2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

# Fast LTL-Based Flexible Planning for Dual-Arm Manipulation

Mizuho Katayama[1], Shumpei Tokuda[1], Masaki Yamakita[1], and Hiroyuki Oyama[2]

***Abstract*— In this paper, we propose a method for automatically generating object handling actions based on simple action definitions. The need to replace workers by robots is increasing, and, in fact, many research projects on robots have worked with simple motion definitions. Many applications are for mobile robots such as drones, however, and if such methods are applied directly to object handling, like a pick and place operation, it is necessary for humans to give detailed instructions. Hence, our contribution is to propose a model that simulates the real world with an augmented hybrid system that includes the states of objects. Then, it becomes possible to automatically generate robot motions with simple motion definitions and calculate them within a reasonable time. We demonstrate through computer simulation with a dual-arm robot that robot motions can be generated by simple definitions even if the environment changes to a certain degree.**

## I. INTRODUCTION

### A. Background

A shortage of workers due to an aging population is a problem in some countries. In particular, there are significant labor shortages in urban areas. Meanwhile, workplaces for human workers are narrower and messier than environments in factories where industrial robots are used. For typical jobs like food manipulation in packing lunch boxes and cashiering at convenience stores, it is difficult for conventional robots to perform these tasks. This is because it is necessary to secure a sufficiently large space and prepare the working environment with jigs. In a convenience store, on the other hand, the space occupied by a cash register is small and surrounded by goods. Such environments are very difficult for operation of conventional robots [1].

Teaching tasks to robots is a major barrier in introducing them to such environments. If a convenience store wants to operate robots correctly in the store environment, specific exceptions must be taken into account, and subtle changes to the work and so on are likely to occur frequently. It is not economical, however, to make these exceptions in accordance with frequent changes. In practice, it is thus necessary for robots to do tasks with the kinds of simple directions that workers are given. Otherwise, it would be difficult for robots to work like such workers. In addition, it is important to ensure safety in the work environment.

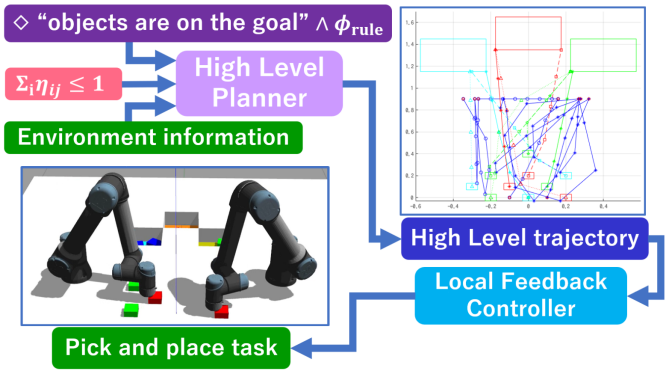


Fig. 1. System configuration: A high-level planner outputs a 2D high-level trajectory from the LTL, constraints, and environment settings. The LTL consists of a description of the task definition itself, such as "objects are on the goal", and a description of some rule such as $\phi_{\text{rule}}$. The high-level trajectory consists of coarse time-series data with position and velocity. By interpolating in the z-direction and between time steps, the robot is controlled by a local feedback controller.

### B. Formal Planning

One approach that simplifies teaching robots is teaching by demonstration[2],[3]. Another approach is to apply goal-oriented action planning (GOAP), which recently has often been used for creating games [4]-[6]. Though these techniques can teach tasks efficiently and are attractive, they should also guarantee work safety in advance. In other words, it is necessary that safety can be verified in advance. One way to describe tasks is to use linear temporal logic (LTL) or signal temporal logic (STL) [7],[9]. Also, it is expected that instructions given in natural language can be translated into LTL [10],[11].

By using LTL to describe safety rules or handling skills that are memorized by a robot as "common sense," it can be checked in advance whether newly learned "common sense" rules and given task are consistent with general stored rules to ensure safety or other common rules. It is generally very computationally intensive, however, to check such consistency [12]. Therefore, practical applications should use a special subclass of LTL, such as GR(1) or syntactically co-safe LTL (scLTL) [11],[13], [14]-[16].

