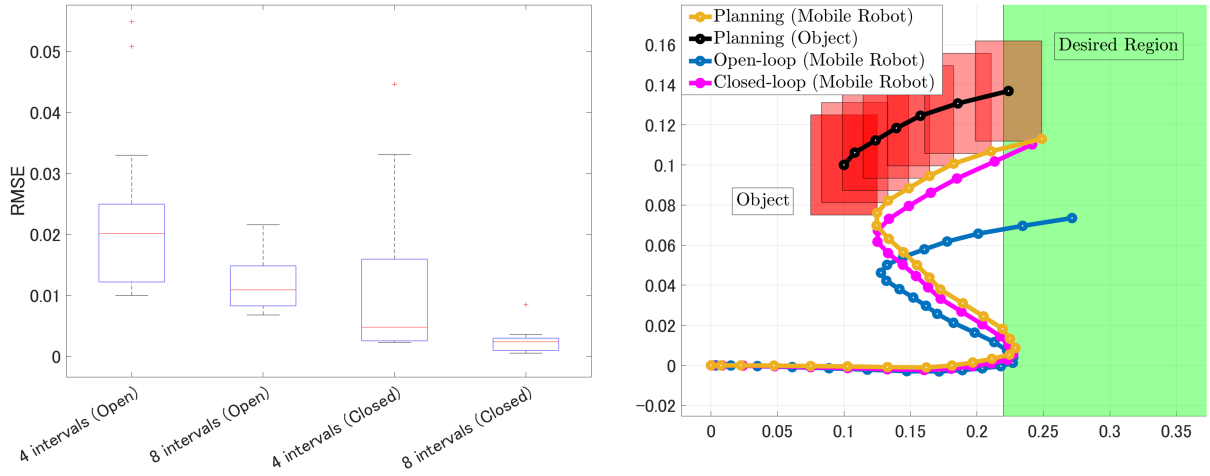

Fig. 3.11: Left: RMSE between the trajectories of x and y of the planning, the open-loop system, and the closed-loop system. Right: The trajectories of planning, open-loop, and closed-loop systems. The yellow trajectory is a trajectory of the planning obtained in a case of 8 intervals. The blue trajectory is a trajectory of the open-loop system. The pink trajectory is a trajectory of the closed-loop system. In this case, we see that the trajectories of both open-loop and closed-loop systems could not complete the task.

3.8 まとめ

本章では, 非線形システムに対して混合整数計画問題による定式化を行うための, MICP や拡大線形化を用いた近似システムを検討した. MICP を用いることで $SO(3)$ に代表されるダイナミクスに双線形項や二乗項を含むシステムを線形システムとして表現でき, 混合整数線形計画問題による定式化が適用できることを示した. また, 拡大線形システムを用いることで, より多くの非線形システムを線形システムとして表現できることや, MICP と組み合わせることで拡大双線形システムに対しても混合整数線形計画問題による定式化が適用できることを示した. その後, MICP, 拡大双線形システムを用いた近似システムに対して混合整数線形制約問題を解くことで, LTL 命題で表現されたタスクを達成する軌道が生成されることを数値シミュレーションによって検証した.

本章で提案した定式化を用いることで, 計算コストを抑えながら抽象化システムとして非線形システムを採用することができる. これにより, より実際のロボットシステムの特性を反映したシステムを抽象化したシステムとして用いることで, 高レベル制御器にて実用的な

軌道を生成することが可能となる．一方で, 各数値実験での開ループ系での評価でも明らかなように, 近似システムと実際のシステムでの応答には差異がある．抽象化システムに関して MICP の近似精度を高めることや, 拡大線形システムの状態の選定方法によって, 近似精度の高いシステムを構築することが可能である．一方で, MICP の近似精度の向上は計算コストの増加を伴うといった問題があり, タスクごとにどのような近似システムを用いるべきかの判断基準は今後の課題である．

第4章　不確かな外的事象に関する命題への軌道最適化

4.1　はじめに

前章では, ロボットの状態またはロボットの振る舞いを模した離散入力に依存する時相論理命題に関するタスクを対象とした. 一方で, 実際のタスクには人間からの指示を受けて行動するといった場面が存在する. この際, 人間からの指示や信号はロボットシステムの状態に依存せず発生する外的事象として考えられる.

これをふまえ, 本章では発生時刻が不確かな外的事象に関する命題を達成するための軌道最適化手法を提案する. まず, 外的事象に関する命題を記述可能な時相論理である Event-based STL について説明する. その後, 外的事象の発生時刻に関する分布が既知であると仮定し, その分布に基づき, 複数の発生時刻を想定して与えられた命題を達成する軌道を生成することが可能な軌道最適化を非線形計画問題として定式化する. 最後に, 不確かな外的事象に関する命題が与えられた移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

4.2　Event-based STL

Event-based STL は STL を拡張した時相論理であり, 解的事象に関する命題を表現することが可能である. Event-based STL の命題 Ψ は下記の様に定義される [44]:

$$
\begin{aligned}
\varphi ::=& \pi \mid \neg\pi \mid \varphi_1 \wedge \varphi_2 \\
\alpha ::=& \nu \mid \neg\alpha \mid \alpha_1 \wedge \alpha_2 \\
\Psi ::=& \mathbf{G}_I\ \varphi \mid \mathbf{F}_I\ \varphi \mid \varphi_1\ \mathbf{U}_I\ \varphi_2 \mid \\
& \mathbf{G}(\alpha \Rightarrow\ \Psi) \mid \mathbf{G}(\varphi \Rightarrow\ \Psi) \mid \Psi_1 \wedge \Psi_2
\end{aligned}
$$

ここで α は環境命題 $\nu \in \mathsf{AP}$ に関する論理命題である. ν の充足関係は外的事象の発生やアラームやスイッチングといった論理入力などによって決定される. そのため, Event-based STL で表現された命題をシステムの状態に対して評価する場合, システムの状態として離散

変数を含む必要がある. Event-based STL の意味論は, STL と同様に下記の様に行われる:

$$\begin{aligned}
(\mathbf{x},\, t) &\models \neg\alpha && \Leftrightarrow \neg((\mathbf{x},\, t) \models \alpha)\\
(\mathbf{x},\, t) &\models \alpha_1 \wedge \alpha_2 && \Leftrightarrow (\mathbf{x},\, t) \models \alpha_1 \ \wedge\ (\mathbf{x},\, t) \models \alpha_2\\
(\mathbf{x},\, t) &\models \mathbf{G}(\alpha \Rightarrow \Psi) && \Leftrightarrow \forall t, (\mathbf{x},\, t) \models \neg\alpha \ \vee\ (\mathbf{x},\, t) \models \Psi\\
(\mathbf{x},\, t) &\models \Psi_1 \wedge \Psi_2 && \Leftrightarrow (\mathbf{x},\, t) \models \Psi_1 \ \wedge\ (\mathbf{x},\, t) \models \Psi_2.
\end{aligned}$$

本章では, 非線形計画問題としての定式化を行うため, 離散変数を用いない外的事象に関する命題のロバストネス関数が必要である. そのため, 本章では Event-based STL 命題に対するなめらかなロバストネス関数を定義し, 付加関数として扱う.

4.3 Event-based STL のロバストネス関数

もし環境命題 ν が複数の外的事象やシステムの離散状態に依存する場合, ロバストネス関数の定義域が分離された集合になる. 本章では計算の複雑さを減らすために, 次の仮定を考える:

仮定 1. 外的事象に関する環境命題 ν の真偽はただ一つの外的事象に依存し, 事象の発生は連続変数の値によって記述できる.

このとき, 連続な補助変数を用いたなめからなロバストネス関数を以下のように定義する:

$$\rho_{\nu_i}\left(\mathbf{x}, \mathbf{x}_\nu, t\right) = \rho_{\nu_i}\left(x_{\nu_i}(t)\right), \tag{4.1}$$

ここで, $\mathbf{x}_\nu = [x_{\nu_1}, \ldots, x_{\nu_{n_\nu}}]^\mathsf{T} \in \mathbb{R}^{n_\nu}$, $x_{\nu_i} \in \mathcal{X}_{\nu_i} = [x_{\nu_i,\min}, x_{\nu_i,\max}] \in \mathbb{R}$, $(i = 1, \ldots, n_\nu)$ は連続な補助変数, $\rho_{\nu_i} : \mathcal{X}_{\nu_i} \to \mathbb{R}$ はなめらかな関数, $x_{\nu_i,\min}$ と $x_{\nu_i,\max}$ は定数とする. 加えて, 本論文では以下を仮定する.

仮定 2. 関数 $\rho_\nu(x_\nu)$ は $\mathcal{X}_\nu$ で有界な関数であり, $|\rho_\nu(x_\nu)| < \epsilon_\nu$, $x_\nu \in \mathcal{X}_\nu$ を満たす定数 $\epsilon_\nu \in \mathbb{R}$ が存在する.

Example 2. 関数 $\rho_\nu(x_\nu) = x_\nu$, $\mathcal{X}_\nu = [-0.1, 0.1]$ を考える. このとき, $x_\nu \geq 0$ ならば, $\rho_\nu(x_\nu) \geq 0$ かつ ν が発生してる. 一方で, $x_\nu < 0$ ならば, $\rho_\nu(x_\nu) < 0$ かつ ν は発生していない. 加えて, $\epsilon_\nu = 0.1$ は $|\rho_\nu(x_\nu)| \leq \epsilon_\nu$ を満たす.

このとき, Event-based STL の命題に対するロバストネス関数を以下のように定義できる.

$$\rho_{\neg\alpha}(\mathbf{x}, \mathbf{x}_\nu, t) := -\,\rho_\alpha(\mathbf{x}, \mathbf{x}_\nu, t), \tag{4.2}$$

$$\rho_{\alpha_1\wedge\alpha_2}(\mathbf{x}, \mathbf{x}_\nu, t) := \min(\rho_{\alpha_1}(\mathbf{x}, \mathbf{x}_\nu, t), \rho_{\alpha_2}(\mathbf{x}, \mathbf{x}_\nu, t)), \tag{4.3}$$

$$\rho_{\mathbf{G}(\alpha\Rightarrow\Psi)}\left(\mathbf{x}, \mathbf{x}_\nu, t\right) := \min_{t'\in[t,T]} \left\{\max(\rho_{\neg\alpha}(\mathbf{x}, \mathbf{x}_\nu, t'), \rho_\Psi(\mathbf{x}, \mathbf{x}_\nu, t'))\right\}. \tag{4.4}$$

4.4 不確かさを持つ外的事象に対する Event-based STL 命題のための行動計画手法

Event-based STL の命題 Ψ に対する軌道最適化問題を次の非線形最適化問題として定式化する:

$$\begin{aligned} \min_{Z} \ & -\rho_\Psi(\mathbf{z}) + \sum_k \lambda_k J_k(\mathbf{x}_k, \mathbf{u}_k) \\ \text{s.t. Eq.(2.2)}, \quad & \rho_\Psi(\mathbf{z}) \geq \epsilon_\Psi. \end{aligned} \tag{4.5}$$

Z は最適化変数で構成されたベクトル(システムの状態, 入力, 補助変数 $\mathbf{x}_0, \ldots, \mathbf{x}_N, \mathbf{u}_0, \ldots, \mathbf{u}_N, \mathbf{x}_{\nu,0}, \ldots, \mathbf{x}_{\nu,N}$), N は計画を行うステップ数, J_k は軌道の性能に関するコスト関数とし, λ_k はそれらのコスト関数に関する重みとする. ϵ_Ψ は正の定数とする. 与えられた命題 Ψ のロバストネス関数が正の値となる制約は, 大域的最適解ならばロバストネス関数の値が正の値になると考えられるため, 先行研究ではこの制約が考慮されていない [10]. しかし, 本章では, 得られた解が局所的最適解の場合でもロバストネス関数の値が正の値となるように, この制約条件を陽に最適化問題で考慮することとする.

このとき, 最適化問題 (4.5) の解は α_i が常に発生しなくても実行可能解になる可能性がある. そのため, α_i が軌道上で少なくとも 1 度は発生することを仮定し, 次の拘束条件を加える:

$$\max_{j \in [0,\ldots,N]} \ \rho_{\alpha_i,j}(\mathbf{z}) \geq \epsilon_{\alpha_i}, \tag{4.6}$$

ここで ϵ_{α_i} は小さな正の定数とする.

本章では, 外的事象の発生時刻が正確には判明していないと仮定し, 次の仮定を置く:

仮定 3. 外的事象 ν の発生時刻に関する分布が $p_\nu(t)$ で与えられている.

外的事象の発生時刻に関する分布に基づいて, 不確かさを持つ外的事象に対する軌道最適化問題を次のように定式化する:

$$\begin{aligned} \min_{Z} \ & -\rho_\Psi(\mathbf{z}) - \sum_i \sum_j \lambda_{p_{\nu_i}}(j) L(\rho_{\nu_i}(x_{\nu_i,j})) \\ & + \sum_k \lambda_k J_k(\mathbf{x}_k, \mathbf{u}_k) \end{aligned} \tag{4.7}$$

$$\text{s.t. Eq.(2.2), (4.6)}, \quad \rho_\Psi(\mathbf{z}) \geq \epsilon_\Psi. \tag{4.8}$$

ここで $\lambda_{p_{\nu_i}}(t) \geq 0$ は発生に関する分布 $p_{\nu_i}(t)$ に依存した重みとする. $L : \mathbb{R} \to \mathbb{R}$ はステージコストであり, 単調増加関数とする (例: $L(x) = 2/(1+\exp(-x)) - 1$).

仮定 2 より, ステージコスト L を含む第 2 項は, ν_i が発生した場合に小さな値となる. 重み $\lambda_{p_{\nu_i}}$ は, $p_{\nu_i}(t)$ が大きな値をとる時刻では大きな値に設定することで, その時刻での ν_i の発生に対して大きな重みをつける. その結果として, 最適化問題 (4.8) の解は, 分布 $p_{\nu_i}(t)$ に従って命題 Ψ を満たす外的事象 ν_i に対する複数のシナリオを考慮した軌道が生成される.

得られた解が複数のシナリオを考慮できる理由は, 命題 Ψ 内に現れる外的事象に関する命題は $\mathbf{G}_{[a,b]}(\alpha \Rightarrow \Psi)$ の形のみであり, それぞれの外的事象の発生に対してこの命題が満たされる必要があるためである. そのため, 提案手法によって外的事象の発生に関する複数のシナリオに対して命題 Ψ を満たすロバストな軌道を生成することができる.

4.5 数値実験

この章では, 2 つの数値実験に提案手法を適用する. まず, 1 次元の通過タスクに提案手法を適用する. 次に, より複雑なタスクである 2 次元の下げ膳タスクに提案手法を適用する. 最後に, 提案手法の十分性と完全性に関して考察を述べる.

本章での全ての計算は, 3.6GHz Intel Core i9 で実行した. 全ての最適化問題は MATLAB 上で CasADi [42] によって実装し, ソルバーとして Ipopt [45] を用いた.

4.5.1 1 次元の通過タスク

1 次元システムの通過タスクとして次の命題を考える: “いつかロボットはゴールエリアに到達し, かつ常にロボットが通過エリアに存在できるのは許可信号 $\alpha_{\mathrm{permit,t}}$ が発生した後の特定の時間のみである” (Fig.4.1).

この命題を以下の Event-based STL 命題として定義する:

$$\begin{aligned}\Psi_{\mathrm{pas}} = &\mathbf{F}_{[0,T]}(\varphi_{\mathrm{exist}}) \\ &\wedge\ \mathbf{G}(\alpha_{\mathrm{permit,t}} \Rightarrow (\mathbf{G}_{[0,t)}\neg\varphi_{\mathrm{cross}} \wedge \mathbf{G}_{[t+T',T]}\neg\varphi_{\mathrm{cross}})),\end{aligned}$$

ここで φ_{exist}, φ_{cross} は命題 “ロボットがゴール領域にいる,” “ロボットが通過領域にいる” を示している. 加えて, $\alpha_{\mathrm{permit,t}}$ は時刻 t でのロボットの通過エリアでの存在を許可する外的事象とする. ロボットのダイナミクスとして 1 次元の 2 重積分器を考え, システムの状態として位置 $x_p \in [-2.5, 2.5] \subset \mathbb{R}$ と速度 $x_v \in [-4, 4] \subset \mathbb{R}$, 制御入力として加速度 $u \in [-1, 1] \subset \mathbb{R}$ とする.

各命題に対して次のロバストネス関数を定義する:

$$\begin{aligned}\rho_{\varphi_{\mathrm{exist}}}(\mathbf{x}) =& \min(x_p - 2,\ -x_p + 2.5, x_v + 0.1,\ -x_v + 0.1), \\ \rho_{\varphi_{\mathrm{cross}}}(\mathbf{x}) =& \min(x_p - 1,\ -x_p + 2), \\ \rho_{\alpha_{\mathrm{permit,t}}}(\mathbf{x}_\nu) =& x_{\alpha_{\mathrm{permit}}}(t),\end{aligned}$$

また, ロボットの初期状態として $[x_p, x_v] = [0, 0]$, 命題に含まれる時間制約を $T = 4$, $T' = 2$, 計画を行うステップ数を $N = 40$ とする. ステージコストを $L(x_{\alpha_{\mathrm{permit}}}) = 2/(1 + \exp(-\kappa_L x_{\alpha_{\mathrm{permit}}})) - 1$, $\kappa_L = 30$ とし, 目的関数は次式とする:

$$-\rho_{\Psi_{\mathrm{pas}}}(\mathbf{z}) - \sum_j \lambda_{p_{\alpha_{\mathrm{permit}}}}(j) L(x_{\alpha_{\mathrm{permit}},j}) + \sum_j \lambda_u u_j^2, \tag{4.9}$$

ここで $\lambda_u = 10^{-6}$ とする.

次の 3 つのケースに対して, 軌道最適化を行った:

I) α_{permit} が計画ステップ全体で一様に発生する,

II) α_{permit} が計画ステップの前半で一様に発生する,

III) α_{permit} が計画ステップ後半で一様に発生する.

ケース I) では, $\lambda_{p_{\alpha_{\text{permit}}}}(j) = 10^{-1}$ $(j = 0, \ldots, N)$ とし, 軌道最適化によって得られた軌道を Fig.4.2 に示す. 黄色の破線は $x_{\alpha_{\text{permit}}}$ の軌道を示しており, 軌道上に発生している α_{permit} は Ψ_{pas} を満たしていることがわかる. ケース II) では, $\lambda_{p_{\alpha_{\text{permit}}}}(j) = 10^{-1}$ $(j = 0, \ldots, \lfloor N/2 \rfloor)$, $\lambda_{p_{\alpha_{\text{permit}}}}(j) = 0$ $(j = \lfloor N/2 \rfloor + 1, \ldots, N)$ とし, 軌道最適化によって得られた軌道を Fig.4.3 に示す. $x_{\alpha_{\text{permit}}}$ の軌道は, ケース I) の軌道を時間方向前方にシフトしたような軌道になっている. ケース III) では, $\lambda_{p_{\alpha_{\text{permit}}}}(j) = 10^{-1}$ $(j = 0, \ldots, \lfloor N/2 \rfloor)$, $\lambda_{p_{\alpha_{\text{permit}}}}(j) = 0$ $(j = \lfloor N/2 \rfloor + 1, \ldots, N)$ とし, 軌道最適化によって得られた軌道を Fig.4.4 に示す. ケース II) 同様に, $x_{\alpha_{\text{permit}}}$ の軌道はケース I) の軌道を時間方向後方にシフトし, 終端時間 T の影響によって切り取られたような軌道になっている.

これらの結果から, それぞれのケースに対して, 得られた軌道が複数の $\alpha_{\text{permit},t}$ の発生シナリオに対して命題 Ψ_{pas} を満たしていることがわかる. そのため, $\alpha_{\text{permit},t}$ の発生時刻が不確かな場合でも, 生成された軌道は多くのシナリオにおいて与えられた命題を満たすことがわかる. この数値実験においては, それぞれのケースに対して, 軌道上に現れず命題 Ψ を満たすような α_{permit} の発生時刻は存在しなかった.

この数値実験では, 最適化問題を解く際に解の最適性を向上するために初期解を変更をしながら複数回計算を行った. この場合でも, 解は 3 回以内の最適化問題の計算で収束し, 合計の計算時間は 5 秒以内に収まっている.

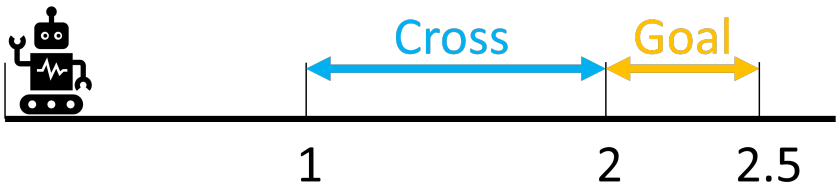

Fig. 4.1: The case of the passage task for the 1-dimensional robot

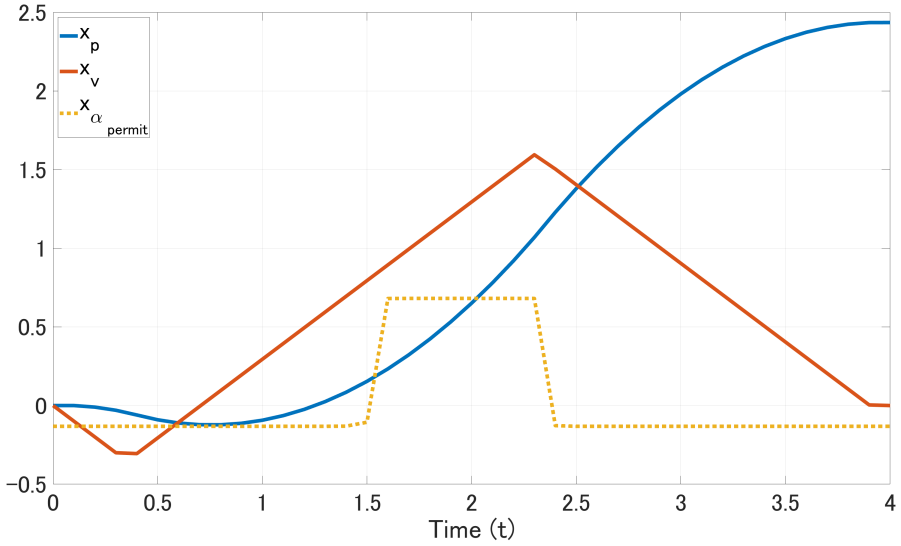

Fig. 4.2: The trajectories of the passage task in case I).

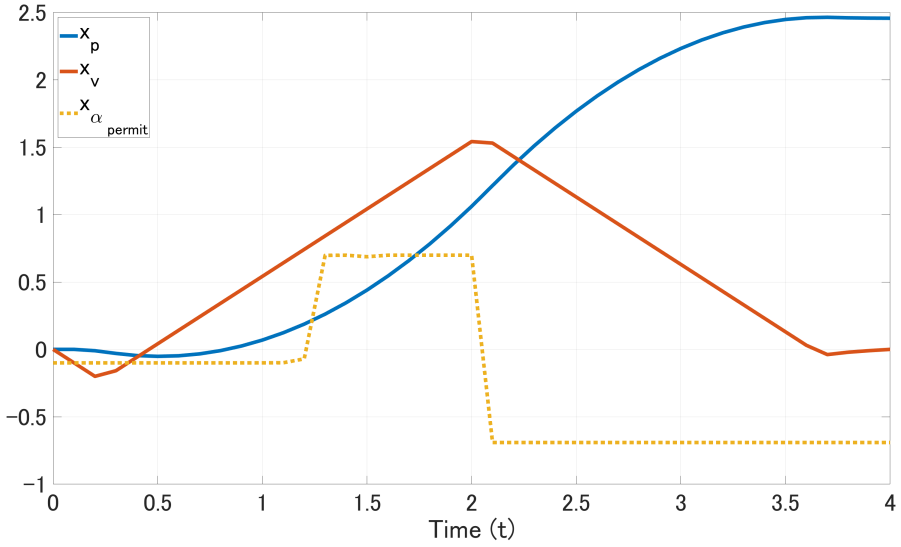

Fig. 4.3: The trajectories of the passage task in case Ⅱ).

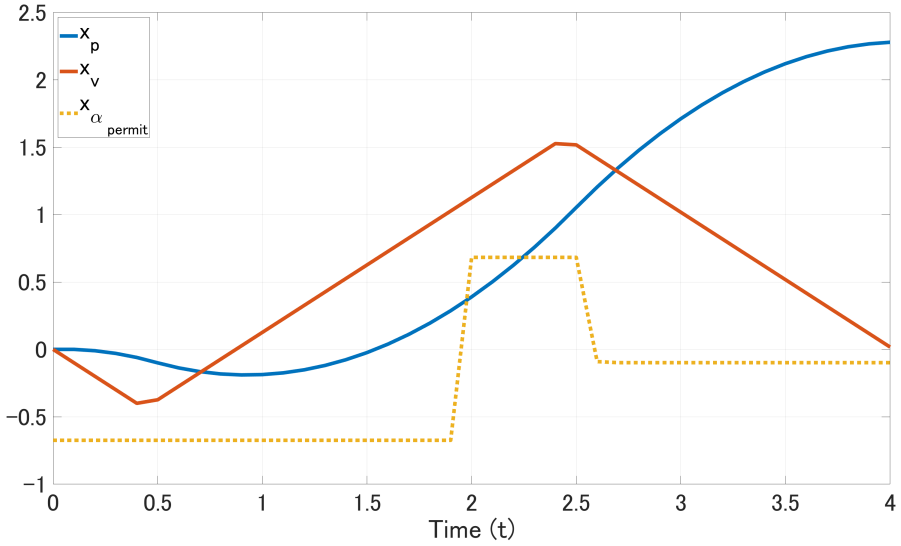

Fig. 4.4: The trajectories of the passage task in case Ⅲ).

4.5.2 2 次元の下げ膳タスク

2 次元の下げ膳タスクとして次の命題を考える: “いつか, ロボットはシンク領域に到達し, かつロボットは各テーブルの客に呼ばれたらテーブルへ到達する.” この命題を以下の Event-based STL 命題として定義する:

$$\begin{aligned}\Psi_{\mathrm{srv}} =& \mathbf{F}_{[0,T]}(\varphi_{\mathrm{Sink}}) \\ & \wedge \mathbf{G}(\alpha_{\mathrm{call}_A,t} \Rightarrow (\mathbf{F}_{[t,t+T']}\varphi_{\mathrm{table}_A})) \\ & \wedge \mathbf{G}(\alpha_{\mathrm{call}_B,t} \Rightarrow (\mathbf{F}_{[t,t+T']}\varphi_{\mathrm{table}_B})),\end{aligned} \tag{4.10}$$

ここで, φ_{Sink}, $\varphi_{\mathrm{table}\,i}\,(i=1,2)$ は命題 “ロボットがシンク領域に存在する,” “ロボットがテーブル i 領域に存在する” を表している. また, $\alpha_{\mathrm{call}i,t}\,(i=A,B)$ は時刻 t で “ロボットがテーブル i に呼ばれている” ことを示す外的事象とする. ロボットのダイナミクスは 2 次元の 2 重積分器とし, システムの状態はロボットの位置 $\mathbf{x}_p=[x_{p_x},x_{p_y}]^{\mathsf{T}}\in[0,3]\times[0,3]\subset\mathbb{R}^2$, 速度 $\mathbf{x}_v=[x_{v_x},x_{v_y}]^{\mathsf{T}}\in[-0.5,0.5]\times[-0.5,0.5]\subset\mathbb{R}^2$ とし, 制御入力は加速度 $\mathbf{u}=[u_x,u_y]^{\mathsf{T}}\in[-1,1]\times[-1,1]\subset\mathbb{R}^2$ とする.

それぞれの命題に対して, 次のロバストネス関数を定義する:

$$\begin{aligned}\rho_{\varphi_{\mathrm{Sink}}}(\mathbf{x}) &= \min(x_{p_x}-2,-x_{p_x}+2.5,x_{p_y}-2,-x_{p_y}+2.5,-||\mathbf{x}_v||^2+0.1^2),\\ \rho_{\varphi_{\mathrm{table}_A}}(\mathbf{x}) &= \min(x_{p_x}-1,-x_{p_x}+1.5,x_{p_y},-x_{p_y}+0.5,-||\mathbf{x}_v||^2+0.3^2),\\ \rho_{\varphi_{\mathrm{table}_B}}(\mathbf{x}) &= \min(x_{p_x}-0.5,-x_{p_x}+1,x_{p_y}-1.5,-x_{p_y}+2,-||\mathbf{x}_v||^2+0.3^2),\\ \rho_{\alpha_{\mathrm{call}_A,t}}(\mathbf{x}) &= x_{\alpha_{\mathrm{call}_A}}(t),\\ \rho_{\alpha_{\mathrm{call}_B,t}}(\mathbf{x}) &= x_{\alpha_{\mathrm{call}_B}}(t).\end{aligned}$$

また, ロボットの初期状態を $[\mathbf{x}^{\mathsf{T}},\mathbf{v}^{\mathsf{T}}]=[0,0,0,0]$, 命題に含まれる時間制約を $T=30$, $T'=4.5$, 計画を行うステップ数を $N=100$ とする. ステージコストを $L(x_{\alpha_{\mathrm{call}_i}})=2/(1+\exp(-\kappa_L x_{\alpha_{\mathrm{call}_i}}))-1\ (i=A,B)$, $\kappa_L=30$ とし, 目的関数は次式とする:

$$-\rho_{\Psi_{\mathrm{srv}}}-\sum_i\sum_j\lambda_{p_{\alpha_{\mathrm{call}_i}}}(j)L(x_{\alpha_{\mathrm{call}_i},j})+\sum_j\lambda_u||\mathbf{u}_j||^2, \tag{4.11}$$

ここで $\lambda_u=10^{-6}$ とする.

次の 2 つのケースに対して, 軌道最適化を行った: α_{call_A} は即座に発生し,

i) α_{call_B} は行動計画期間前半に一様に発生する,

ii) α_{call_B} は行動計画期間後半に一様に発生する.

重みを $\lambda_{p_{\alpha_{\mathrm{call}_A}}}(j)=10^{-1}\times5\exp(-j)\ (j=0,\ldots,N)$ と設定する.

ケース i) では, $\lambda_{p_{\alpha_{\mathrm{call}_B}}}(j) = 10^{-1}\ (j = 0, \ldots, \lfloor N/2 \rfloor)$, $\lambda_{p_{\alpha_{\mathrm{call}_B}}}(j) = 0\,(j = \lfloor N/2 \rfloor + 1, \ldots, N)$ とし, 軌道最適化によって得られた軌道を Fig.4.5, 4.6 に示す. ロボットが命題を達成し, テーブル A, テーブル B, シンク領域の順番で到達していることがわかる.

ケース ii) では, $\lambda_{p_{\alpha_{\mathrm{call}_B}}}(j) = 10^{-1}\ (j = 0, \ldots, \lfloor N/2 \rfloor)$, $\lambda_{p_{\alpha_{\mathrm{call}_B}}}(j) = 0\,(j = \lfloor N/2 \rfloor + 1, \ldots, N)$ とし, 軌道最適化によって得られた軌道を Fig. 4.5, 4.7 に示す. ケース ii) の場合でもロボットは命題を達成しており, 行動としてはテーブル A, シンク, テーブル B 領域の順番で到達していることがわかる. 加えて, 得られた軌道から, $\alpha_{\mathrm{call}_A,t}$ と $\alpha_{\mathrm{call}_B,t}$ の発生に関する複数のシナリオにおいて命題 Ψ_{srv} を満たす軌道が得られたことがわかる.

提案手法の有効性を確認するために, 外的事象 $\alpha_{\mathrm{call}_A,t}$ と $\alpha_{\mathrm{call}_B,t}$ の発生時刻として, ある固定した一つの時刻を仮定し軌道最適化を行った際に得られた軌道との比較を行った. 仮定した発生時刻は, $\alpha_{\mathrm{call}_A,t}$ は $t = 0$, $\alpha_{\mathrm{call}_B,t}$ は各ケースで発生しうる時間区間の中央の時刻とした.

比較のために, $\alpha_{\mathrm{call}_A,t}$ と $\alpha_{\mathrm{call}_B,t}$ の発生シナリオをそれぞれの分布に従って生成し, 各シナリオにおいて軌道が命題 Ψ_{srv} を満たすか評価した. それぞれの分布は $p_{\alpha_{\mathrm{call}_A}}(t) \sim \exp(-1/0.3t)$, ケース i) では $p_{\alpha_{\mathrm{call}_B}}(t) \sim 1$, $(t \in [0, T/2])$, $p_{\alpha_{\mathrm{call}_B}}(t) \sim 0$, $(t \in (T/2, T])$, ケース ii) では $p_{\alpha_{\mathrm{call}_B}}(t) \sim 0$, $(t \in [0, T/2))$, $p_{\alpha_{\mathrm{call}_B}}(t) \sim 1$, $(t \in [T/2, T])$ とした. 10000 通りのシナリオを用意し評価を行った結果, 命題を満たすことができたシナリオの割合を Table.4.1 に示す. 結果から, ケース ii) において提案手法によって命題を満たすことのできる発生シナリオが増加していることが確認できる.

Case	i)	ii)
fixed occurrence time	76.21%	65.94%
proposed method	76.21%	86.33%

Table 4.1: Comparison of trajectories of the proposed method and the optimization with fixed occurrence time

しかし, この数値実験では軌道上に現れず命題 Ψ を満たす $\alpha_{\mathrm{call}_i,t}\,(i = A, B)$ の発生時刻が存在する. このことについては, 次の小節にて議論を行う.

この数値実験では, 最適化問題を解く際に解の最適性を向上するために初期解を変更をしながら複数回計算を行った. この場合でも, 解は 3 回以内の最適化問題の計算で収束し, 合計の計算時間は 4 分以内に収まっている.

4.5.3 提案手法の十分性

この節では, 提案手法で得られる解が外的事象の発生時刻に対して十分性を持つことを示す. ここで, 十分性とは解に現れる外的事象の発生時刻のなかで, 与えられた命題を満たさない時刻が存在しないことである. 提案手法では, 解として得られた軌道内の各 ν_i の発生に対

して, 解として得られた軌道は与えられた命題を満たしている. しかし, 本論文で用いている max 関数のなめらかな近似が under-approximation ではないため, 実際の最適化計算によって得られた解は十分性を満たさない可能性がある. この問題に関しては under-approximation であるなめらかな近似を用いることで解決することができる [46]. 一方で, 提案手法で得られる解は完全性を持たない. なぜならば, 提案手法の解として得られる軌道上に現れず命題 Ψ を満たすような ν_i の発生時刻が存在するためである. 提案手法が完全性を満たさない理由のひとつは, 現実には提案手法の大域的最適解を得ることが難しいことであると考えられる.

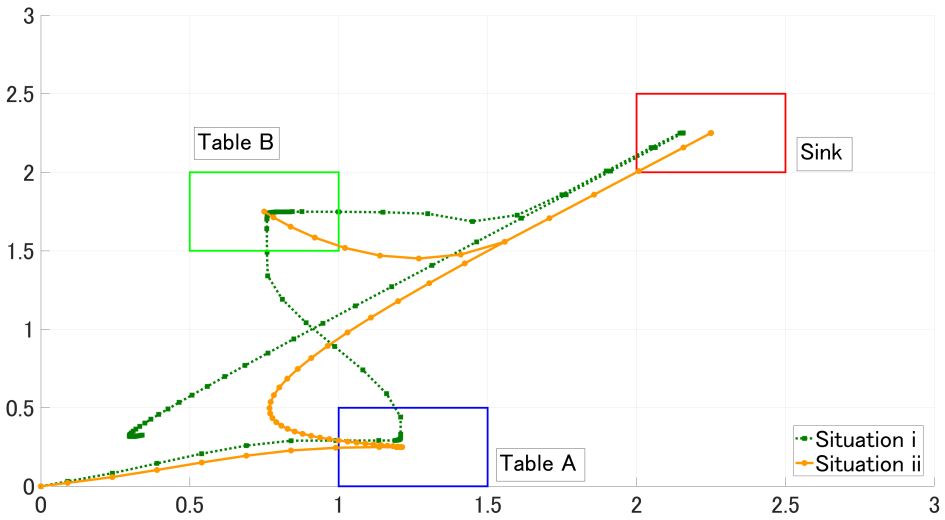

Fig. 4.5: The trajectories of the service task in 2-dimensional space for each cases. Green: case i). Orange: case ii).

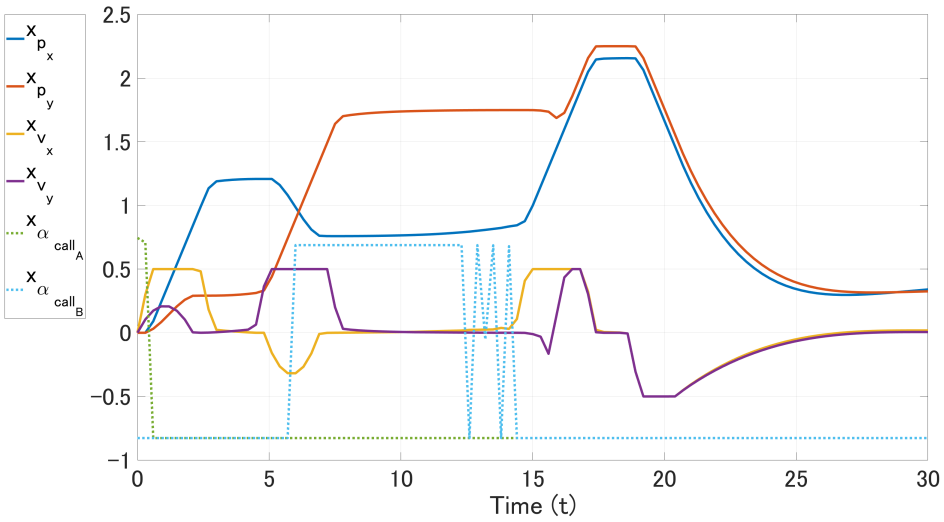

Fig. 4.6: The trajectories of the service task in case i). We can see that the robots visited the table A, the table B, and the Sink area in that order.

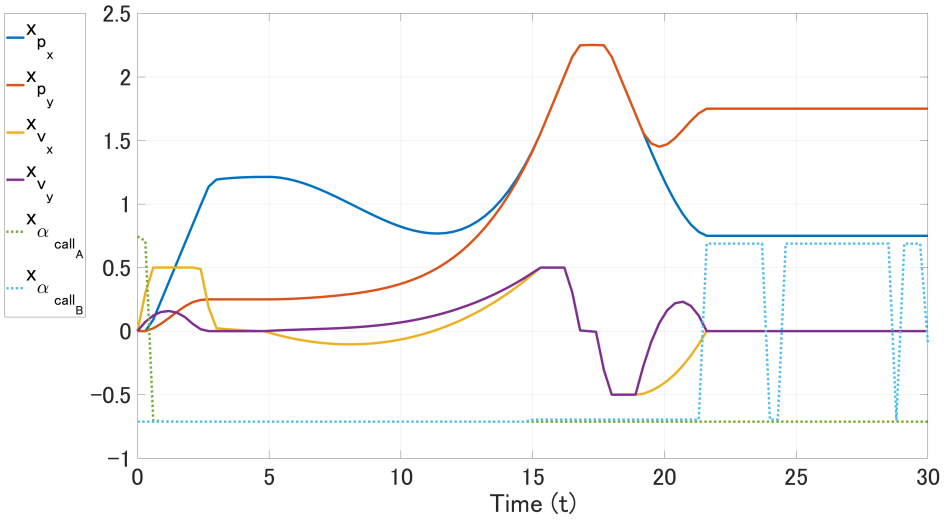

Fig. 4.7: The trajectories of the service task in case ii). We can see that the robots visited the table A, the Sink, the table B area in that order.