In addition to ensuring security in advance, simple task description and flexible planning are another important points. By modeling the real world with a hybrid system and performing task description in a simple LTL, it is possible to flexibly deal with some changes in the environment (e.g., when the place to carry objects has changed, when robot arm

[1]M.Katayama, [1]S.Tokuda, and [1]M.Yamakita are with the Department of Systems and Control Engineering, Tokyo Institute of Technology, 1-11-1 Oh-Okayama, Meguro, Tokyo 151-8551, Japan katayama@ac.sc.e.titech.ac.jp, tokuda@ac.sc.e.titech.ac.jp, yamakita@ac.sc.e.titech.ac.jp ([1]M.Katayama is with Biometrics Research Laboratories, NEC Corporation now. m.katayama.g@nec.com)

[2]H.Oyama is with Data Science Research Laboratories, NEC Corporation, 1753, Shimonumabe, Nakahara-ku, Kawasaki, Kanagawa, 211-8666, Japan. h.oyama@nec.com.

does not reach its destination!) and change the behavior of the robot logically (We see in section V).

### C. Issues with Existing Methods

In the proposed methods [7],[15],[17] of defining the behavior of robots in LTL, the behaviors of a state that can be directly input are indicated in LTL, as in the case of drones. If this method is simply applied to handling operations like the pick and place task, a human must precisely specify the behavior of the robot's hands, and the task cannot be performed with only simple instructions. Another problem is that the method of defining logical propositions proposed by [7] cannot be applied to release an object once grasped, making it unsuitable for modeling robot behavior in the pick and place task. In addition, there is a problem in that the number of calculation steps becomes large for the task of moving an object many times, as in transportation. In the above method [7],[15], the problem is converted to a mixed integer linear programming (MILP) formulation, which is a kind of combinatorial optimization. For the pick and place task, that approach would not be albe to generate a solution within a reasonable amount of time on a standard computer.

### D. Contributions

In this paper, we show that a planning using LTLs and hybrid system is effective for pick and place tasks by our modeling methods and problem relaxation. In this planning, simple task descriptions are available and logically modifiable, and flexible for environmental changes. We consider the pick and place task with a dual-arm system, which is similar to the case of handling operations by a person. Then, we propose algorithms that generate the desired trajectories by using LTL task specification, as shown in Fig. 1. The paper has three main contributions:

1) Simple LTL operation definition by a hybrid system:
   We propose a method for a robot to perform the pick and place task with simple instructions by handling not only the state of the hand but also the states of objects as the state of the system, and by using LTL to specify the behavior of the objects' states.
2) Model of picking behavior:
   We propose a new model for pick and place movement that can release an object once it has been grabbed by using a secondary effect obtained from [18]. This effect is related to the determinism of the logical value of a proposition. By our modeling method, the method proposed in [7] can be applied to manipulation tasks.
3) Fast calculation by sequential transportation:
   The pick and place task involves a large number of calculation steps, which may take a long time. We thus propose an algorithm for sequentially transporting target objects so that the desired trajectories can be calculated at high speed.

Specifically, we consider the task of having a dual-arm robot move all objects from a collection site to different designated goal locations as quickly as possible. Through simulation, we check whether motion commands are generated in real time and objects are correctly gripped and released with only simple task definitions. For the form of the object movement, we consider two cases, in which the robot arm grips and carries objects and in which the robot slides objects on a table. As for the structure of the paper, the next section discusses related work, and then section III gives the preliminaries. Section IV describes the modeling method, specifications and algorithms. We show the computer simulation results in section V and then give the conclusion in section VI.

## II. RELATED WORK

A previous research project on defining robot motions with LTL was presented in [17]. Although it uses LTL, the path itself is generated by using a planning algorithm like rapidly expanding random tree (RRT) and selecting the path that satisfies the LTL specification. Therefore, this approach is suitable for use in route planning for drones, but it requires much human assistance for application to the pick and place task.

In [8], motion planning for workpiece pickup operation by dual-armed robots is solved by optimization, but the formulation requires human design (e.g., the variable $B$, indicating the place where a workpiece is picked up), as do the paths of the arms. Then, the timing of the arms (and $B$) is obtained. Notions like "place where the workpiece is picked up" are extremely problem-dependent and abstract matters requiring human design. In this paper, by modeling primitive phenomena in the real world, we enable an object to be moved by grabbing it via a hybrid system. The system performs motion planning based on language definition while minimizing the need for human design of problem-dependent variables and formulation of optimization problems.

In [7], a trajectory is calculated from LTL and a given system dynamics without the aid of a planning algorithm, but the example in that paper is the problem of directing the behavior of a directly controllable state by LTL. In the task of pick and place, however, the target object is not always controllable.