4.6 まとめ

本章では, 不確かな外部事象に関する Event-based STL 命題のための新しい非線形最適化による行動計画手法を提案した. まず, Event-based STL 命題のなめらかなロバストネス関数を定義した. 次に, 不確実な外的事象を含む Event-basd STL 命題のための軌道最適化手法を提案した. 最後に, 数値シミュレーションにより本手法の有効性を実証した.

本章では, 外的事象の発生に関する分布が事前に得られていることを仮定している. しかし, 実際にはタスクの実行中に分布が変化することが考えられる. そのため, 分布を観測情報を用いて更新することで, よりロバストな軌道を生成し, 現実的なケースに対応することが可能であると考えられる.

第5章　時間制約をもつ信号時相論理に対する制御

5.1　はじめに

時相論理で表現されたタスクを達成するためには, 抽象化したシステムで生成した軌道を基に実際のシステム上で制御を行い, 動作を実行する必要がある. 加えて, 実際のロボットシステム上で制御を行う際, タスクを達成するための時間的な制約や状態に関する制約条件を違反することなく動作を行う必要がある.

これをふまえ, 本章では時間制約をもつ様相作用素を含む信号時相論理を達成するための制御バリア関数を用いた制御手法を提案する. まず, 本章で考える時間制約を持つ STL と, それに対応する時間オートマトンについて説明する. 次に, 本章で提案する制御手法を説明する準備として, 零化制御バリア関数に関するいくつかの定義と時間軸変換について述べる. その後, 本章で提案する制御器の構造について説明し, 有限時間で目的の不変集合へ遷移させるための制御バリア関数を拡張した制御器を提案する. 加えて, 関数の値によって進行速度が変化する仮想時間を定義し, 時間オートマトン上の各状態遷移に対応する命題を達成するための制御器を提案する. 最後に, 移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

5.2　準備

5.2.1　対象システム

本章では, 以下の入力アフィンシステムを対象とする.

$$\dot{\mathbf{x}} = f(\mathbf{x}) + g(\mathbf{x})\mathbf{u}, \tag{5.1}$$

ここで $f(\mathbf{x}) : \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}^{n_{\mathbf{x}}}$, $g(\mathbf{x}) : \mathbb{R}^{n_{\mathbf{x}}} \to \mathbb{R}^{n_{\mathbf{x}} \times n_{\mathbf{u}}}$ は局所リプシッツ連続とする.

5.2.2　時間オートマトン

STL 論理式 (2.4) で記述される各命題に対して, 対応した時間オートマトンを考えることができる [29]. 時間オートマトンは有限オートマトンの拡張であり, 事象と時間経過の両方による状態遷移をもつモデルである (Fig.5.1).

定義 3. (時間オートマトン [47]) 時間オートマトン $\mathcal{A}$ は組 $\mathcal{A} = (\mathsf{Q}, \mathsf{Q}_0, \mathsf{C}, \mathrm{Inv}, \mathsf{E}, \mathsf{F}, \mathsf{AP})$ によって定義される. ここで Q はロケーションの集合, $\mathsf{Q}_0 \subseteq \mathsf{Q}$ は初期ロケーションの集合, C はクロックの集合, $\mathrm{Inv} : \mathsf{Q} \to \Xi(\mathsf{C})$ はクロックに関する関数, $\Xi(\mathsf{C})$ はクロックに関する制約の集合, $\mathsf{E} \subseteq \mathsf{Q} \times \mathsf{Q} \times 2^{\mathsf{AP}} \times \Xi(\mathsf{C}) \times 2^{C}$ はエッジの集合, $\mathsf{F} \subseteq \mathsf{Q}$ は受理ロケーションの集合とする.

時間オートマトンの状態 (q, c) はロケーション $q \in \mathsf{Q}$ とクロックの値 c の組として定義される.

また, 本章では STL の原子論理命題の真偽はロボットシステムの状態を引数とするロバストネス関数によって決定されると仮定する. このとき, ロボットシステムの軌道が与えられたとき, 軌道上の各時刻の状態に対して対応する時間オートマトンの状態が存在し, 軌道に対応するオートマトンの上のパスを計算することができる.

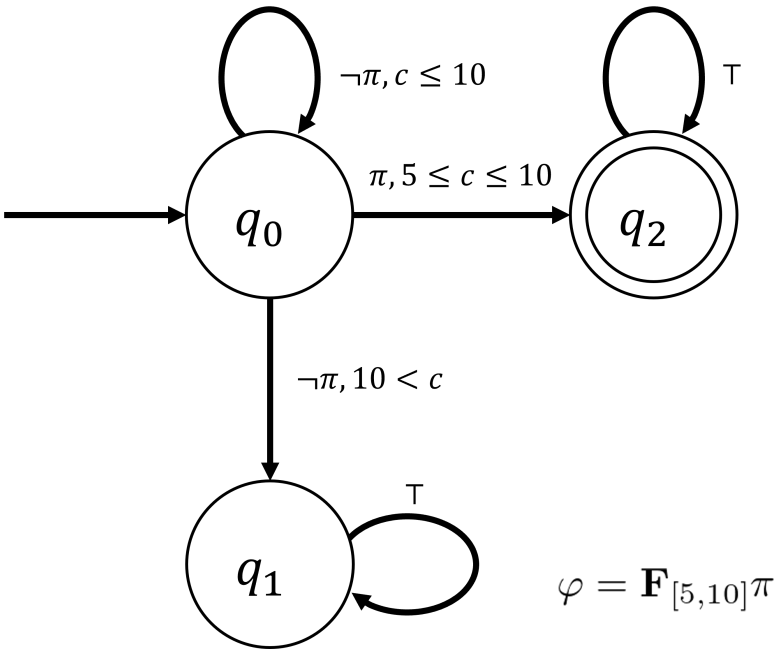

Fig. 5.1: An example of a timed automaton for STL

5.2.3 零化制御バリア関数

零化制御バリア関数ついて定義する準備として, 以下の閉集合を定義する.

$$\mathcal{C}_\delta = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} : b(\mathbf{x}) \geq \delta\}, \tag{5.2}$$

$$\partial\mathcal{C}_\delta = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} : b(\mathbf{x}) = \delta\}, \tag{5.3}$$

$$Int(\mathcal{C}_\delta) = \{\mathbf{x} \in \mathbb{R}^{n_\mathbf{x}} : b(\mathbf{x}) > \delta\}, \tag{5.4}$$

ここで $b : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}$ は微分可能な連続関数, $\delta \geq 0$ は制約条件 $b(\mathbf{x}) \geq 0$ に対するマージンとする. ただし $\delta = 0$ のとき, $\mathcal{C}_\delta$ の代わりに $\mathcal{C}$ と記述する.

次に, 拡張クラス $\mathcal{K}$ 関数は以下のように定義される.

定義 4. (拡張クラス $\mathcal{K}$ 関数 [48]) 連続関数 $\zeta : (-l_1, l_2) \to (-\infty, \infty)$, l_1, $l_2 > 0$ が厳密に単調増加し $\zeta(0) = 0$ を満たすとき, ζ は拡張クラス $\mathcal{K}$ 関数に属する.

ここで, 入力アフィンシステム (5.1) に対して, 零化制御バリア関数 (ZCBF) を以下のように定義する.

定義 5. (零化制御バリア関数 [48]) 微分可能な連続関数 $b : \mathbb{R}^{n_\mathbf{x}} \to \mathbb{R}$ に対して式 (5.2)-(5.4) で定義される集合 $\mathcal{C} \subset \mathbb{R}^{n_\mathbf{x}}$ を考える. このとき下記の不等式を満たす拡張クラス $\mathcal{K}$ 関数 ζ が存在するとき, b は集合 $\mathcal{D}$, $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^{n_\mathbf{x}}$ に対する ZCBF である:

$$\sup_{\mathbf{u} \in \mathcal{U}} [L_f b(\mathbf{x}) + L_g b(\mathbf{x}) \mathbf{u} + \zeta(b(\mathbf{x}))] \geq 0, \forall \mathbf{x} \in \mathcal{D}, \tag{5.5}$$

ZCBF b が与えられたとき, $\mathbf{x} \in \mathcal{D}$ に対して集合 $\mathcal{U}_{zcbf}(\mathbf{x})$ を下記の様に定義する:

$$\mathcal{U}_{zcbf}(\mathbf{x}) = \{\mathbf{u} \in \mathcal{U} \,|\, L_f b(\mathbf{x}) + L_g b(\mathbf{x}) \mathbf{u} + \zeta(b(\mathbf{x})) \geq 0\}. \tag{5.6}$$

もし b が $\mathcal{D}$ に対する ZCBF ならば, 任意のリプシッツ連続な制御器 $\mathbf{u} : \mathcal{D} \to \mathcal{U}$, $\mathbf{u}(\mathbf{x}) \in \mathcal{U}_{zcbf}(\mathbf{x})$ によって $\mathcal{C}$ は不変集合となる.

5.2.4 時間軸変換

時間軸変換は非線形システムの制御や解析に用いられている [49]. 仮想時間を用いることで, システムを扱いやすい構造に変換することや望ましい特性を付与することができる. 仮想時間軸 $\tau \in \mathcal{T}$ を次のように定義する:

$$\frac{dt}{d\tau} = s(\mathbf{x}, t, \tau),\ \tau(t_0) = \tau_0, \tag{5.7}$$

ここで $s(\mathbf{x}, t, \tau)$ は時間軸関数と呼び, 次の不等式を満たすと仮定する:

$$0 < s(\mathbf{x}, t, \tau) < \infty. \tag{5.8}$$

新しい時間軸を用いてシステム (5.1) は次のように表現できる:

$$\frac{d\mathbf{x}}{d\tau} = s(\mathbf{x}, t, \tau) \left(f(\mathbf{x}(t)) + g(\mathbf{x}(t)) \mathbf{u}(t) \right). \tag{5.9}$$

5.3 仮想時間を用いた ϵ-零化制御バリア関数 (ϵ-ZCBFs)

Fig. 5.2 に本章の提案手法の概要を示す. 提案手法は参考文献 [50] の制御手法と同様の構成をしている. 参考文献 [50] との大きな違いは, 制御器の設計方法である. 時間制約をもつ STL 命題を達成するための CBF の設計手法は様々な手法が提案されている [50, 51]. 参考文献 [51] では, CBF を時間を引数とする関数として定義することで, 命題の持つ時間制約に対応している. 参考文献 [50] では, CBF を時間を引数とする関数として定義した上で, 有限

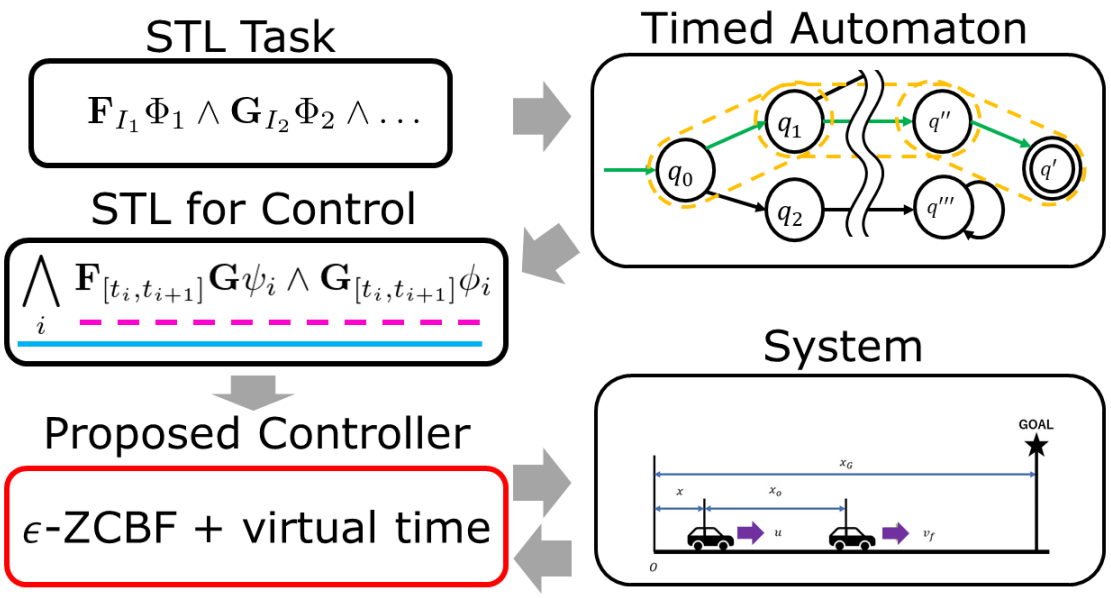

Fig. 5.2: Overview of proposed method in Chapter 5. First, we consider a timed automaton of given STL task. We assume that we can obtain an accepting run (green run) of the timed automaton. We convert the accepting run to a new STL specification (blue underline) which has STL propositions (pink underline) corresponding to each transition between locations of the accepting run (orange enclosed parts). Then, we set ϵ-ZCBFs and the virtual time from the new STL specification. The proposed controller controls a system and finally achieve given STL task.

時間整定が可能な CBF を用いて制御を行ってる. 一方で, 本章では, 後述するように時不変な関数である ϵ-ZCBF を定義し, 制御を行うことで命題の持つ時間制約に対応してる. また, CBF を設計する際のロバストネス関数も異なっており, 参考文献 [50, 51] では遷移に関する 2 つの命題に関する softmin 用いた関数に基づいて設計しているの対して, 本章の手法では各命題に対して CBF を設計することで, 後述するように仮想時間を導入し, 解の消失のリスクを低減できる. そのため, 本章の手法では参考文献とは異なり, 新たな時間関数を定義する必要がなく, 加えて, 仮想時間を導入した制御器を設計できる.

提案手法では, まず与えられた STL に対する時間オートマトンを考える. 本章では, 時間オートマトンに対して受理されるパスが, 既存の手法によって得られると仮定する [13, 52]. 得られた受理されるパスに対して, パス上の時間オートマトンのロケーション間の遷移条件をそれぞれ新しく STL を用いて記述する. 実際のシステム上で新しく生成した STL 命題を達成するように制御を行い, 各ロケーション間の遷移を達成することで本来のタスクを達成することができる.

本節では, 提案手法の各要素について説明する. まず ϵ-ZCBF を定義し, ϵ-ZCBF を用いた制御器が, 有限時間で状態を目的の集合に遷移させることを示す. 次に, 時間制約を緩和するための時間軸変換について説明する. 提案する時間軸変換によって, 関数の値と時間に依存

して速度が変化する仮想時間を定義する. その後, ϵ-ZCBF と仮想時間を組み合わせ, 仮想時間上で ϵ-ZCBF を用いた制御器が, 指定された時間制約を守りながら状態を目的の不変集合に遷移させることを示す. 最後に, 提案方法を STL で記述されたタスクに適用する方法を示す. この際, 仮想時間上で ϵ-ZCBF を用いた制御器を実装することにより、達成するべきロケーション間の遷移に関する制約を緩和することで, 常に安全性に関する命題を満たすことを示す.

5.3.1 ϵ-零化制御バリア関数

状態集合に到達するタスクを考えた際, そのタスクに時間制約があるならば, 状態を有限時間内に目的の不変集合に収束させる必要がある. 時間制約がある到達タスクを達成するための制御手法として ϵ-ZCBF を提案する. ϵ-ZCBF を次のように定義する.

定義 6. (ϵ-零化制御バリア関数) 動的システム (5.1) と式 (5.2)-(5.4) で定義される微分可能な連続関数 $b : \mathcal{X} \to \mathbb{R}$ に関する集合 $\mathcal{C} \subset \mathbb{R}^{n_\times}$ を考える. もし定数 ϵ, ϱ_ϵ が存在し, 以下の不等式を満たすならば b は集合 $\mathcal{D}$, $\mathcal{C} \subseteq \mathcal{D} \subset \mathbb{R}^{n_\times}$ に対する ϵ-ZCBF である:

$$\sup_{\mathbf{u}\in\mathcal{U}}[L_f b(\mathbf{x}(t)) + L_g b(\mathbf{x}(t))\mathbf{u} + \varrho_\epsilon(b(\mathbf{x}(t)) - \epsilon)] \geq 0, \tag{5.10}$$

加えて, 定数 ϵ, ϱ_ϵ は次の条件を満たす:

$$b_{max} > \epsilon > 0,\ \varrho_\epsilon > 0, \tag{5.11}$$

ここで $b_{max} = \max_{\mathbf{x}\in\mathcal{X}} b(\mathbf{x})$.

ϵ-ZCBF b が与えられたとき, 状態 $\mathbf{x}$ に対して不等式 (5.10) を満たす入力の集合を $\mathcal{U}_\epsilon(\mathbf{x}) = \{\mathbf{u} \in \mathcal{U} \,|\, L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u} + \varrho_\epsilon(b(\mathbf{x}) - \epsilon) \geq 0\}$ とする. 次の補題にて $\mathcal{U}_\epsilon$ の性質を述べる.

補題 1. 動的システム (5.1) と式 (5.2)-(5.4) で定義される微分可能な連続関数 $b : \mathcal{X} \to \mathbb{R}$ に関する集合 $\mathcal{C} \subset \mathbb{R}^{n_\times}$ を考える. もし b が集合 $\mathcal{D} \supseteq \mathcal{C}$ に対する ϵ-ZCBF ならば, 任意のリプシッツ連続な制御器 $\mathbf{u} : \mathcal{D} \to \mathcal{U}$, $\mathbf{u}(\mathbf{x}) \in \mathcal{U}_\epsilon(\mathbf{x})$ は有限時間 $0 < T^* < \infty$ で状態を集合 $\mathcal{C}$ に遷移させる: $\mathbf{x}(T^*) \in \mathcal{C}$.

このとき, 到達時間 T^* は, もしシステムの初期状態が $\mathbf{x}(0) \in \mathcal{D}\backslash\mathcal{C}$ ならば, $T^* = \frac{1}{\varrho_\epsilon}\ln(\frac{-b(\mathbf{x}(0))+\epsilon}{\epsilon})$ である. $\mathbf{x}(0) \in \mathcal{C}$ ならば $T^* = 0$ である. さらに, システムの状態は, 全ての時刻 $t \geq T^*$ で $\mathcal{C}$ に存在する.

Proof. 定義 6 から, $\forall \mathbf{x} \in \mathcal{D}\backslash Int(\mathcal{C}), b(\mathbf{x}) \leq 0$ であり $b(\mathbf{x}) - \epsilon \leq 0$ となる. そのため, $\mathcal{U}_\epsilon$ の定義から $\forall \mathbf{x} \in \mathcal{D}\backslash Int(\mathcal{C}), \forall t \in \mathbb{R}_+, \frac{d}{dt}(b(\mathbf{x}(t)) - \epsilon) \geq -\varrho_\epsilon(b(\mathbf{x}(t)) - \epsilon) > 0$ が成り立つ. このとき, $\forall t$ に対して,

$$\begin{aligned} b(\mathbf{x}(t)) - \epsilon &= b(\mathbf{x}(0)) - \epsilon + \int_0^t \frac{d}{d\tau}(b(\mathbf{x}(\tau)) - \epsilon)d\tau \\ &\geq e^{-\varrho_\epsilon t}(b(\mathbf{x}(0)) - \epsilon), \end{aligned}$$

を満たす．さらに, 時刻 $T^* = \frac{1}{\varrho_\epsilon}\ln(\frac{-b(\mathbf{x}(0))+\epsilon}{\epsilon})$ で, $b(\mathbf{x}(T^*)) \geq e^{-\varrho_\epsilon T^*}(b(\mathbf{x}(0)) - \epsilon) + \epsilon = 0$ かつ, $\forall \mathbf{x} \in \partial\mathcal{C}, \dot{b}(\mathbf{x}) > 0$ が成り立つ．加えて, 南雲の定理から, 集合 $\mathcal{C}$ は不変集合となる [53] [54]. □

実際のタスクでは, 到達時間 $T_\mathcal{C}$ は制約条件として事前に与えられている．そのため, 時間制約を満たすように定数 ϵ, ϱ_ϵ を以下のように決定する．

補題 2. フィードバック制御器 $\mathbf{u}(\mathbf{x}) \in \mathcal{U}_\epsilon(\mathbf{x})$ と到達時間 $T_\mathcal{C} \in \mathbb{R}$, 定義 6 で定義された入力の集合 $\mathcal{U}_\epsilon(\mathbf{x})$ を考える．もし b が ϵ-ZCBF で定数 ϵ, ϱ_ϵ が次の不等式 (5.12) を満たすとき, 制御器 $\mathbf{u}(\mathbf{x})$ は状態を式 (5.2)-(5.4) で定義された b に関する集合 $\mathcal{C}_\delta$ に有限時間 T^*, $0 < T^* \leq T_\mathcal{C} < \infty$ で遷移させる: $\mathbf{x}(T^*) \in \mathcal{C}_\delta$. さらに, システムの状態は全ての時刻 $t \geq T^*$ で $\mathcal{C}_\delta$ に存在する．

$$e^{-\varrho_\epsilon T}(b(\mathbf{x}_0) - \epsilon) + \epsilon \geq \delta,\ \epsilon \geq \delta. \tag{5.12}$$

Proof. ϵ, ϱ_ϵ が不等式 (5.12) を満たすとき, 補題 1 の証明から, $b(\mathbf{x}(T)) \geq e^{-\varrho_\epsilon T}(b(\mathbf{x}_0)-\epsilon)+\epsilon \geq \delta$ が時刻 $T_\mathcal{C}$ で成立. そのため, $\mathbf{x}(T^*) \in \partial C_\delta$ を満たす $T^* \leq T_\mathcal{C}$ が存在し, システムの状態は T^* で $\mathcal{C}_\delta$ に遷移される. 加えて, $\forall \mathbf{x} \in \partial\mathcal{C}_\delta, \dot{b}(\mathbf{x}) > 0$ から, 補題 1 と同様に集合 $\mathcal{C}_\delta$ は不変集合となる. □

5.3.2 関数値に基づく時間軸変換

最適化問題を解く際, 複数の制約条件が存在するとき, 解が消失する場合がある．もし制御入力を計算するための最適化問題の解が得られない場合, システムを制御することが不可能になる．そのため, 時間軸変換及び仮想時間を導入し, 入力に関する制約を時間制約の観点から緩和することで解の消失のリスクを低減する．ここでは, 制御器に導入する仮想時間の定義とその特性について述べる．関数 ρ に対して仮想時間を次のように定義する．

定義 7. (関数値に基づく仮想時間) 微分可能な連続関数 $\rho : \mathcal{X} \to \mathbb{R}$ と到達時間 $T_\mathcal{C}$ を考える．$\forall t \in [t_0, T_\mathcal{C}), \infty > \rho(\mathbf{x}(t)) > 0$ を仮定する．このとき, $\rho(\mathbf{x})$ に関する仮想時間 τ を以下のように定義する:

$$\frac{dt}{d\tau} = s(\mathbf{x}, t, \tau) := s_1(\mathbf{x}) \cdot s_2(t, \tau), \tag{5.13}$$

$$\tau(t_0) = t_0, \tag{5.14}$$

$$s_1(\mathbf{x}) = \max\left(-\ln\left(\frac{1}{\beta}\rho(\mathbf{x})\right), 0\right) + 1, \tag{5.15}$$

$$s_2(t, \tau) = \frac{T_\mathcal{C} - t}{T_\mathcal{C} - \tau + \Delta_\tau},\ t_0 \leq t < T_\mathcal{C} \tag{5.16}$$

ここで, β は制約を緩和する領域に関わるパラメーター, $\Delta_\tau > 0$ は仮想時間の進行速度に関わるパラメーターとする．

時刻 t に対する仮想時間 τ の進行速度は $\frac{d\tau}{dt} = \frac{1}{s(\mathbf{x},t,\tau)}$ で定義される. もし $s(\mathbf{x},t,\tau) > 1$ ならば, τ は t より遅く進み, $1 > s(\mathbf{x},t,\tau) > 0$ ならば, τ は t より速く進む. もし $s(\mathbf{x},t,\tau) = 1$ ならば, τ は t と同じ速度で進む.

仮想時間 τ の時間微分は2つの要素で構成されている: 制約を緩和する項 $s_1(\mathbf{x})$ と時間を早送りする項 $s_2(t,\tau)$ である. もし $\mathbf{x} \notin \{\mathbf{x}|\rho(\mathbf{x}) < \beta\}$ ならば, $s_1(\mathbf{x}) = 1$ かつ $s(\mathbf{x},t,\tau) = s_2(t,\tau)$. もし $\mathbf{x} \in \{\mathbf{x}|\rho(\mathbf{x}) < \beta\}$ ならば, $s_1(\mathbf{x}) > 1$ であり, 加えて $s_2(t,\tau) > 1/s_1(\mathbf{x})$ ならば, $s(\mathbf{x},t,\tau) > 1$ であり, τ は t より遅く進む. 加えて, τ が t に対して遅れるほど, $s_2(t,\tau)$ は小さくなる. そのとき, $s(\mathbf{x},t,\tau) < 1$ であれば, τ は t より速く進む.

仮想時間 τ について, 次の補題が成立する.

補題 3. 時間 t および定義 7 で定義された仮想時間 τ, パラメーター $\Delta_\tau > 0$ と到達時間 $T_\mathcal{C} > 0$ を考える. このとき, 任意の実数 σ_τ, $T_\mathcal{C} + \Delta_\tau > \sigma_\tau > 0$ に対して, ある時刻 $\tilde{T_\mathcal{C}}$, $T_\mathcal{C} > \tilde{T_\mathcal{C}}$ が存在し, 時刻 $\tilde{T_\mathcal{C}}$ で $\tau(\tilde{T_\mathcal{C}}) \geq T_\mathcal{C} + \Delta_\tau - \sigma_\tau$ が成立する.

Proof. まず, 時間区間 $t \in [t_0, T_\mathcal{C})$ で $\tau < T_\mathcal{C} + \Delta\tau$ を満たすことを示す. 変数分離法により τ の時間微分 $\frac{d\tau}{dt} = \frac{1}{s_1(\mathbf{x}(t))s_2(t,\tau)}$ を $\tau(t_0) = \tau_0 < T_\mathcal{C} + \Delta_\tau$ のもとで τ について解き, t の関数として次式を得る.

$$\tau(t) = T_\mathcal{C} + \Delta_\tau - C_1 \exp\left(\int_{t_0}^{t} \frac{-1}{s(\mathbf{x}(t'))(T_\mathcal{C} - t')} dt'\right) \tag{5.17}$$

$$= T_\mathcal{C} + \Delta_\tau - C_1 \Theta(t), \tag{5.18}$$

ここで $C_1 = T_\mathcal{C} + \Delta_\tau - \tau_0$ は積分定数, $\Theta(t) = \exp\left(\int_{t_0}^{t} \frac{-1}{s(\mathbf{x}(t'))(T_\mathcal{C} - t')} dt'\right)$ とする. このとき, $\tau(t^\star) \geq T_\mathcal{C} + \Delta\tau$ かつ $t^\star < T_\mathcal{C}$ を満たす $t^\star$ が存在すると仮定する. そのとき, 以下の関係が成り立つ;

$$-C_1 \Theta(t^\star) \geq 0. \tag{5.19}$$

しかし, $-C_1 < 0$ かつ $\Theta(t) > 0$ が任意の $t \in [t_0, T_\mathcal{C})$ で成り立つため, $-C_1\Theta(t) < 0$ が成立する. したがって背理法により, $\tau(t) < T_\mathcal{C} + \Delta\tau$ が $t \in [t_0, T_\mathcal{C})$ で成立.

次に, 時間区間 $t \in [t_0, \tilde{T_\mathcal{C}}]$, $\tilde{T_\mathcal{C}} < T_\mathcal{C}$ における τ の軌道を考える. ここで, $\bar{\tau} = \tau - T_\mathcal{C} - \Delta\tau$ を定義する. $\frac{d\tau}{dt}$ は時間区間 $t \in [t_0, T_\mathcal{C})$ でリプシッツ連続であるため, $t \in [t_0, \tilde{T_\mathcal{C}}]$ で τ の解が存在する. そのため, $t \in [t_0, \tilde{T}]$ で $\bar{\tau}$ の解が存在し $\bar{\tau} < 0$ が成り立つ. 加えて, $[t_0, \tilde{T_\mathcal{C}}]$ でのシステムの軌道に対する $\frac{1}{s_1(\mathbf{x}(t))}$ の最小値 $c = \min_t \frac{1}{s_1(\mathbf{x}(t))}$ を考える. 定義 7 から, $\frac{d\bar{\tau}}{dt} = \frac{1}{s_1(\mathbf{x}(t))s_2(t,\bar{\tau})}$ は下記の様に記述できる:

$$\begin{aligned}
\frac{d\bar{\tau}}{dt} &= -\frac{1}{s_1(\mathbf{x}(t))(T_\mathcal{C} - t)}\bar{\tau} \\
&= -\frac{c}{T_\mathcal{C} - t}\bar{\tau} + \left(-\frac{1}{T_\mathcal{C} - t}\left(\frac{1}{s_1(\mathbf{x}(t))} - c\right)\bar{\tau}\right) \\
&= -\frac{c}{T_\mathcal{C} - t}\bar{\tau} + \theta(t,\bar{\tau}),\ \theta(t,\bar{\tau}) = \left(-\frac{1}{T_\mathcal{C} - t}\left(\frac{1}{s_1(\mathbf{x}(t))} - c\right)\bar{\tau}\right).
\end{aligned} \tag{5.20}$$

さらに, 新たな仮想時間 $\hat{\tau}$ を考える,

$$\frac{d\hat{\tau}}{dt} = -\frac{c}{T_{\mathcal{C}} - t}\hat{\tau} + \theta(t, \bar{\tau}). \tag{5.21}$$

このとき, $\mathrm{e}_\tau = \bar{\tau} - \hat{\tau}$ の時間微分は,

$$\begin{aligned}\frac{d\mathrm{e}_\tau}{dt} &= \frac{d\bar{\tau}}{dt} - \frac{d\hat{\tau}}{dt} \\ &= -\frac{c}{T_{\mathcal{C}} - t}\bar{\tau} + \theta(t, \bar{\tau}) - \left(\frac{c}{T_{\mathcal{C}} - t}\hat{\tau} + \theta(t, \bar{\tau})\right) \\ &= -\frac{c}{T_{\mathcal{C}} - t}\mathrm{e}_\tau.\end{aligned} \tag{5.22}$$

よって, e_τ の軌道は,

$$\mathrm{e}_\tau = C_{\mathrm{e}}(T_{\mathcal{C}} - t)^c, \tag{5.23}$$

ここで, $C_{\mathrm{e}} = \frac{\bar{\tau}(t_0) - \hat{\tau}(t_0)}{(T_{\mathcal{C}} - t_0)^c}$ は積分定数. さらに, $\hat{\tau}$ の軌道は,

$$\hat{\tau}(t) = (T_{\mathcal{C}} - t)^c \left(\int_{t_0}^{t} \frac{1}{(T_{\mathcal{C}} - t')^c}\theta(t', \bar{\tau})dt' + C_2\right), \tag{5.24}$$

ここで, $C_2 = \frac{\hat{\tau}(t_0)}{(T_{\mathcal{C}} - t_0)^c}$ は積分定数. $t \in [t_0, \tilde{T}_{\mathcal{C}}]$ で $\bar{\tau} < 0$, $\frac{1}{T_{\mathcal{C}} - t} > 0$ かつ $1 \geq \frac{1}{s_1(\mathbf{x}(t))} \geq c$ が成り立つため, 下式を得る

$$\begin{aligned}&\left(-\frac{1}{T_{\mathcal{C}} - t}(1 - c)\,\bar{\tau}\right) \geq \theta(t, \bar{\tau}) \geq \left(-\frac{1}{T_{\mathcal{C}} - t}(c - c)\,\bar{\tau}\right) \\ &\left(-\frac{1}{T_{\mathcal{C}} - t}(1 - c)\,\bar{\tau}\right) \geq \theta(t, \bar{\tau}) \geq 0.\end{aligned} \tag{5.25}$$

式 (5.24) と不等式 (5.25) から, 次の不等式を得る:

$$\begin{aligned}\hat{\tau}(t) &= (T_{\mathcal{C}} - t)^c \left(\int_{t_0}^{t} \frac{1}{(T_{\mathcal{C}} - t')^c}\theta(t', \bar{\tau}) + C_2\right) \\ &\geq (T_{\mathcal{C}} - t)^c \left(\int_{t_0}^{t} \frac{1}{(T_{\mathcal{C}} - t')^c} \cdot 0dt' + C_2\right) \\ &= C_2(T_{\mathcal{C}} - t)^c.\end{aligned} \tag{5.26}$$

このとき, $t \in [t_0, \tilde{T}_{\mathcal{C}}]$ で $\hat{\tau}(t) \geq C_2(T_{\mathcal{C}} - t)^c$ が成り立つ. 式 (5.23) から, $t \in [t_0, \tilde{T}_{\mathcal{C}}]$ で $0 > \bar{\tau}(t) \geq (C_{\mathrm{e}} + C_2)(T_{\mathcal{C}} - t)^c$ かつ $C_{\mathrm{e}} + C_2 = \frac{\bar{\tau}(t_0)}{(T_{\mathcal{C}} - t_0)^c}$ が成立. よって, 全ての σ_τ に対して, ある $\tilde{T}_{\mathcal{C}}$ が存在して, $(C_{\mathrm{e}} + C_2)(T - \tilde{T}_{\mathcal{C}})^c \geq -\sigma_\tau$ かつ $T_{\mathcal{C}} + \Delta_\tau > \sigma_\tau > 0$, $T_{\mathcal{C}} > \tilde{T}_{\mathcal{C}}$ を満たす. 従って, 全ての σ_τ, $T_{\mathcal{C}} + \Delta_\tau > \sigma_\tau > 0$ に対して, ある時刻 $\tilde{T}_{\mathcal{C}} < T_{\mathcal{C}}$ が存在し, $t = \tilde{T}_{\mathcal{C}}$ のとき, $\tau(t) \geq T_{\mathcal{C}} + \Delta_\tau - \sigma_\tau$ を満たす. □

定義 7 と補題 3 から, $t \in [t_0, T_{\mathcal{C}})$ で $s(\mathbf{x}, t, \tau)$ は条件 (5.8) を満たす. この補題の逆を考えることで, 次の補題を得る.

補題 4. 時間 t, および 定義 7 で定義された仮想時間 τ と到達時間 $T_{\mathcal{C}} > 0$ を考える. このとき, ある時刻 $t < T_{\mathcal{C}}$ で $\tau = T_{\mathcal{C}}$ が成立する.

Proof. 補題 3 より, $t = \tilde{T}_{\mathcal{C}} < T_{\mathcal{C}}$ のとき, $\tau(t) \geq T_{\mathcal{C}} + \Delta_\tau - \sigma_\tau > T_{\mathcal{C}}$ を満たす σ_τ と $\tilde{T}_{\mathcal{C}}$ が存在する. そのため, $\sigma_\tau = \Delta_\tau$ のとき, $t = \tilde{T}_{\mathcal{C}} < T_{\mathcal{C}}$ で $\tau \geq T_{\mathcal{C}}$ を満たす. 加えて, 定義 5 より, τ の時間微分の値は常に正の値である. したがって, $\tau(t) = T_{\mathcal{C}}$ のとき, $t < \tilde{T}_{\mathcal{C}} < T_{\mathcal{C}}$ を満たす. □

5.3.3 ϵ-ZCBF への仮想時間の適用

ここでは, 前述した ϵ-ZCBF に仮想時間を導入する. 提案する仮想時間上での ϵ-ZCBF 制約を定義し, いくつかの注意点を述べる.

定理 2. (ϵ-ZCBF と時間軸変換) ϵ-ZCBF $b : \mathcal{X} \to \mathbb{R}$ と定数 ϱ_ϵ と ϵ, δ, 定義 7 で定義された $\rho(x)$ および到達時間 $T_{\mathcal{C}}$ に対して定義された仮想時間 τ を考える. 仮想時間 τ 上での ϵ-ZCBF 制約を以下のように定義する:

$$\sup_{\mathbf{u}\in U}[s(\mathbf{x}, t, \tau)\left(L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u}\right) + \varrho_\epsilon(b(\mathbf{x}) - \epsilon)] \geq 0. \tag{5.27}$$

加えて, ϵ-ZCBF と時間軸変換が与えられたとき, 不等式 (5.27) を満たす入力の集合を $\mathcal{U}_{\epsilon-\tau}(\mathbf{x}, t) = \{\mathbf{u} \in \mathcal{U} \mid s(\mathbf{x}, t, \tau)(L_f b(\mathbf{x}) + L_g b(\mathbf{x})\mathbf{u}) + \varrho_\epsilon(b(\mathbf{x}) - \epsilon) \geq 0\}$ とする. フィードバック制御器 $\mathbf{u}(\mathbf{x}, t) \in U_{\epsilon-\tau}$ を考える. このとき, 制御器 $\mathbf{u}(\mathbf{x}, t)$ はシステムの状態を元の時間軸 t 上で有限時間 $0 < T^*_{\mathcal{C}} < T_{\mathcal{C}}$ で集合 C_δ に遷移させる: $\mathbf{x}(t) \in C_\delta\,(t = T^*_{\mathcal{C}})$. 加えて, システムの状態は全ての時刻 $T^*_{\mathcal{C}} \leq t < T_{\mathcal{C}}$ で C_δ に留まる.

Proof. 補題 2 から, 制御器 $\mathbf{u}(\mathbf{x}, t)$ はシステムの状態を時間軸 τ 上で有限時間 $T_{\mathcal{C}}$ 以内に集合 C_δ に遷移させる: $\mathbf{x}(\tau) \in C_\delta\,(\tau = T_{\mathcal{C}})$. そのため, 補題 4 から, $\mathbf{x}(\tau) \in C_\delta\,(\tau = T_{\mathcal{C}})$ のとき, $\mathbf{x}(t) \in C_\delta$ を満たす時刻 $t = T^*_{\mathcal{C}} < T_{\mathcal{C}}$, $\tau(t) = T_{\mathcal{C}}$ が存在する. 加えて, 補題 2 から, システムの状態は任意の時刻 $T_{\mathcal{C}} \leq \tau < T_{\mathcal{C}} + \Delta_\tau$ および $T^*_{\mathcal{C}} \leq t < T_{\mathcal{C}}$ において C_δ に留まる. □

注意 1. 提案する制御器は, $\rho(\mathbf{x})$ の値に応じて ϵ-ZCBF 制約を緩和する; $\rho(\mathbf{x})$ の値が β より小さい場合, τ の速度が遅くなり, 制約（5.27）が緩和される. ただし, t が時間 T に近づくと, $s_2(t, \tau)$ が大きくなり, τ の速度が速くなる. これにより, τ が t に追いつくため, 有限時間 $T_{\mathcal{C}}$ 以内にシステムを集合 C_δ に遷移させる.

注意 2. 仮想時間は時間区間 $[t_0, T_{\mathcal{C}})$ で定義されているため, 定理 2 の制御器も $[t_0, T_{\mathcal{C}})$ で定義される. したがって, 時間 T の以降の制御には, 制御器を変更または修正する必要がある. システムを C_δ に留まるように ϵ-ZCBF を用いた制御器を用いて制御する場合, 時間 $T^*_{\mathcal{C}}$ で仮想時間の速度を $s(\mathbf{x}, t, \tau) = 1$ に固定することで実現できる.