In summary, trajectory optimization such as [8] requires humans to design the details for each individual case, therefore we would like to use LTLs to make a simple and flexible planning. However, existing LTL-based methods [17] and [7] have problems in application to pick and place. Therefore, we propose a modeling method to enable the method in [7] in pick and place task planning. A key feature of this paper is that a robot is moved only with instructions regarding the state that cannot be directly controlled, that is, where an object should be located, without clearly specifying the logic of how to grasp the object. We propose a method for achieving this in a reasonable amount of time, following the paradigm proposed by [7].

## III. PRELIMINARIES

### A. Notation

We usually use discrete-time instances, so we denote a set of times as $\mathcal{T} = \{0, 1, \ldots, T\} \subset \mathbb{Z}$, and the state at time $k$ as $x(k)$, $k \in \mathcal{T}$. We write $x(k+1)$ as $x^+$

and the transpose of a matrix $M$ as $M'$. For convenience, we sometimes use notations like $x(0:T)=1$, which means that the value of $x$ from time 0 to $T$ is 1. Let $a \cdot o = [a_1' o_1, \ldots, a_e' o_e]'$ for matrices $a, o \in \mathbb{R}^{n \times e}$, where $a_i, o_i$ are their column vectors. Let * denote any string. A rectangle consisting of $(x_1, y_1), (x_1, y_2), (x_2, y_1), (x_2, y_2)$ is represented as $(x_1, x_2) \times (y_1, y_2)$. The $x$ position of the arm $i$ is written as $x_{\text{arm}ix}\mathbb{R}$. $l$-dimensional closed-interval $[a, b]$ is represented as $[a, b]^l$.

### B. Modeling Settings

As in section IV, we consider end-effector dynamics $x_{\text{arm}i}$ and object dynamics $x_{\text{obj}j}$ because the full-dynamics of the robot arm is non-linear and difficult to be optimized. We assume floor-object friction is very small. Actual objects and arms' dynamics is in the 3D space but for easy implementation, we consider 2D dynamics.

### C. Mixed Logical Dynamical Systems

Our goal is to generate control inputs that satisfy the defined motions even in a real environment by a robot itself. This requires modeling the real world, and the hybrid system used here provides one such method. It is a dynamic system that mixes continuous and discrete-valued signals.

A mixed logical dynamical (MLD) system is one representation of a hybrid system. In this paper, we denote the system in an augmented form with the time index $k$ omitted:

$$\begin{cases} x^+ = Ax + B_1 u + B_2 z + B_3 \delta + B_4 \theta + B_5 p \\ Cx + D_1 u + D_2 z + D_3 \delta + D_4 \theta + D_5 p \leq E, \end{cases} \tag{1}$$

where $k \in \mathcal{T}$, $x(k) \in \mathbb{R}^n$ is the state, $u(k) \in \mathbb{R}^m$ is the input, $z(k) \in \mathbb{R}^{l_1}$ denotes continuous auxiliary variables, $\delta(k) \in \{0,1\}^{l_2}$ denotes discrete auxiliary variables. $A, B_1$ are the typical system dynamics matrices, $B_2$ is an auxiliary dynamics matrix, $B_3$ to $B_5$ are just formal ones (actually 0) and $C, D_*, E$ are the system constraints matrices. In addition, $\theta(k) \in \{0,1\}^{k_1}$ and $p(k) \in [0\ 1]^{k_2}$ are discrete and continuous logic variables for LTL, respectively. For convenience, the discrete variables are separated into $\delta$ and $\theta$, the traditional logic variables in MLD and the logic variables for LTL, respectively.

### D. Linear Temporal Logic

*1) Preliminaries:* To define the behavior of a robot, we need a language that can express time-varying events. Linear temporal logic (LTL) is a modal logic that can express changes in the values of propositions over time.

For example, the proposition "$x \in H$" is true if a vector $x$ is in a set $H$. In this situation, we consider only one context, "a vector $x$ exists in a set $H$." To interpret the proposition "always $x \in H$," however, we need a time series of contexts (i.e., $x(k)$) from time 0 to $+\infty$. If the time series includes even one context of "$x(k) \notin H$," then the proposition "always $x \in H$" is false. In this example, a proposition and contexts are given, and the value of the proposition is explained. In contrast, our goal is to give a proposition and its value as true, and then have a robot generate a time series of contexts (i.e., the robot's behavior). For details, see [7].