5.3.4 STL 命題への提案手法の適用

提案手法を STL で記述されたタスクに適用する. まず, 与えられた STL 命題に対応する時間オートマトンに受理されるパスの振る舞いを次の STL 命題によって表現する.

$$\bigwedge_i \mathbf{F}_{[t_i,t_{i+1}]}\mathbf{G}\psi_i \wedge \mathbf{G}_{[t_i,t_{i+1}]}\phi_i, \tag{5.28}$$

ここで, $t_i < t_{i+1}$, ϕ_i と ψ_i は様相作用素を含まない STL 命題とする. この命題は [50] で用いられている命題と同様の形式をしている. 時間区間 $[t_i, t_{i+1}]$ に対するそれぞれの命題は時間オートマトンのロケーション間の遷移に対応している. [50] で用いられている命題との違いは, 本章で用いる命題は時間オートマトンから生成されているため, 各命題の様相作用素に関する時間区間が自動的に決定する点である. また, ϕ_i は安全性に関わる命題, ψ_i は達成するべきロケーション間の遷移に関わる命題と考えることができる. ここで, 次の問題を考える:

問題 1. 各時間区間 $[t_i, t_{i+1}]$ で, システム (5.1) が命題 (5.28) を満たすように制御する.

加えて, 本章では ϕ_i に関する制約は ψ_i 制約よりも優先して常に満たされるべきであると仮定する.

問題 1 から, 下記の CBF 制約を考える:

- ψ_i に対する ϵ-ZCBF 制約,
- ϕ_i に対する CBF 制約.

最適化問題が実行不可能になるリスクを低減するために, これらの CBF 制約を仮想時間上で実装する. CBF 制約を満たす制御入力を計算するための最適化問題は下記の様に定式化される:

$$\min_{\mathbf{u}} \quad ||\mathbf{u}||^2 \tag{5.29}$$

s.t.

$$s_i(\mathbf{x},t,\tau_i)(L_f\rho_{\psi_i}(\mathbf{x}) + L_g\rho_{\psi_i}(\mathbf{x})\mathbf{u}) + \varrho_{\epsilon_i}(\rho_{\psi_i}(\mathbf{x}) - \epsilon_i) \geq 0, \tag{5.30}$$

$$s_i(\mathbf{x},t,\tau_i)(L_f\rho_{\phi_i}(\mathbf{x}) + L_g\rho_{\phi_i}(\mathbf{x})\mathbf{u}) + \varrho_{\phi_i}\rho_{\phi_i}(\mathbf{x}) \geq 0, \tag{5.31}$$

ここで, $\varrho_{\phi_i} > 0$ は定数:

$$\tau_i(t_i) = t_i,\ s_i(\mathbf{x},t,\tau_i) = s_{i1}(\mathbf{x}) \cdot s_{i2}(t,\tau_i), \tag{5.32}$$

$$s_{i1}(\mathbf{x}) = \max\left(-\ln\left(\frac{1}{\beta_i}\rho_{\phi_i}(\mathbf{x})\right), 0\right) + 1, \tag{5.33}$$

$$s_{i2}(t,\tau) = \frac{t_{i+1} - t}{t_{i+1} - \tau_i + \Delta_{\tau_i}}. \tag{5.34}$$

仮想時間によって ρ_{ϕ_i} の値が β より小さくなると $s(\mathbf{x}, t, \tau)$ が大きくなり, ϵ-ZCBF 制約は時間の観点から緩和される. これは, もし ϕ_i に関する制約が違反されそうになると, ψ_i に関する制約が緩和されることを意味している. ρ_{ϕ_i} に関する CBF 制約から, $\forall t \in [t_i, t_{i+1}], \rho_{\phi_i}(\mathbf{x}) > 0$ が保証され, これにより $\forall t \in [t_i, t_{i+1}], s(\mathbf{x}) < \infty$ が成り立つ.

ψ_i が満たされ, 時刻 t が t_{i+1} になったとき, 次の時間区間の CBF 制約を用いた制御器に切り替える. これを逐次的に繰り返し, 時間オートマトンのロケーション間の遷移が達成される. 受理状態への遷移が達成された際, 元のタスクが達成されることで, 問題 1 が解決される.

5.4 数値実験

ここでは, 提案手法をアダプティブクルーズコントロールに対する 1 つの時間間隔を持つタスクに適用する. 本章では, 全ての計算は, 3.6GHz Intel Corei9 で実行した. 加えて, 全ての最適化問題は MATLAB によって実装され, シミュレーションは Simulink [55] で実行した.

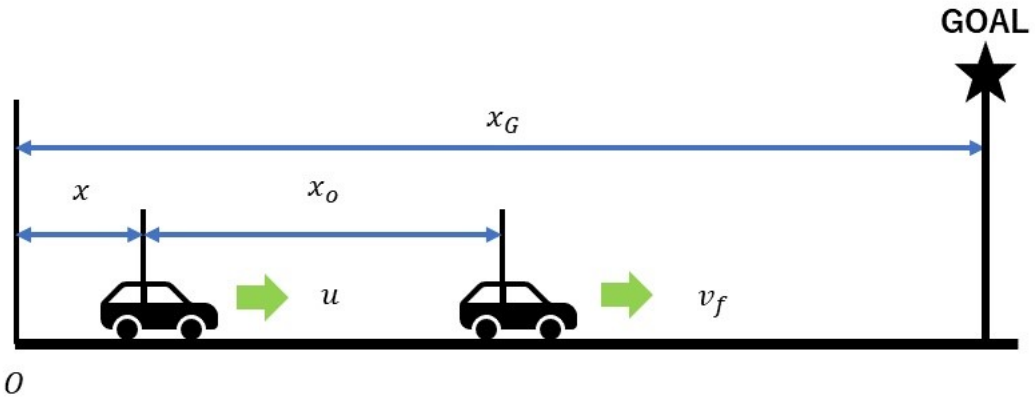

Fig. 5.3: Model of an adaptive cruise control

5.4.1 問題設定

1 次元のアダプティブクルーズコントロールを考える (Fig. 5.3). 対象とするシステムとして, 次のダイナミクスを考える:

$$\dot{x} = u, \dot{x}_o = v_f - u, \tag{5.35}$$

ここで, u は入力速度, x は制御する車の位置, x_o は前方の車との距離, v_f は前方の車の速度とする. 前方の車の速度は観測可能と仮定する.

タスクとして “いつか, 制御する車は時間 T までに目標位置 x_G に到達し, 常に前方の車までの距離を 0.5 以上に保つ” という命題を考える. この命題を以下の STL 命題として定義する:

$$\mathbf{F}_{[0,T]}\mathbf{G}\psi_A \wedge \mathbf{G}_{[0,T]}\phi_B, \tag{5.36}$$

$$\rho_{\psi_A} = x - x_G,\ \rho_{\phi_B} = x_o - 0.5. \tag{5.37}$$

このタスクでは, 入力速度の上限は前方の車までの距離の条件によって制限され, 入力速度の下限は指定された時間までに目標位置に到達するための条件によって制限される. そのため, 時間に関する制約により入力速度の下限が上限を超える可能性があり, その場合, 実行可能な入力が消失する. 提案手法により, この制御入力が消失するリスクを低減する.

仮想時間による入力の消失のリスクの低減と提案手法で STL タスクが達成されることを確認するため, 次の状況に対してシミュレーションを行った.

- CBF パラメーター ϱ_ϵ, ϱ_ϕ を仮想時間ありと仮想時間なしの場合に対して変更する
- 前の車の動作を変更する

20 組の CBF パラメーターを用意し, 前方の車の 3 つの動作パターンに対してシミュレーションを行った. また, 最適化問題が実行不可能になった場合, シミュレーションを停止した. この際に使用した CBF パラメーターは, 動作パターン 1 でタスクを達成し, 他の動作パターンの初期状態で最適化問題の CBF 制約を満たすようにランダムに生成した. その他のパラメーターは, $x\,(0) = 0$, $x_o(0) = 2$, $T = 10$, $x_G = 15$, $\delta = 0.1$, $\Delta_\tau = 0.1$ および $\beta = 2$ とした. また ϵ は, ϱ_ϵ, T および δ に対して, 補題 2 の不等式 $e^{-\varrho_\epsilon T}(b\,(x_0) - \epsilon) + \epsilon = \delta$, $\epsilon \geq \delta$ を満たすように決定した.

5.4.2 結果・考察

Fig.5.4 は, 前方の車の 3 つの動作パターンを示してる. 3 つの動作パターンは,「前方の車は常に速く進む」, 「前方の車は最初は速く, 次にゆっくりと進み, 最後に速く進む」,「前方の車は最初はゆっくりと進み, 次に速く進む」の 3 パターンとした. 表 5.1 は, シミュレーションしたパラメーターの組の数に対するタスクの命題を達成したパラメーターの組の数を示している. パターン 1 では, 命題を達成したパラメーターの組の数は、シミュレーションしたパラメーターの組の数と同じ 20 である. パターン 2 では, 仮想時間なしでは命題を達成したパラメーターの組が存在しなかったが, 仮想時間を用いた場合, 15 組のパラメーターの組が命題を達成した. また, パターン 3 では, 仮想時間を用いることで, 命題を達成するパラメーターの組の数が増加した. これより, 提案した制御器を用いてシステムがタスクである STL 命題を達成したことが確認できる. さらに, 命題を達成するパラメーターの組の数が仮想時間を導入することで増加したことから, 仮想時間が制御入力の消失のリスクを低減していることがわかる. 加えて, ϵ-ZCBF はパラメーターとロバストネス関数が補題 2 を満たすよ

うに, ロバストネス関数を使用していくつかのパラメーターを設計することで容易に定義できる.

さらに, Fig.5.5 は, パターン 3 の時の 1 つのパラメーターの組に対するロバストネス関数 ρ_{ψ_A} と ρ_{ϕ_B} の軌道を示している. ρ_{ϕ_B} は常に正であり, ρ_{ψ_A} は $T = 10$ の前に正になるため, ϵ-ZCBF を使用するコントローラーが命題を達成していることがわかる. また, 一時的に ρ_{ϕ_B} の値が β より小さい期間が存在し, その際には仮想時間の進行が遅くなり ρ_{ψ_A} の軌道は仮想時間がない場合よりも遅く進行していることがわかる. しかし, 仮想時間の時間を早送りする項の影響による部分的な高速化によって, 仮想時間が元の時間に追いつき, 命題を達成している.

	w/o virtual time	w/ virtual time
Behavior 1	20/20	20/20
Behavior 2	0/20	15/20
Behavior 3	15/20	20/20

Table 5.1: Solved parameters ratio in the simulations

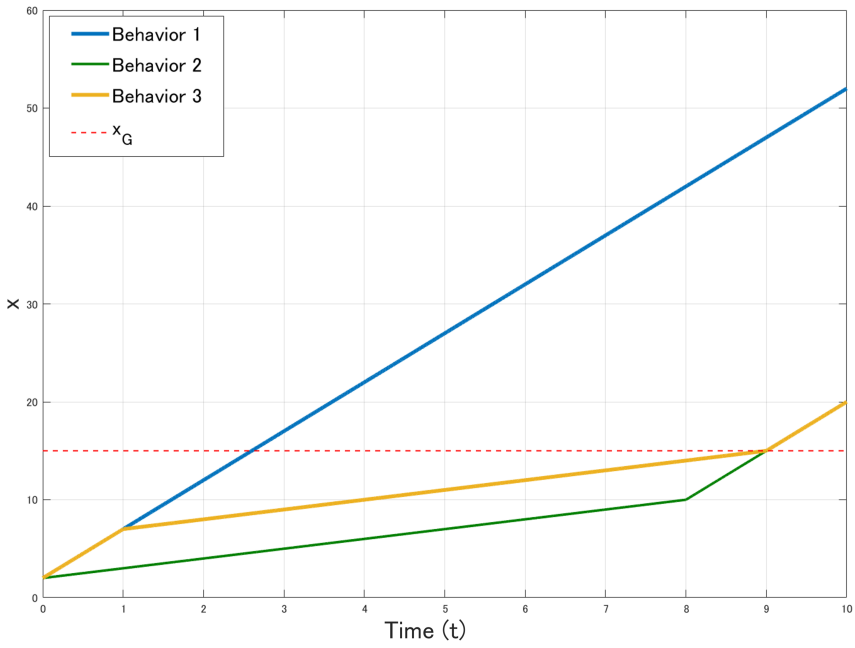

Fig. 5.4: Three behaviors of car in front: 1) fast, 2) slow at first, and 3) fast at first and last.

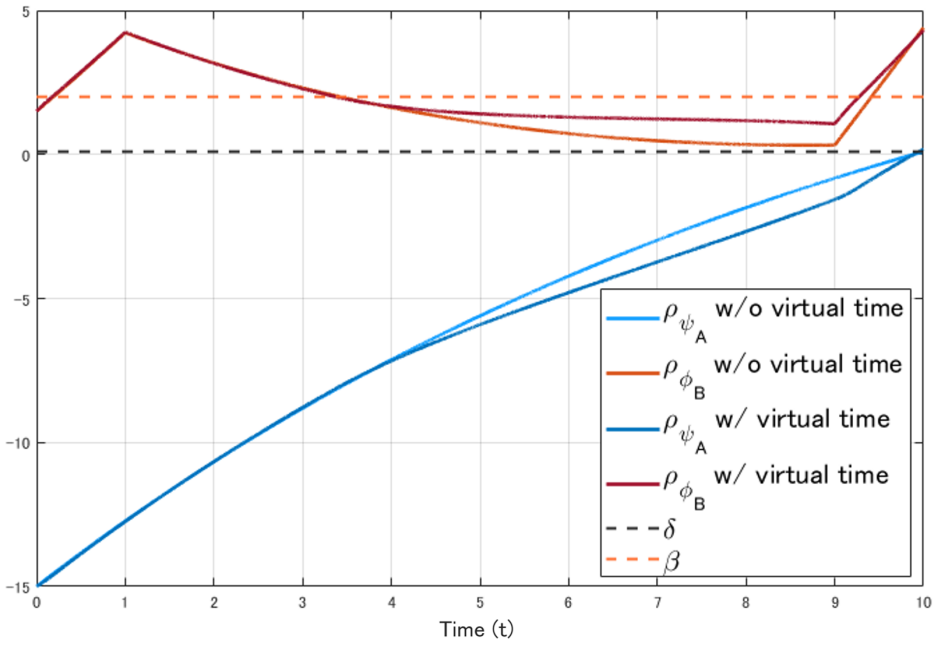

Fig. 5.5: The Trajectories of robustness functions of ψ_A and ϕ_B with and without virtual time in behavior 3. We can see that ψ_A is a positive value at time T and ϕ_B is always positive.

5.5 まとめ

本章では, 時間制約をもつ様相作用素を含む STL 命題を達成するための制御手法を提案した. まず, 指定された時間制約を守りながら状態を目的の集合に到達させるために ϵ-ZCBF を定義した. 次に, 制御入力が消失するリスクを軽減するため, 制約条件を緩和する仮想時間を導入した. その後, 最後に, 移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示した.

本章の数値実験では, パス上に存在するロケーション間の遷移が 1 つのタスクを対象としている. そのため, 複数のロケーション間の遷移がある STL 命題で表現されたタスクでの有効性検証は今後の課題と考えられる.

第6章　信号時相論理に対する確率的動作プリミティブの再配合

6.1　はじめに

タスクを達成するために複数のシステムが行動する際, 通信などによってお互いの状態を共有することができない場合がある. 特に, ロボットが人間と同時に行動する場合には, ロボットが人間の行動を推定し, 場合によっては行動を変更する必要がある.

これを踏まえ, 本章ではタスクを達成するための複数のシステムの行動が与えられた際に, 他のシステムの行動に合わせて自身の行動を修正するための手法を提案する. 時相論理で表現されたタスクを達成するための複数のシステムに対するタスクの配分手法に関しては様々な手法 [56,57] が提案されており, 本章では, タスクの配分方法については既存の手法を用いて高レベル制御器内で実現されていると仮定する.

以降では, まず提案手法で用いる確率的動作プリミティブについて説明する. 次に, 生成された確率的動作プリミティブを用いて, 他のシステムの状態が属している確率的動作プリミティブをカルマンフィルタを用いて推定する手法を提案する. その後, 推定された確率的動作プリミティブを基に, 自身の確率的動作プリミティブを更新することで, 他のシステムの行動を考慮して与えられた命題を達成する軌道を生成する方法を提案する. 最後に, 移動ロボットに対する数値シミュレーションにて, 提案手法の有効性を示す.

6.2　確率的動作プリミティブ

確率的動作プリミティブ (ProMP) はロボットシステムの軌道上での確率分布を表現する手法である [58,59]. ProMP を用いて, ロボットシステムの軌道 $\mathbf{x}(t)$ は以下のように表現できる [59]:

$$\mathbf{x}(t) = \mathbf{\Phi}(t)\mathbf{w} + \mathbf{n}_{\mathbf{x}}, \tag{6.1}$$

ここで, $\mathbf{\Phi} \in \mathbb{R}^{n_{\mathbf{x}} \times n_{\mathbf{w}} \cdot n_{\mathbf{x}}}$ は基底関数で構成された行列, $\mathbf{w} = [w_1^{\mathsf{T}}, \ldots, w_{n_{\mathbf{x}}}^{\mathsf{T}}] \in \mathbb{R}^{n_{\mathbf{x}} \cdot n_{\mathbf{w}}}$, $w_i \in \mathbb{R}^{n_{\mathbf{w}}}$ は各状態に対する重みベクトル, $\mathbf{n}_{\mathbf{x}} \sim \mathcal{N}(0_{n_{\mathbf{x}}}, \Sigma_{\mathbf{x}})$ は状態に関するノイズとする. $\mathbf{\Phi}$ は以下

の式の様に定義される:

$$\mathbf{\Phi}(t) = \begin{pmatrix} \Phi_1(t) & \cdots & 0_{1\times n_{\mathbf{w}}} \\ \vdots & \ddots & \vdots \\ 0_{1\times n_{\mathbf{w}}} & \ddots & \Phi_{n_{\mathbf{x}}}(t) \end{pmatrix},$$

ここで, $\Phi_i(t)$ は時刻を引数とする基底関数の集合で構成されたベクトルである. 本章では, 基底関数として動径基底関数を用いる. 加えて, 重みベクトル w_i について以下を仮定する:

$$p(w_i|\mu_{w_i}, \Sigma_{w_i}) = \mathcal{N}(\mu_{w_i}, \Sigma_{w_i})$$

ここで, μ_{w_i}, Σ_{w_i} は ProMP を表現するためのパラメーターである. また, $\mathbf{w}$ の分布は以下のように表現できる:

$$\mathcal{N}\left(\begin{pmatrix} w_1 \\ \vdots \\ w_{n_{\mathbf{x}}} \end{pmatrix} \middle| \begin{pmatrix} \mu_{w_1} \\ \vdots \\ \mu_{w_{n_{\mathbf{x}}}} \end{pmatrix}, \begin{pmatrix} \Sigma_{w_1,w_1} & \cdots & \Sigma_{w_1,w_{n_{\mathbf{x}}}} \\ \vdots & \ddots & \vdots \\ \Sigma_{w_{n_{\mathbf{x}}},w_1} & \cdots & \Sigma_{w_{n_{\mathbf{x}}},w_{n_{\mathbf{x}}}} \end{pmatrix}\right) \tag{6.2}$$

ProMP はシステムの軌道データから最尤推定を用いて学習することができる. 本章の数値実験では, 参考文献 [59] で提案されている手法を用いて ProMP を学習している.

6.2.1 確率的動作プリミティブの配合

複数の ProMP が与えられたとき, それらを配合することによって新しい軌道分布を生成することができる [58]. ここでは, M 個の ProMP を配合することを考える. また, 先行研究 [58,60] と異なり, ProMP の重みベクトルの分布を合成することで新しい ProMP を定義する. 新しい軌道分布の特性として $\mathbf{x}(t) = \mathbf{\Phi}(t)\mathbf{w}^* + \mathbf{n}_{\mathbf{x}}$, $p(\mathbf{w}^*) = \prod_i p^{\gamma_i}(\mathbf{w}_i), \gamma_i \in [0,\, 1]$ を仮定する. このとき, 新しい軌道分布の重みベクトルの分布 $p(\mathbf{w}^*)$ は次式となる.

$$\mathbf{w}^* \sim \mathcal{N}\left(\mu_{\mathbf{w}^*}, \Sigma_{\mathbf{w}^*}\right), \tag{6.3}$$

$$\Sigma_{\mathbf{w}^*} = \left(\sum_i \frac{1}{\gamma_i}\Sigma_{\mathbf{w}_i}\right)^{-1},\ \mu_{\mathbf{w}^*} = \Sigma_{\mathbf{w}^*}\left(\sum_i \left(\frac{1}{\gamma_i}\Sigma_{\mathbf{w}_i}\right)^{-1}\mu_{\mathbf{w}_i}\right) = \mathbf{T}_{\mathbf{w}}\mu_{\mathbf{w}}, \tag{6.4}$$

ここで, $\mathbf{T}_{\mathbf{w}^*} = [\Sigma_{\mathbf{w}^*}(\frac{1}{\gamma_1}\Sigma_{\mathbf{w}_1})^{-1}, \ldots, \Sigma_{\mathbf{w}^*}(\frac{1}{\gamma_M}\Sigma_{\mathbf{w}_M})^{-1}]$, $\mu_{\mathbf{w}} = [\mu_{\mathbf{w}_1}^{\mathsf{T}}, \ldots, \mu_{\mathbf{w}_M}^{\mathsf{T}}]^{\mathsf{T}}$.

特定の γ_i を 1 に近い値, その他の γ_i を 0 に近い値にすることで, 配合された ProMP は特定の ProMP に近い分布になる. 加えて, γ_i を時変にすることで, 配合された ProMP を時間に沿って変化させることも可能である.

6.3 無香料カルマンフィルタ

無香料カルマンフィルタ (Uncented Kalman Filter, UKF) は非線形システムの状態の分布をサンプル点を用いて推定することで, 高い精度で分布を近似する手法である [61] [62].

6.3.1 アルゴリズム

UKF のアルゴリズムについて述べる [62]. 対象のシステムとして以下の式で表現される離散時間非線形システムを考える;

$$\begin{aligned}\mathbf{x}(t+1) &= F(\mathbf{x}(t),\, \mathbf{n}_x(t)) \\ \mathbf{y}(t) &= H(\mathbf{x}(t),\, \mathbf{n}_y(t)),\end{aligned}$$

ここで, F, H は遷移関数と出力関数, $\mathbf{n}_*$ はそれぞれ $\mathbf{x}$, $\mathbf{y}$ に関するノイズである.

準備

前述したシステムに対して, 補助変数 $\mathbf{x}_{UKF} \in \mathbb{R}^{n_{\mathrm{UKF}}}$ を考える;

$$\mathbf{x}_{UKF}(t-1) = \begin{bmatrix}\mathbf{x}(t-1) \\ \mathbf{n}_x(t-1) \\ \mathbf{n}_y(t-1)\end{bmatrix},$$

加えて, 状態とノイズに関する共分散行列を用いて, 補助変数 $\mathbf{X}_{UKF}$ の共分散行列を定義する;

$$\Sigma_{UKF} = \begin{bmatrix}\Sigma_{\mathbf{x}} & 0 & 0 \\ 0 & \Sigma_{\mathbf{n}_x} & 0 \\ 0 & 0 & \Sigma_{\mathbf{n}_y}\end{bmatrix}$$

ただし, ノイズの平均値はゼロと仮定している.

シグマポイントの生成

共分散行列を用いて, $2n_{\mathrm{UKF}}+1$ 個のシグマポイントを以下のように生成する;

$$\mathbf{X}^i_{UKF}(t-1) = \begin{cases}\mathbf{x}_{UKF}(t-1), & i = 0 \\ \mathbf{x}_{UKF}(t-1) + \kappa_1 \mathcal{S}_i, & i = 1, \ldots, n_{\mathrm{UKF}} \\ \mathbf{x}_{UKF}(t-1) - \kappa_1 \mathcal{S}_i, & i = n_{\mathrm{UKF}}+1, \ldots, 2n_{\mathrm{UKF}}\end{cases}$$

ここで, $\mathcal{S}_i$ は行列 $\mathcal{S} = \sqrt{\Sigma_{UKF}}$ の i 列目の列ベクトルである. また, κ_1 はスケーリングパラメーターである [63]. このシグマポイントを状態とノイズに関する部分に以下のように分割する;

$$\mathbf{X}^i_{UKF}(t-1) = \begin{bmatrix}\mathbf{X}^i_{\mathbf{x}}(t-1) \\ \mathbf{X}^i_{\mathbf{n}_x}(t-1) \\ \mathbf{X}^i_{\mathbf{n}_y}(t-1)\end{bmatrix}.$$

シグマポイントの更新

シグマポイントそれぞれについて, 遷移関数を用いて状態の更新を行う;

$$\mathbf{X}^i_{\mathbf{x},t/t-1} = F(\mathbf{X}^i_{\mathbf{x}}(t-1), \mathbf{X}^i_{\mathbf{n}_x}(t-1)).$$

遷移した状態を用いて事前分布を計算する:

$$\begin{aligned}
\hat{\mathbf{x}}^-_t &= \sum_{i=0}^{2n_{\mathrm{UKF}}} \lambda^i_m \mathbf{X}^i_{\mathbf{x},t/t-1}, \\
\Sigma^-_{\mathbf{x}_t} &= \sum_{i=0}^{2n_{\mathrm{UKF}}} \lambda^i_c (\mathbf{X}^i_{\mathbf{x},t/t-1} - \mathbf{x}^-_t)(\mathbf{X}^i_{\mathbf{x},t/t-1} - \mathbf{x}^-_t)^{\mathsf{T}},
\end{aligned}$$

ここで, 各重み λ^i_m, λ^i_c は以下のように定義される;

$$\begin{aligned}
\lambda^0_m &= \frac{\kappa_2}{n_{\mathrm{UKF}} + \kappa_2}, && i = 0 \\
\lambda^0_c &= \frac{\kappa_2}{n_{\mathrm{UKF}} + \kappa_2} + (1 - \kappa_3), && i = 0 \\
\lambda^i_m &= \lambda^i_c = \frac{\kappa_2}{2(n_{\mathrm{UKF}} + \kappa_2)}, && i = 1, \ldots, 2n_{\mathrm{UKF}}
\end{aligned}$$

ここで, κ_2, κ_3 は調整パラメーターである.

状態の推定

シグマポイントの更新と同様に, シグマポイントに対応する出力, 平均と共分散を計算する;

$$\begin{aligned}
\mathbf{Y}^i_{t/t-1} &= H(\mathbf{X}^i_{\mathbf{x},t/t-1}, \mathbf{X}^i_{\mathbf{n}_y}(t-1)), \\
\hat{\mathbf{y}}^-_t &= \sum_{i=0}^{2n_{\mathrm{UKF}}} \lambda^i_m \mathbf{Y}^i_{t/t-1}, \\
\Sigma^-_{\mathbf{y}_t} &= \sum_{i=0}^{2n_{\mathrm{UKF}}} \lambda^i_c (\mathbf{Y}^i_{t/t-1} - \hat{\mathbf{y}}^-_t)(\mathbf{Y}^i_{t/t-1} - \hat{\mathbf{y}}^-_t)^{\mathsf{T}}, \\
\Sigma_{\mathbf{x}_t,\mathbf{y}_t} &= \sum_{i=0}^{2n_{\mathrm{UKF}}} \lambda^i_c (\mathbf{X}^i_{\mathbf{x},t/t-1} - \mathbf{x}^-_t)(\mathbf{Y}^i_{t/t-1} - \hat{\mathbf{y}}^-_t)^{\mathsf{T}}.
\end{aligned}$$

このとき, カルマンゲインは次式で計算される;

$$K_t = \Sigma_{\mathbf{x}_t,\mathbf{y}_t} (\Sigma^-_{\mathbf{y}_t})^{-1}.$$

また, 推定値と共分散行列は次式で計算される;

$$\begin{aligned}
\hat{\mathbf{x}}(t) &= \hat{\mathbf{x}}^-_t + K_t(\mathbf{Y}^i_{t/t-1} - \hat{\mathbf{y}}^-_t), \\
\Sigma_{\mathbf{x}_t} &= \Sigma^-_{\mathbf{x}_t} - K_t \Sigma^-_{\mathbf{y}_t} K_t^{\mathsf{T}}.
\end{aligned}$$

6.4 観測情報に基づく確率的動作プリミティブの推定

複数のシステムで行動する際, 他のシステムの行動を考慮して行動決定を行う必要がある. そのため, システムが想定している行動の内, どの行動を行っているのかを観測情報から推定する. ここでは, システムの行動に関する ProMP が与えられていると仮定し, 観測情報を用いてカルマンフィルタによって現在の行動が属している ProMP を逐次的に推定する手法を提案する.

6.4.1 配合された確率的動作プリミティブに対する無香料カルマンフィルタ

カルマンフィルタを構築する準備として, 推定する ProMP の配合に関するパラメーター $\gamma = [\gamma_1, \ldots, \gamma_M]$, $\gamma_i \in [0, 1]$, $(i = 1, \ldots, M)$ と ProMP で表現された軌道分布上での時刻 $\tau \geq 0$ の推定値について, 以下のシステムを仮定する.

$$\gamma_i(t+1) = \gamma_i(t) + \mathbf{n}_\gamma, \qquad \mathbf{n}_\gamma \sim \mathcal{N}(O, \Sigma_{\mathbf{n}_\gamma}), \tag{6.5}$$

$$\tau(t+1) = \tau(t) + d\tau + \mathbf{n}_\tau, \qquad \mathbf{n}_\tau \sim \mathcal{N}(O, \Sigma_{\mathbf{n}_\tau}), \tag{6.6}$$

$$\mathbf{y}(t) = C(\tau)\mu^*_{\mathbf{w}}(\gamma(t)) + \mathbf{n}_\mathbf{y}, \qquad \mathbf{n}_\mathbf{y} \sim \mathcal{N}(O, \Sigma_{\mathbf{n}_\mathbf{y}}). \tag{6.7}$$

ここで, $d\tau > 0$ は定数, $\mathbf{n}_*$ はノイズ, $\mu^*_{\mathbf{w}}$ は γ_i から式 (6.4) で計算される配合された ProMP の重みベクトルの期待値, $C(\tau)$ は τ から計算される基底関数行列を含む行列とする. 加えて, パラメーター γ_i には, $\sum_i \gamma_i = 1$ の拘束を仮定する. このシステムは, 推定するパラメーター γ_i, τ に対して非線形なシステムであるため, UKF を用いて観測情報 $\mathbf{y}$ から, パラメーター $\hat{\gamma}_i$, $\hat{\tau}$ を推定する.

γ の制約条件として, $\sum_{i=1}^{M} \gamma_i = 1$ を仮定しているため, 制約条件を考慮して推定を行う必要がある. ここでは, $\gamma_i\,(i = 1, \ldots, M)$ から γ_M を省いた $\gamma_i\,(i = 1, \ldots, M-1)$ に対して, 式 (6.7) のシステムを構築し, $\sum_{i=1}^{M-1} \gamma_i \leq 1 - \epsilon_\gamma$ を制約条件に加えたシステム上で推定を行う. ここで, $\epsilon_\gamma > 0$ は十分小さな定数である. 加えて, シグマポイントを生成する際に制約条件を満たす領域に対して射影を行う. 具体的には以下の手順によって, シグマポイントを射影する.

1. γ_i, τ が制約を満たす場合, 手順を終了
2. γ_i が制約を違反している場合:
   (a) $\gamma_i < \epsilon_\gamma$ ならば, $\gamma'_i = \epsilon_\gamma$.
   (b) $\gamma_i > 1 - (M-1)\epsilon_\gamma$ ならば, $\gamma'_i = 1 - (M-1)\epsilon_\gamma$.
   (c) $\sum_{i=1}^{M-1} \gamma_i > 1 - \epsilon_\gamma$ ならば, $\mathbf{v}_\gamma^\mathsf{T}\gamma + c_\gamma = 0$ へ射影;

$$\gamma' = \gamma - \frac{\mathbf{v}_\gamma \mathbf{v}_\gamma^\mathsf{T}}{||\mathbf{v}_\gamma||_2^2}(\gamma - c_\gamma \mathbf{v}_\gamma),$$

ここで, $\gamma = [\gamma_1, \ldots, \gamma_{M-1}]^\mathsf{T}$,

$$\mathbf{v}_\gamma = \frac{1}{\sqrt{M-1}}[1, \ldots, 1]^\mathsf{T},\ c_\gamma = -\frac{1}{\sqrt{M-1}}(1-\epsilon_\gamma).$$

3. τ が制約を違反している場合, $\tau = 0$.

6.4.2 推定された確率的動作プリミティブを用いた判別

UKF によって推定した ProMP の配合に関するパラメーターを用いて, システムの行動が属している ProMP を判別する. ここで, $\bar{\Sigma}_\mathbf{w}$, $\bar{\mu}_\mathbf{w}$ を判別した ProMP とする. 判別方法としては, 推定された最大の $\hat{\gamma}_i$ に対応する ProMP がそのシステムの行動が属する ProMP とする方法や推定した配合された ProMP 自体をシステムの行動が属する ProMP とする方法がある.

一方で, システムの行動が配合に用いたどの ProMP にも属していない場合がある. そのため, ProMP に関するパラメーターを推定した後, 判別した ProMP に対して外れ値であるかどうかを判別することで, システムの状態がどの ProMP にも属していない場合を判別する. 外れ値 $\mathbf{d}_{Pro}$ は次式で計算した;

$$\mathbf{d}_{Pro} = (y(t) - C(t)\bar{\mu}_\mathbf{w}(t))^\mathsf{T} \left(C(t)\bar{\Sigma}_\mathbf{w}C^\mathsf{T}(t)\right)^{-1} (y(t) - C(t)\bar{\mu}_\mathbf{w}(t)) \tag{6.8}$$

6.5 観測情報による確率的動作プリミティブの再配合

推定した他のシステムの確率的動作プリミティブを用いて, STL 命題を達成するための軌道を生成するため行動計画手法を定式化する. 複数のシステムを用いてタスクを達成する場合, システム全体でタスクを達成するために, 他のシステムの行動を考慮して行動計画を行う必要がある. そのため, 推定した ProMP からシステムの軌道 $\mathbf{X}_c(1:T)$ を生成し, 他のシステムの行動を獲得する. 生成されたシステムの軌道 $\mathbf{X}_c(1:T)$ を用いて, STL 命題を満たす軌道を生成するために ProMP の配合に関するパラメーターを最適化によって更新する. ProMP の配合に関するパラメーターの最適化問題は以下のように定式化できる:

$$\min_{\gamma, \mathbf{a}} -\rho_\varphi(\mathbf{x}, \mathbf{X}_c) + L(\gamma, \mathbf{a}) \tag{6.9}$$

s.t.

$$\mathbf{x}(t) = \Phi(\tau_\mathbf{x}(t))\mu^*_\mathbf{w}(\gamma), \tag{6.10}$$

$$\mu^*_\mathbf{w}(\gamma) = \mathbf{T}_w(\gamma)\mu_\mathbf{w}(\gamma), \tag{6.11}$$

$$\rho_\varphi(\mathbf{x}, \mathbf{X}_c) \geq \epsilon_\varphi, \tag{6.12}$$

$$\tau_\mathbf{x}(t) = \sum_{n=0}^{L} a_n t^n, \tag{6.13}$$

$$\frac{d}{dt}\tau_\mathbf{x}(t) \geq 0. \tag{6.14}$$

ここで, $\mathbf{a} = [a_1, \ldots, a_L]^\mathsf{T} \in \mathbb{R}_+^L$ は時間関数 $\tau_\mathbf{x} : \mathcal{T} \to \mathbb{R}_+$ の係数, ρ_φ は STL 命題 φ に対応するロバストネス関数, $L(\gamma, \mathbf{a})$ はその他の目的関数である. 時間関数 $\tau_\mathbf{x}$ を導入することによって, システムの軌道の自由度を高めることが可能となっている.

この最適化問題の解であるパラメーター γ, $\mathbf{a}$ から, 式 (6.10),(6.13) を用いて, STL 命題を満たすシステムの軌道を計算することができる.

6.6 数値実験

ここでは, 提案手法を 2 台の移動ロボットのタスクに適用する. 本章での全ての計算は, 3.6GHz Intel Core i9 で実行した. 全ての最適化問題は MATLAB 上で CasADi [42] によって実装し, ソルバーとして Ipopt [45] を用いた.

6.6.1 問題設定

2 台の移動ロボットのタスクとして, Fig.6.1 に示す環境において次の命題を考える: "2 台のロボットの内, 領域 1, 2 を少なくとも 1 台が通過し, その後領域 3 に移動する". この命題を以下の STL 命題として定義する:

$$
\begin{aligned}
\varphi &= \varphi_1 \wedge \varphi_2 \\
\varphi_1 &= \mathbf{F}_{[0,T]}(\varphi_{1,1} \Rightarrow \mathbf{F}\varphi_{3,1}) \vee \mathbf{F}_{[0,T]}(\varphi_{2,1} \Rightarrow \mathbf{F}\varphi_{3,1}) \\
\varphi_2 &= \mathbf{F}_{[0,T]}(\varphi_{1,2} \Rightarrow \mathbf{F}\varphi_{3,2}) \vee \mathbf{F}_{[0,T]}(\varphi_{2,2} \Rightarrow \mathbf{F}\varphi_{3,2})
\end{aligned}
$$

ここで, $\varphi_{i,j}$ は命題 "領域 i にロボット j が存在する" を表わしている. ロボットのダイナミクスとして独立 2 輪型モデルを考え, システム全体の状態を $\mathbf{x} = [(\mathbf{x}^1)^\mathsf{T}, (\mathbf{x}^2)^\mathsf{T}]^\mathsf{T}$, 各ロボットの状態を $\mathbf{x}^j = [x_{p_x}^j, x_{p_y}^j, x_\theta^j]^\mathsf{T} \in \mathbb{R}^3$ とし, 入力を $\mathbf{u} = [u_v, u_w]^\mathsf{T}$, $u_v \in [0,\, 1]$, $u_w \in [-1,\, 1]$ とする. それぞれの命題に対して, 次のロバストネス関数を定義する:

$$
\begin{aligned}
\rho_{\varphi_{1,j}}(\mathbf{x}) &= \min(x_{p_x}^j - 1.1, -x_{p_x}^j + 1.3, x_{p_y}^j - 1, -x_{p_y}^j + 1.2), \\
\rho_{\varphi_{2,j}}(\mathbf{x}) &= \min(x_{p_x}^j - 1, -x_{p_x}^j + 1.2, x_{p_y}^j, -x_{p_y}^j + 0.2), \\
\rho_{\varphi_{3,j}}(\mathbf{x}) &= \min(x_{p_x}^j - 2, -x_{p_x}^j + 2.2, x_{p_y}^j - 1, -x_{p_y}^j + 1.2).
\end{aligned}
$$

また, ロボットの初期値を $\mathbf{x}^1 = [0, 0, 0]^\mathsf{T}$, $\mathbf{x}^2 = [0, 1, 0]^\mathsf{T}$, 命題に関する時間制約を $T = 8$ とした. ProMP の学習のために, 個々のロボットについて初期値にノイズを加えた状態から以下の行動を達成する軌道を式 (4.5) と同様な最適化問題を解くことによって生成する. 本実験では, 各ロボットの行動はタスクの命題を基に手動でタスクの配分を行った. 加えて, タスクの中に現れるロボットの状態は各ロボットの平面座標 $x_{p_x}^j$, $x_{p_y}^j$ のみであるため, 各ロボット毎に $x_{p_x}^j$, $x_{p_y}^j$ の軌道を用いて ProMP を学習する. この際, 各行動に対して 20 個の軌道を生成し, ProMP の学習を行った.