*2) LTL Operators:* In addition to classical logical operators($\wedge, \vee, \neg, \rightarrow$), LTL has temporal modal operators: *always* ($\Box$), *eventually* ($\Diamond$), *next* ($\bigcirc$), and *until* (**U**). Usually LTL syntax is defined in Backus-Naur form [7], but for this paper, it is sufficient to interpret the following LTL:

$$\phi = \Diamond\theta_1 \wedge \Box\neg\theta_2, \tag{2}$$

$$\phi = (\Diamond\theta_1 \wedge \Diamond\theta_2) \vee (\Diamond\theta_2 \wedge \Diamond\theta_3) \vee (\Diamond\theta_3 \wedge \Diamond\theta_1), \tag{3}$$

$$\phi = \Box(\theta \rightarrow \bigcirc\theta), \tag{4}$$

$$\theta_i = \text{“}x \in H_i\text{”},$$

where $\theta_*$ is a time-varying atomic proposition (see III-D.3). Among these statements, (2) means "$x$ must always avoid $H_2$ and eventually reach $H_1$," while (3) means "$x$ eventually reaches at least two of $H_1, H_2, H_3$." The final statement is more confusing, but the LTL statement "$\theta \rightarrow \bigcirc\theta$" means "if $\theta$ is true now (time 0), then the next $\theta$ is true," and the operator $\Box$ is then applied. Hence, (4) is equivalent to

$$\theta(k) \rightarrow \theta(k+1), \quad k \in \mathcal{T}/\{T\}.$$

*3) LTL Semantics:* We define atomic propositions and LTL formula.

**Definition** *4.1:* (Atomic propositions, $AP$) *An atomic proposition is a declarative sentence on the system state, which is either* `True` *or* `False`*. Any atomic proposition $\theta \in AP$ is an LTL formula.*

In this paper, a variable $\theta$ in the MLD system is an atomic proposition, and it means "the state $x$ of the system exists in a set $H$," that is, $x \in H$. In practice, the connection from states to atomic propositions is implemented by linear inequality constraints. Note that $\theta \in \{0,1\}^{k_1}$ in (1) is the value of the atomic proposition, while $\theta \in AP$ above is the atomic proposition, so they are not exactly the same, but we express them by using the same $\theta$.

**Definition** *4.2:* (LTL formula) *An LTL formula on AP is a sentence that consists of atomic propositions and operators of LTL and obeys the grammar of LTL.*

Also, the variables $p$ in an MLD system are LTL propositions that consist of (atomic) propositions. Such $p$ behave like logic variables but actually are continuous variables [7].

In summary, we use MLD to model the real world and LTL to define the behaviors of a robot. LTL propositions consist of atomic propositions, which are like "$x \in H$." In practice, we give a system dynamics and an LTL proposition $\phi$, where we let $\phi$ be true; then, we generate the robot's behavior. This is done by mixed integer programming. For details, see [7].

## IV. PICK AND PLACE TASK WITH LTL SPECIFICATIONS

This section describes the equations for modeling a picking motion, LTL specification and the algorithms for implementing object transportation.

### A. Modeling

In this subsection, we describe the state-augmented hybrid system, use of atomic propositions, picking model, LTL formulation, and rules of picking and carrying.

*1) Simple LTL by Hybrid System:* The most important point is to have the states of the arms and objects as the state of the system. This enables definition of simple operations as shown in Fig. 2.

Then, the state of the system is augmented in the following way:

$$\begin{aligned} x &= \begin{bmatrix} X'_{\mathrm{a}1} & X'_{\mathrm{a}2} & X'_{\mathrm{o}1} & \dots & X'_{\mathrm{o}n_o} \end{bmatrix}', \\ X'_{\mathrm{a}i} &= \begin{bmatrix} x'_{\mathrm{arm}i} & \dot{x}'_{\mathrm{arm}i} \end{bmatrix}, \\ X'_{\mathrm{o}j} &= \begin{bmatrix} x'_{\mathrm{obj}j} & \dot{x}'_{\mathrm{obj}j} \end{bmatrix}, \end{aligned} \tag{5}$$

where the $x_{\mathrm{arm}*}, \dot{x}_{\mathrm{arm}*}$ are states of the arms, and the $x_{\mathrm{obj}*}, \dot{x}_{\mathrm{obj}*}$ are the states of objects. We assume that $x_*$ is position and $\dot{x}_*$ is velocity. $n_o$ is the number of the objects (see sub-subsection IV-B.1).