- ロボット 1

行動 1.a: $\mathbf{F}_{[0,T]}(\varphi_{1,1} \Rightarrow \mathbf{F}\varphi_{3,1})$ を達成する行動

行動 1.b: $\mathbf{F}_{[0,T]}(\varphi_{1,1} \Rightarrow \mathbf{F}\varphi_{3,1}) \wedge \mathbf{F}_{[0,T]}(\varphi_{2,1} \Rightarrow \mathbf{F}\varphi_{3,1})$ を達成する行動

- ロボット 2

行動 2.a: $\mathbf{F}_{[0,T]}(\varphi_{2,2} \Rightarrow \mathbf{F}\varphi_{3,2})$ を達成する行動

行動 2.b: $\mathbf{F}_{[0,T]}(\varphi_{1,2} \Rightarrow \mathbf{F}\varphi_{3,2})$ を達成する行動

ここでは, ロボット 1 がロボット 2 の状態から行動を更新することを考える. ロボット 1 はロボット 2 の位置を観測することができ, 各ロボットには事前に行動 1.a, 行動 2.a が割り当てられているとする. このとき, 各ロボットが割り当てられた行動を実行することが出来れば, 命題 φ を満たすことができる. このとき, ロボット 1 がロボット 2 の行動を推定し, 行動を更新することでタスクを達成するための軌道が得られることを確認する.

6.6.2 結果・考察

本実験では, 行動 2.b の ProMP の平均軌道をロボット 2 の軌道としてロボット 1 が観測すると仮定し, ロボット 2 に対応する配合された ProMP に関する UKF を更新した際の挙動を確認した. ロボット 2 の動作が行動 2.a の ProMP に対して外れ値として判定された際にロボット 1 の行動を更新する. このとき, 最適化問題を計算する際のロボットの 2 の将来の軌道は, 更新する時刻でのロボット 2 の状態から変化しないものとする. また, 外れ値の判定は $\mathbf{d}_{Pro} > 4$ で外れ値と判定し, 行動 2.a, 行動 2.b に対応する ProMP と推定した ProMP に対する外れ値を $\mathbf{d}^a_{Pro}, \mathbf{d}^b_{Pro}, \mathbf{d}^{\hat{\gamma}}_{Pro}$ とする.

シミュレーションの結果を Fig.6.2, シミュレーション中の外れ値の値を Fig.6.3, 推定されたロボット 1, 2 の ProMP の配合に関するパラメーター γ_1^1, γ_1^2 と τ の軌道を Fig.6.4, 6.5, 6.6 に示す. Fig.6.2 で, 黒円はロボット 2 の状態, マゼンタの円はロボット 1 の ProMP 上での現在の状態, マゼンタの軌道はロボット 1 の ProMP の軌道, 緑と紫の軌道はロボット 2 の推定された ProMP の軌道 ($\mathbf{d}_{Pro} > 4$ の時刻では緑), 茶色の軌道は最適化問題を計算する際のロボットの 2 の軌道である. 本実験では, シミュレーションの早い段階で外れ値として検知され, ロボット 1 の行動の更新が行われた. Fig.6.2 および Fig.6.4 から, 最適化問題を解くことによってロボット 1 の行動が行動 1.b に切り替わり命題 φ を満たす行動をとっていることがわかる. また, 本実験における最適化問題の計算では 15 – 20 秒程度を要した. 加えて, Fig.6.3, 6.5, 6.6 から UKF によってロボット 2 の行動が行動 2.b に属していることが推定されていることが確認できる.

6.7 まとめ

本章では, 観測情報に基づきシステムの行動を変更するための手法を提案した. まず, 複数の ProMP が配合された ProMP に対して, UKF を用いて配合に関するパラメーターを推定

する手法を提案した. その後, UKF から得られたパラメーターからシステムの行動を推定し, 推定したシステムの行動から STL 命題を達成する軌道を生成するための軌道最適化問題を定式化した. 最後に, 移動ロボットに対して提案手法を適用し, システムの行動の推定や軌道の変更が可能なことを示した.

本章の数値実験では, 2 つシステムが動作するタスクに対して提案手法を適用した. 一方で, 提案手法を用いて行動を更新したのは 1 つのシステムのみである. 複数のシステムが動作するタスクにおいて, 提案手法によって互いの行動を推定し, 行動を更新する状況で提案手法を検討することは今後の課題と考えられる. また, 提案手法で生成した目標軌道を用いてシステムの制御を行い, 有効性を検証することや計算時間の短縮も今後の課題である.

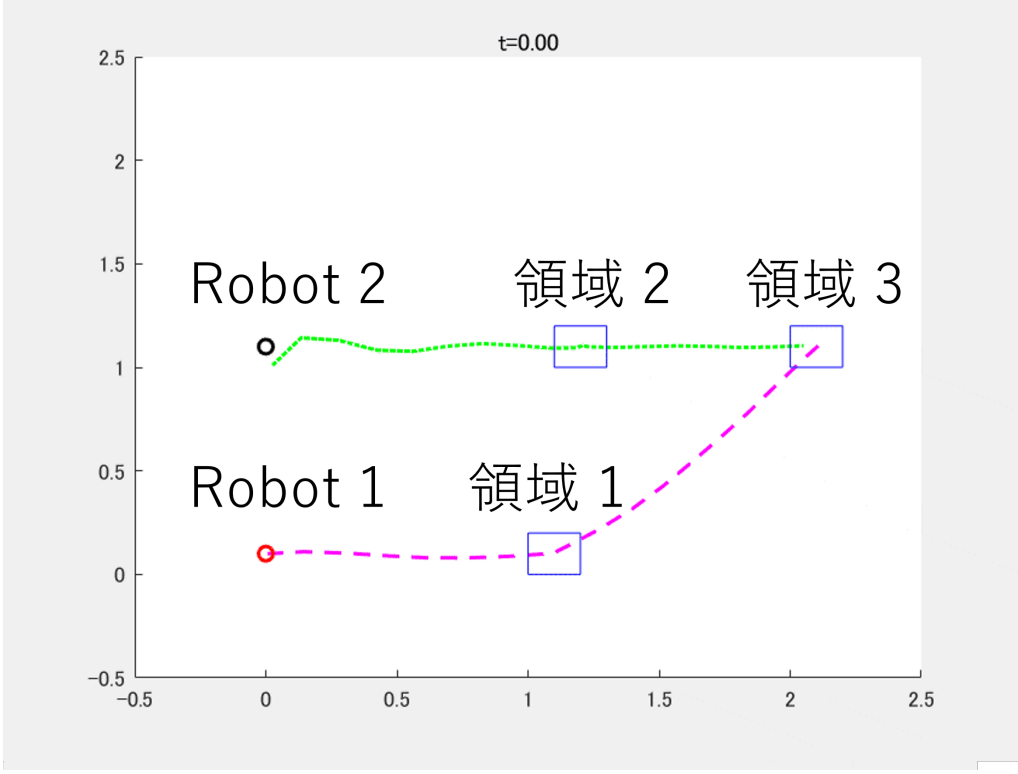

Fig. 6.1: Workspace of the two mobile robots

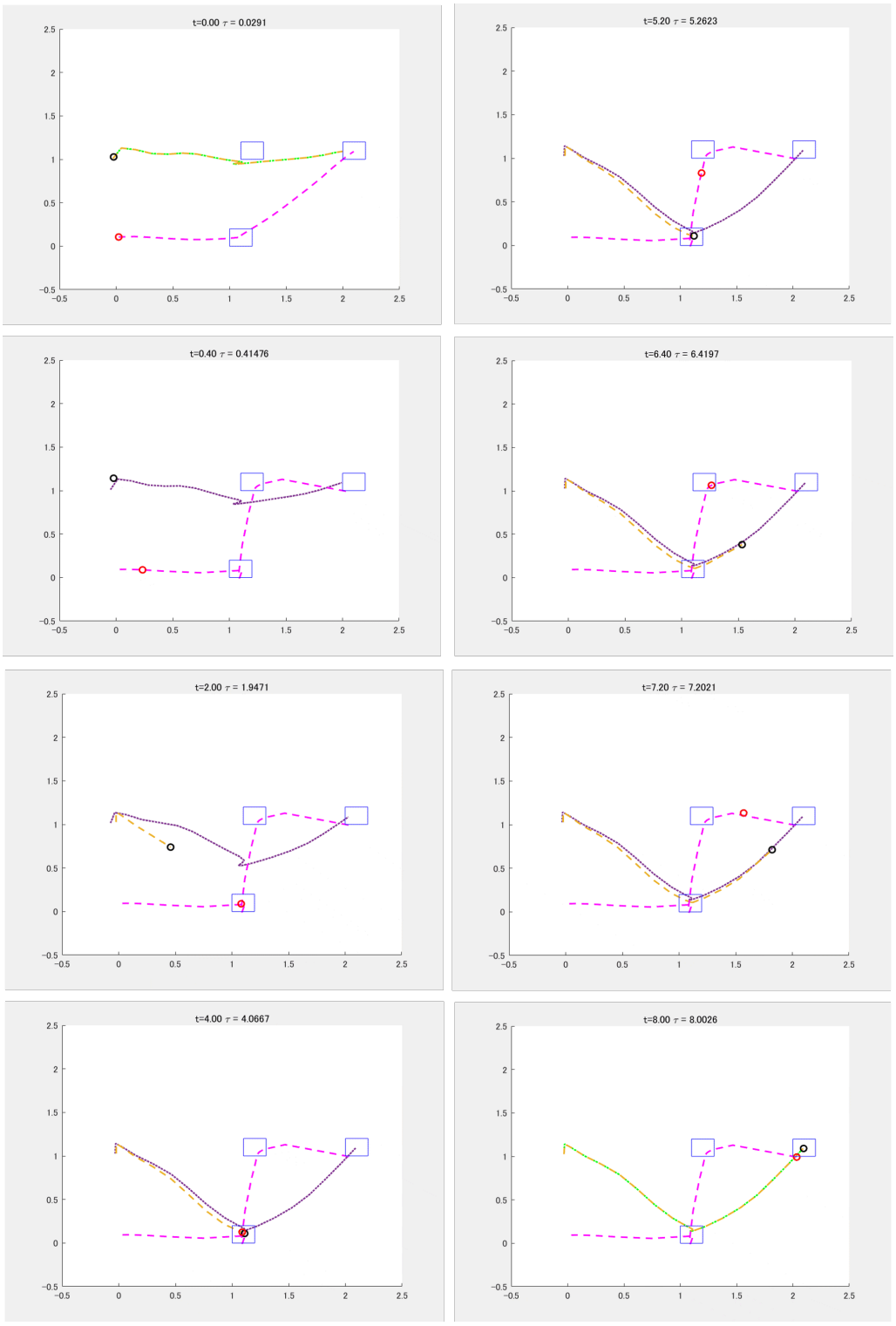

Fig. 6.2: Behaviors of ProMPs in the numerical simulation. Black circles are states of Robot 2, magenta circles are current states of ProMPs, magenta trajectories are trajectories of ProMP of robot 1, perple and green trajectories are trajectories of estimated ProMP of robot 2: when $\mathbf{d}^1_{Pro} < 4$, trajectories are green, and brown trajectoies are trajectories used in optimization problem for ProMP of robot 1.

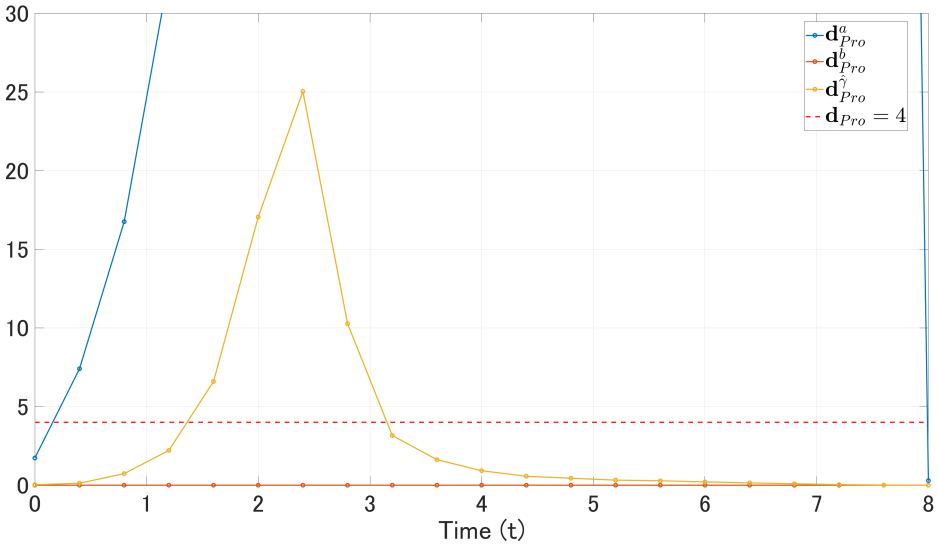

Fig. 6.3: The trajectories of outliers for ProMPs of Motion 2.a, Motion 2.b and esatimated one.

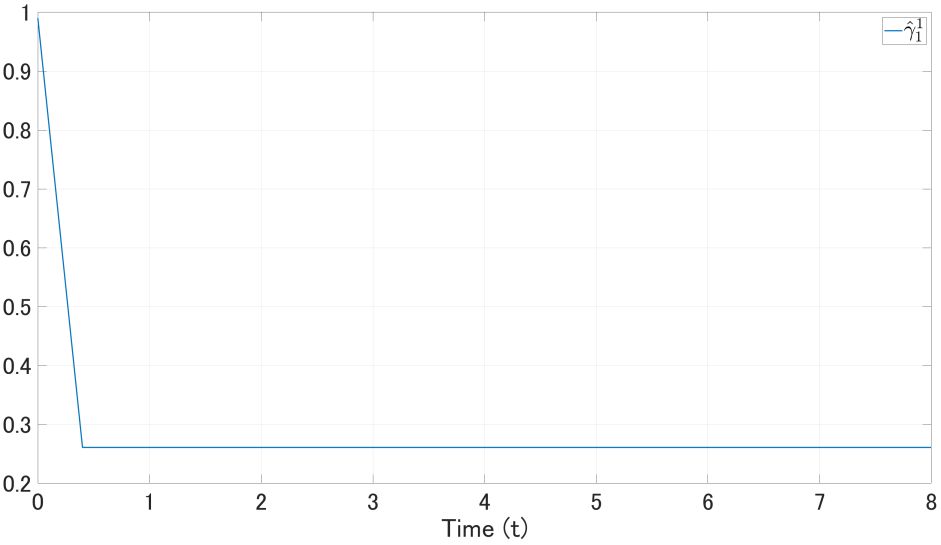

Fig. 6.4: The trajectory of the blending parameter of ProMP of robot 1: γ^1_1

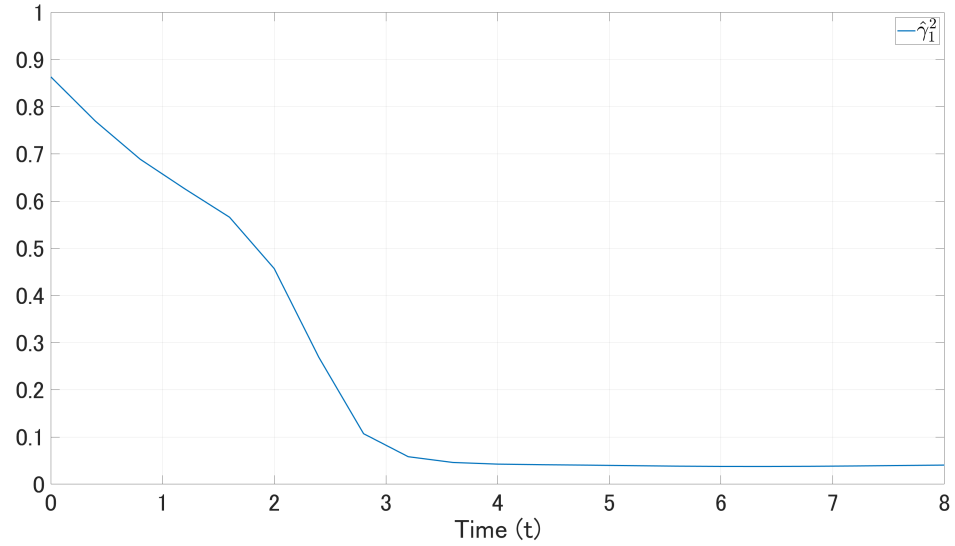

Fig. 6.5: The trajectory of the blending parameter of ProMP of robot 2: γ_1^2

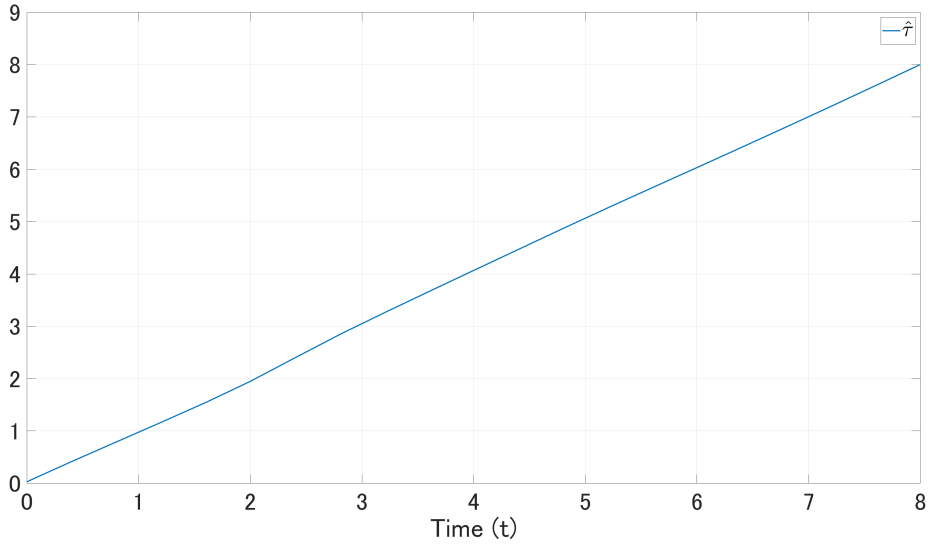

Fig. 6.6: The trajectory of the time parameter of ProMP of robot 2: τ

第7章　結論

本論文では, 時相論理で表現されたタスクをロボットシステムが達成するための軌道最適化手法と, 制御手法の提案を行った. また, 時相論理命題で表現されたタスクを達成する軌道最適化問題に対する混合整数線形計画問題や非線形計画問題による定式化において, 適用可能なロボットシステムや命題のクラスを拡張した. 加えて, 生成した軌道や行動計画を用いてタスクを達成するための制御手法として, 時間制約をもつ制約条件を達成するための制御手法や行動を変更する手法の提案および性能の改善を行った. 本論文で得られた成果を以下にまとめる.

2 章では, 本論文で利用する時相論理に関するいくつかの定義の導入を行い, ロボットシステムの振る舞いを時相論理を用いて表現できることを説明した.

3 章では, 非線形システムに対する LTL 命題を満たす軌道を生成するための行動計画問題を, 混合整数線形計画問題として定式化する手法を構築した. まず, ハイブリッドシステムである線形システム, MLD モデルに対する行動計画が混合整数計画問題として定式化することが可能なことについて説明した. その後, 混合整数凸計画を用いることでダイナミクスに二乗項や双線形項を持つシステムを線形システムとして表現でき, 先に述べた混合整数計画問題に組み込むことができることを示した. 加えて, 拡大線形化を用いて拡大線形システムまたは拡大双線形システムを設計することで, 非線形システムを混合整数計画問題に組み込むことができることを示した. 数値シミュレーションで提案手法の有効性検証を行い, LTL 命題を満たす軌道を生成することが可能であることを確認した.

4 章では, 発生時刻が不確かな外的事象に関する命題を達成するための軌道最適化手法を提案した. 正確な発生時刻が不明な外的事象について発生時刻に関する分布を仮定し, その分布に基づき, Event-based STL 命題を達成する軌道を生成することが可能な軌道最適化を非線形計画問題として定式化した. 軌道最適化を行う際に, 外的事象の発生に関する変数を最適化変数に含めることで, 外的事象の発生時刻に対する軌道のロバスト性を評価することを可能にした. 最後に, 不確かな外的事象に関する命題が与えられた移動ロボットに対する数値シミュレーションにて, 提案手法を用いることで外的事象の発生に関する複数のシナリオに対して与えられた命題が達成されることを確認した.