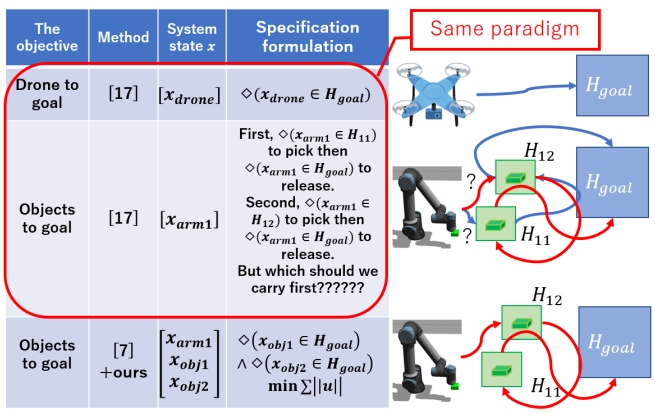


Fig. 2. Examples of action definition using LTL: In the first drone example, a simple LTL statement specifies the position of the drone, which can be directly controllable. Next, in the pick and place example, the state that can be input directly is the hand state, which requires a complicated specification. The complete LTL, in that case, is more complicated and not unique. This is a disadvantage of directly applying the conventional method. Finally, if the state of the object is also included in the system state, however, then a simple LTL specification is possible. Drone by https://www.irasutoya.com.

*2) Linear Constraints for Atomic Propositions:* The connection from system states to atomic propositions is made by a linear constraints as defined in [18]. The important point is that these constraints not only reduce the number of discrete variables (the main idea in [18]) but also are convenient for modeling picking because of their secondary effect, as shown in Fig. 3.

First, we define an $m-$dimensional convex polytope set $H$ $(m < n)$ for an atomic proposition $\theta =$ "$x_{\mathrm{sub}} \in H$", where $x_{\mathrm{sub}}$ is a part of the state $x$ so $x_{\mathrm{sub}} \in \mathbb{R}^m$. We assume $H$ is consists of $e$ hyperplanes. Let the column vectors of $a \in \mathbb{R}^{m \times e}$ be normal vectors to the hyperplanes that make up $H$, $o \in \mathbb{R}^{m \times e}$ are offsets from the origin, and $x_0 \in \mathbb{R}^m$ is the origin. Then, we define $H$ as

$$H = \{x_{\mathrm{sub}} \in \mathbb{R}^m : a'(x_{\mathrm{sub}} - x_0(k)) \leq a \cdot o\}. \tag{6}$$

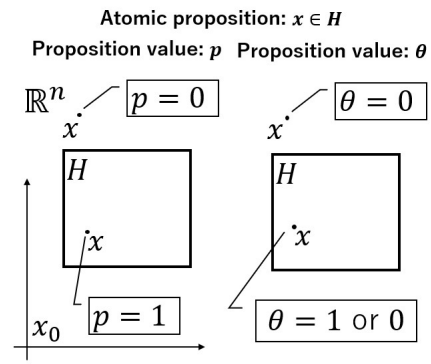


Fig. 3. Difference in atomic proposition between [7] and [18]: This figure shows the relationship between "$x \in H$" and $p, \theta$, for the state $x$ of any dimension and the set $H$. In the definition in [7], the value $p$ for the atomic proposition $x \in H$ is as shown on the left side of the figure, namely, $p = 1 \leftrightarrow x \in H$. In the definition in [18], however, the logical value is not fixed even if $x \in H$, as shown on the right side, giving $\theta = 1 \rightarrow x \in H$. This property is useful for modeling picking. Usually the notation $p =$ "$x \in H$" means $p = 1 \leftrightarrow x \in H$, but even for convenience, we also use $\theta =$ "$x \in H$" for $\theta = 1 \rightarrow x \in H$.

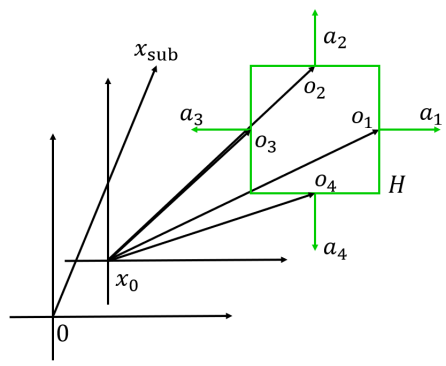


Fig. 4. Example of a set $H$: The $a_i$ are normal vectors to the edges, and the $o_i$ are their offsets. $x_0$ is the origin of $o_i$ and can move over time.

For example, Fig. 4 shows a rectangle in 2D space.