5 章では, 時間制約をもつ信号時相論理を達成するための制御バリア関数を用いた制御手法を提案した. STL 命題に対応する時間オートマトンと時間オートマトン上で STL 命題を満たすパスを考え, 各ロケーション間の遷移を達成することを目的とし, 制御バリア関数を拡張した制御器を提案した. 加えて, 関数の値によって進行速度の変化する仮想時間を定義し, 定義した仮想時間上で提案した制御器を用いて制御した場合, 元の時間軸上でも時間制約を

満たすことを示した. その後, 本章で考える制御構造を説明し, パス上でのロケーションの遷移を逐次達成することで与えられた本来の命題が達成されることを示した. 最後に, 移動ロボットに対する数値シミュレーションにて, 制約を満たしながら時間内に命題を達成することが可能であり, 仮想時間を導入することで解の消失のリスクが低減されることを確認した.

6 章では, 複数のシステムでタスクを行う際に, 他のシステムの行動に合わせて自身の行動を修正するための軌道最適化手法を提案した. まず, 観測情報からシステムの状態が属している確率的動作プリミティブをカルマンフィルターを用いて推定する手法について説明した. その後, 推定された他のシステムの確率的動作プリミティブを基に, 確率的動作プリミティブを更新することで, 他のシステムの動作を考慮して与えられた命題を達成する軌道を生成する軌道最適化問題を定式化した. 最後に, 2 台の移動ロボットに対する数値シミュレーションにて, ロボットが他のロボットに関する観測情報を基に自身の行動を変更することが可能なことを確認した.

これらの結果は, ロボットのタスク実行という課題のアプローチの一つである時相論理を用いた行動計画手法に対して, 新たな観点から高性能化と拡張性の向上の可能性を示したものであり，この分野の研究ならびに応用を大きく進展させると期待している.

最後に本論文の研究成果に対する今後の課題について述べる.

- 動作中の行動計画の再計算
  実際の作業環境では, タスクを実行する際に動作の失敗や環境の変化が発生し動作を変更する必要がある場合がある. 本論文では, 4 章にて外的事象の発生に対してロバスト性を持つ軌道を生成することや, 6 章にて ProMP の推定や再配合によって, 環境の変化や動作変更に対応することを検討した. しかし, 動作のやり直しが必要な状況など, 本論文で提案した手法で対応することが難しい状況も存在する. そういった状況に対して対応するための手法として, 動作中に再び行動計画を行うことなどが挙げられる. しかし, 本論文で提案した手法を含め, 高レベル制御器として考えれている手法は計算コストが高く, 動作途中に実行することは現実的では無いといった問題がある. そのため, 計算コストを抑えるために近似システムの構築方法のさらなる検討や計算の高速化が課題であると考えられる.

- 実機実験における有効性検証
  本論文では, 数値シミュレーションを用いて提案手法毎に有効性を確認した. 今後の課題として, 実機実験に対して本論文で考える制御アーキテクチャを適用し, 提案手法の有効性検証も行う必要があると考えている.

参考文献

[1] 経済産業省，“ロボットを取り巻く環境変化と今後の施策の方向性〜ロボットによる社会変革推進計画〜”，https://www.meti.go.jp/shingikai/mono_info_service/robot_shakaihenkaku/pdf/20190724_report_01.pdf，2019.

[2] S. Karaman, R. G. Sanfelice, and E. Frazzoli, “Optimal control of mixed logical dynamical systems with linear temporal logic specifications,” in *2008 47th IEEE Conference on Decision and Control (CDC)*, 2008, pp. 2117–2122.

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

[4] C. Baier, and J.-P. Katoen, *Principles of model checking*. MIT press, 2008.

[5] O. Maler, and D. Nickovic, “Monitoring temporal properties of continuous signals,” in *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, Y. Lakhnech, and S. Yovine, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2004, pp. 152–166.

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

[7] C. Belta, B. Yordanov, and E. Gol, *Formal Methods for Discrete-Time Dynamical Systems*. Springer, 2017, vol. 89.

[8] X. Ding, S. L. Smith, C. Belta, and D. Rus, “Optimal control of markov decision processes with linear temporal logic constraints,” *IEEE Transactions on Automatic Control*, vol. 59, no. 5, pp. 1244–1257, 2014.

[9] V. Nenchev, C. Belta, and J. Raisch, “Optimal motion planning with temporal logic and switching constraints,” in *2015 European Control Conference (ECC)*, 2015, pp. 1141–1146.

[10] C. Dawson, and C. Fan, “Robust counterexample-guided optimization for planning from differentiable temporal logic,” in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 7205–7212.

[11] J. Envall, R. Poranne, and S. Coros, "Differentiable task assignment and motion planning," in *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2023, pp. 2049–2056.

[12] M. Cai, S. Xiao, Z. Li, and Z. Kan, "Optimal probabilistic motion planning with potential infeasible ltl constraints," *IEEE Transactions on Automatic Control*, vol. 68, no. 1, pp. 301–316, 2021.

[13] R. Takano, H. Oyama, and M. Yamakita, "Continuous optimization-based task and motion planning with signal temporal logic specifications for sequential manipulation," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*, 2021, pp. 8409–8415.

[14] K. Takaya, R. Takano, and H. Oyama, "Metaheuristics approach for mathematical programs with switching constraints and application to robotic task planning," in *2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, 2022, pp. 77–84.

[15] S. Haesaert, S. Soudjani, and A. Abate, "Temporal logic control of general markov decision processes by approximate policy refinement," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 73–78, 2018, 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318311261

[16] S. Haesaert, P. Nilsson, C. Vasile, R. Thakker, A. Agha-mohammadi, A. Ames, and R. Murray, "Temporal logic control of pomdps via label-based stochastic simulation relations," *IFAC-PapersOnLine*, vol. 51, no. 16, pp. 271–276, 2018, 6th IFAC Conference on Analysis and Design of Hybrid Systems ADHS 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896318311625

[17] V. Raman, A. Donze, M. Maasoumy, R. M. Murray, A. Sangiovanni-Vincentelli, and S. A. Seshia, "Model predictive control with signal temporal logic specifications," in *53rd IEEE Conference on Decision and Control*, 2014, pp. 81–87.

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

[19] L. Lindemann, and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[20] P. Nilsson, and A. D. Ames, “Barrier functions: Bridging the gap between planning from specifications and safety-critical control,” in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 765–772.

[21] M. Srinivasan, S. Coogan, and M. Egerstedt, “Control of multi-agent systems with finite time control barrier certificates and temporal logic,” in *2018 IEEE Conference on Decision and Control (CDC)*, 2018, pp. 1991–1996.

[22] M. Srinivasan, and S. Coogan, “Control of mobile robots using barrier functions under temporal logic specifications,” *IEEE Transactions on Robotics*, vol. 37, no. 2, pp. 363–374, 2021.

[23] A. D. Ames, X. Xu, J. W. Grizzle, and P. Tabuada, “Control barrier function based quadratic programs for safety critical systems,” *IEEE Transactions on Automatic Control*, vol. 62, no. 8, pp. 3861–3876, 2017.

[24] A. Li, L. Wang, P. Pierpaoli, and M. Egerstedt, “Formally correct composition of coordinated behaviors using control barrier certificates,” in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 3723–3729.

[25] C.-I. Vasile, J. Tumova, S. Karaman, C. Belta, and D. Rus, “Minimum-violation scltl motion planning for mobility-on-demand,” in *2017 IEEE International Conference on Robotics and Automation (ICRA)*, 2017, pp. 1481–1488.

[26] 小野寛晰，情報科学における論理　日本評論社，1994.

[27] M. Bauland, M. Mundhenk, T. Schneider, H. Schnoor, I. Schnoor, and H. Vollmer, “The tractability of model checking for ltl: The good, the bad, and the ugly fragments,” *ACM Trans. Comput. Logic*, vol. 12, no. 2, 2011. [Online]. Available: https://doi.org/10.1145/1877714.1877719

[28] S. Dathathri, and R. M. Murray, “Decomposing gr(1) games with singleton liveness guarantees for efficient synthesis,” in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*, 2017, pp. 911–917.

[29] R. Alur, T. Feder, and T. A. Henzinger, “The benefits of relaxing punctuality,” *J. ACM*, vol. 43, no. 1, p. 116–146, 1996. [Online]. Available: https://doi.org/10.1145/227595.227602

[30] H. Dai, G. Izatt, and R. Tedrake, “Global inverse kinematics via mixed-integer convex optimization,” *The International Journal of Robotics Research*, vol. 38, no. 12-13, pp. 1420–1441, 2019.

[31] J. P. Vielma, and G. L. Nemhauser, "Modeling disjunctive constraints with a logarithmic number of binary variables and constraints," *Mathematical Programming*, vol. 128, no. 1, pp. 49–72, 2011.

[32] S. L. Brunton, B. W. Brunton, J. L. Proctor, and J. N. Kutz, "Koopman invariant subspaces and finite linear representations of nonlinear dynamical systems for control," *PLOS ONE*, vol. 11, no. 2, pp. 1–19, 2016. [Online]. Available: https://doi.org/10.1371/journal.pone.0150171

[33] A. Surana, "Koopman operator based observer synthesis for control-affine nonlinear systems," in *2016 IEEE 55th Conference on Decision and Control (CDC)*, 2016, pp. 6492–6499.

[34] D. Bruder, X. Fu, and R. Vasudevan, "Advantages of bilinear koopman realizations for the modeling and control of systems with unknown dynamics," *IEEE Robotics and Automation Letters*, vol. 6, no. 3, pp. 4369–4376, 2021.

[35] M. Korda, and I. Mezić, "Linear predictors for nonlinear dynamical systems: Koopman operator meets model predictive control," *Automatica*, vol. 93, pp. 149–160, 2018. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S000510981830133X

[36] E. Yeung, S. Kundu, and N. Hodas, "Learning deep neural network representations for koopman operators of nonlinear dynamical systems," in *2019 American Control Conference (ACC)*, 2019, pp. 4832–4839.

[37] E. M. Wolff, and R. M. Murray, "Optimal control of mixed logical dynamical systems with long-term temporal logic specifications," 2013.

[38] M. Earl, and R. D'Andrea, "Iterative milp methods for vehicle-control problems," *IEEE Transactions on Robotics*, vol. 21, no. 6, pp. 1158–1167, 2005.

[39] M. Yamakita, K. Jimura, and K. Furuta, "Impedance posture control of robot manipulators using euler parameters," in *Proceedings of 35th IEEE Conference on Decision and Control*, vol. 2, 1996, pp. 1964–1966 vol.2.

[40] E. Süli, and D. F. Mayers, *An introduction to numerical analysis*. Cambridge university press, 2003.

[41] L. Gurobi Optimization, "Gurobi optimizer reference manual," 2021. [Online]. Available: http://www.gurobi.com

[42] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[43] L. Niu, and A. Clark, "Control barrier functions for abstraction-free control synthesis under temporal logic constraints," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 816–823.

[44] D. Gundana, and H. Kress-Gazit, "Event-based signal temporal logic tasks: Execution and feedback in complex environments," *IEEE Robotics and Automation Letters*, vol. 7, no. 4, pp. 10 001–10 008, 2022.

[45] A. Wächter, and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Math. Program.*, vol. 106, no. 1, pp. 25–57, 2006.

[46] Y. Gilpin, V. Kurtz, and H. Lin, "A smooth robustness measure of signal temporal logic for symbolic control," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 241–246, 2021.

[47] R. Alur, and D. L. Dill, "A theory of timed automata," *Theoretical Computer Science*, vol. 126, pp. 183–235, 1994.

[48] X. Xu, P. Tabuada, J. W. Grizzle, and A. D. Ames, "Robustness of control barrier functions for safety critical control," *IFAC-PapersOnLine*, vol. 48, no. 27, pp. 54–61, 2015, analysis and Design of Hybrid Systems ADHS. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2405896315024106

[49] K. Sekiguchi, and M. Sampei, "On multi time-scale form of nonlinear systems," *IFAC Proceedings Volumes*, vol. 46, no. 23, pp. 524–529, 2013, 9th IFAC Symposium on Nonlinear Control Systems. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S147466701631713X

[50] L. Niu, and A. Clark, "Control barrier functions for abstraction-free control synthesis under temporal logic constraints," in *2020 59th IEEE Conference on Decision and Control (CDC)*, 2020, pp. 816–823.

[51] L. Lindemann, and D. V. Dimarogonas, "Control barrier functions for signal temporal logic tasks," *IEEE Control Systems Letters*, vol. 3, no. 1, pp. 96–101, 2019.

[52] K. G. Larsen, P. Pettersson, and W. Yi, "Uppaal in a nutshell," *Int. J. Softw. Tools Technol. Transf.*, vol. 1, no. 1–2, p. 134–152, 1997. [Online]. Available: https://doi.org/10.1007/s100090050010

[53] F. Blanchini, “Set invariance in control,” vol. 35, no. 11, 1999, pp. 1747–1767. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0005109899001132

[54] F. Blanchini, and S. Miani, *Set-Theoretic Methods in Control*, 1st ed. Birkhäuser Basel, 2007.

[55] S. Documentation, “Simulation and model-based design,” 2023. [Online]. Available: https://www.mathworks.com/products/simulink.html

[56] M. Charitidou, and D. V. Dimarogonas, “Signal temporal logic task decomposition via convex optimization,” *IEEE Control Systems Letters*, vol. 6, pp. 1238–1243, 2022.

[57] Y. Chen, X. C. Ding, A. Stefanescu, and C. Belta, “Formal approach to the deployment of distributed robotic teams,” *IEEE Transactions on Robotics*, vol. 28, no. 1, pp. 158–171, 2012.

[58] A. Paraschos, C. Daniel, J. R. Peters, and G. Neumann, “Probabilistic movement primitives,” *Advances in neural information processing systems*, vol. 26, 2013.

[59] S. Gomez-Gonzalez, G. Neumann, B. Schölkopf, and J. Peters, “Adaptation and robust learning of probabilistic movement primitives,” *IEEE Transactions on Robotics*, vol. 36, no. 2, pp. 366–379, 2020.

[60] A. Paraschos, C. Daniel, J. Peters, and G. Neumann, “Using probabilistic movement primitives in robotics,” *Autonomous Robots*, vol. 42, pp. 529–551, 2018.

[61] E. Wan, and R. Van Der Merwe, “The unscented kalman filter for nonlinear estimation,” in *Proceedings of the IEEE 2000 Adaptive Systems for Signal Processing, Communications, and Control Symposium (Cat. No.00EX373)*, 2000, pp. 153–158.

[62] R. Kandepu, B. Foss, and L. Imsland, “Applying the unscented kalman filter for nonlinear state estimation,” *Journal of Process Control*, vol. 18, no. 7, pp. 753–768, 2008. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0959152407001655

[63] R. Merwe, and E. Wan, “Sigma-point kalman filters for probabilistic inference in dynamic state-space models,” *Proceedings of the Workshop on Advances in Machine Learning*, 2003.

研究業績

研究業績 (発表順に掲載)

学術雑誌論文

- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, Rin Takano, “Reblending of Probabilistic Movement Primitives for Signal Temporal Logic Specifications,” IEEE Control Systems Letters (L-CSS). (submitted to L-CSS and CDC). (6 章に関連)

国際会議論文

- Mizuho Katayama, Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, “Fast LTL Based Flexible Planning for Dual-Arm Manipulation,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020. (3 章 3.3 節に関連)

- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, Rin Takano, “Convex Approximation for LTL-based Planning,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2021. (3 章 3.4, 3.6 節に関連)

- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, Rin Takano, “Linear Temporal Logic-based Mixed-Integer Linear Problem Planning with the Koopman Operator,” Annual Conference of the IEEE Industrial Electronics Society (IECON), 2022. (3 章 3.5, 3.7 節に関連)

- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, Rin Takano, “Zeroing Control Barrier Function with Time Scale Transformation for Time Interval Signal Temporal Logic Task,” IEEE Conference on Control Technology and Applications (CCTA), 2022. (5 章に関連)

- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, Rin Takano, “Robust Signal Temporal Logic-based Planning for Uncertain External Events,” IEEE International Conference on Robotics and Biomimetics (ROBIO), 2023. (4 章に関連)

博士論文に関係しないもの

国際会議発表論文

- Shumpei Tokuda, Masaki Yamakita, “Hierarchical Linearization and Output Zeroing Control for Wheeled Robots,” Annual Conference of the Society of Instrument and Control Engineers of Japan (SICE), 2019
- Shumpei Tokuda, Masaki Yamakita, Hiroyuki Oyama, “Generating New Lower Abstract Task Operator using Grid TLI,” IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2020.

国内会議発表論文

- 大山博之, 伊藤岳大, 佐藤峰斗, 片山瑞穂, 徳田俊平, 山北昌毅, “時相論理を用いた双腕ロボットの自律制御,” 第 7 回 制御部門マルチシンポジウム (MSCS2020), 2020

特許

- 日本電気株式会社, 国立大学法人東京工業大学, 高野凜, 大山博之, 山北昌毅, 徳田俊平, “制御システム、制御方法およびプログラム,” 特開 2023-047980, 2023

謝辞

本論文は東京工業大学工学院システム制御系山北研究室と日本電気株式会社 データサイエンス研究所との共同研究の成果を元に作成されており, 本研究を遂行するにあたり，様々な方々からのご助言やご指導などを賜りました.

特に, 指導教員・論文審査員主査としてご指導・ご助言をくださいました山北昌毅准教授に心より御礼申し上げます. 学士課程から博士課程までの 6 年間の長い間, 数学や制御工学など様々な学問の知識に加えて, 論文執筆や学会発表の方法, 研究の進め方など様々なご指導を賜りました. 6 年間のご指導のおかげで, 多岐にわたる分野の知識や研究に対する考え方に関して多くのことを学び, 研究者として成長することができたと実感しています. 重ねて感謝申し上げます.

本論文の審査員を務めていただき，丁寧な論文審査の過程で様々なご助言をくださいました倉林大輔教授, 三平満司教授，田中正行教授, 畑中健志准教授に深く感謝申し上げます.

日本電気株式会社 データサイエンス研究所の皆様には共同研究を行っていただき，大変お世話になりました．研究の方向性や内容に関する多くの議論を通じて, 研究の質を大きく向上させることができました. 深く感謝申し上げます. 特に大山博之様, 高野凛様, 田屋裕輝様には, 様々な面でご指導・御鞭撻を賜りました．大山様, 高野様には研究室の OB として, 田屋様には学生時代の同期として, 共同研究以外でも学生生活における相談にのっていただき, 大変お世話になりました.

カンボジア工科大学 Sarot Srang 博士とは, UAV 制御の共同研究ができたことを感謝申し上げます. また, カンボジアにお招きいただき講演する機会をいただいたことを嬉しく思っております.

学会の手続きや共同研究等の事務手続きに関して大変お世話になりました小林真理事務官に心よりお礼申し上げます．山北研究室での研究生活において様々な議論をするとともに，日々の充実した生活を共に過ごした山北研究室の学生の皆様に感謝申し上げます．特に, 共同研究グループにて共に研究を行った, 片山瑞穂氏, 福田健二氏とは互いに研究内容について相談することができ, 結果として研究を進める手助けになったと思います. ありがとうございました. また, 山北准教授の奥様には様々な点でご配慮いただき, 大変お世話になりました. 深く感謝申し上げます．加えて, 学士課程から博士課程まで同期として学生生活を共に過ごした 久米啓太氏, 持田峻佑氏, 谷口晃大氏, 横山陽彦氏に感謝申し上げます.

最後に，精神的・経済的に私を支えてくれた家族に心よりお礼申し上げます.

なお，本研究は JST SPRING, Grant Number JPMJSP2106 の助成を受けたものです．ここに記し，感謝の意を表します.