Next, to connect the system state $x$ to the atomic proposition value $\theta \in \{0, 1\}$, we use linear constraints as defined in [18]:

$$\begin{aligned} & a'(x_{\mathrm{sub}} - x_0) \leq a \cdot o + M(1-\theta)\mathbb{1} \Leftrightarrow \\ & \because \bar{H} = a'(S_1 x - S_2 x) \text{ for matrices } S_* : \mathbb{R}^n \rightarrow \mathbb{R}^m \\ & \bar{H}x \leq \bar{K} + M(1-\theta)\mathbb{1} \Leftrightarrow \\ & \bar{H}x + M\theta\mathbb{1} \leq \bar{K} + M\mathbb{1}, \end{aligned} \tag{7}$$

where $\bar{H} \in \mathbb{R}^{e \times n}$, $\bar{K} \in \mathbb{R}^e$, and $M \in \mathbb{R}$ is a large number. In practice, typically $x_{\mathrm{sub}}$ is $x_{\mathrm{arm}*}$ or $x_{\mathrm{obj}*}$. In equation (7), $\theta = 0$ when a state $x$ is not in set $H$, but when $x$ is in $H$, the constraint is not broken regardless of whether $\theta = 0$ or $\theta = 1$. Therefore, the logical value is not fixed as shown in Fig. 3. This is useful for modeling the behavior "the arm grabs the object" as described below and illustrated in Fig. 5.

*3) Picking Model:* Next, we propose a modeling method for picking. By making the object's velocity coincide with that of the arm when the state of the hand is in the object's bounding box, we model picking as shown in Fig. 5.

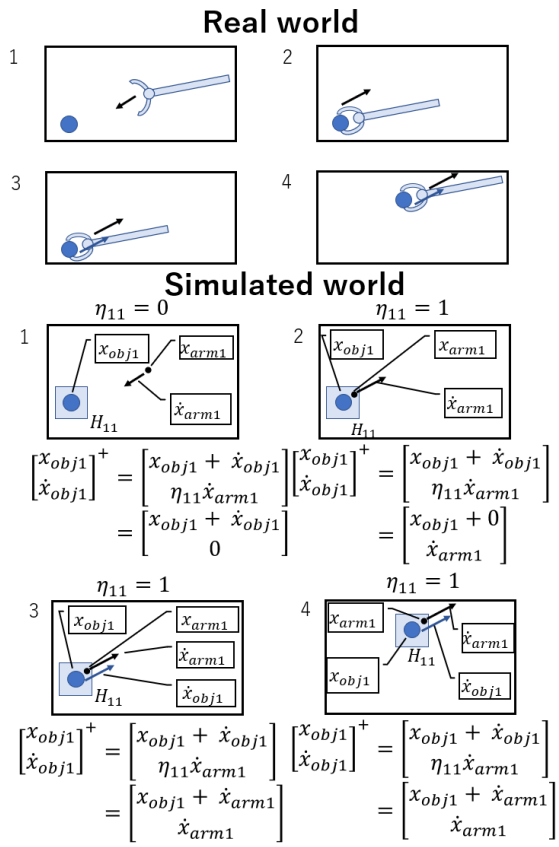


Fig. 5. Picking model: Let $x_{\text{arm1}}$ and $x_{\text{obj1}}$ be the hand position and object position, respectively. Let $H_{11}$ be the bounding box of the object and proposition $\eta_{11}$ be $x_{\text{arm1}} \in H_{11}$. When the hand enters the object area $H_{11}$, $\eta_{11}$ becomes 1, which activates the velocity constraint of the object, and causes it to follow the hand. This is the picking model. Note that, even after the object is gripped (as seen on the bottom right), the hand can release it for the reason shown in Fig. 3. In practice, the optimizer determines when to grasp and release.

Let $\eta_{ij} \in \{0, 1\}$ be a logic input indicating that "hand $i$ grabs object $j$"; then, the hybrid system models a picking motion as a jump behavior:

$$\eta_{ij} = 1 \rightarrow \dot{x}_{\text{arm}i} = \dot{x}_{\text{obj}j}, \tag{8}$$

where $x_{\text{arm}i}$ is the state of the hand of arm $i$, and $x_{\text{obj}j}$ is the state of object $j$.

In the MLD system, changes in dynamics are expressed in the following way:

$$x^+ = \eta(A^{\eta\text{ON}}x + B^{\eta\text{ON}}u) + (1-\eta)(A^{\eta\text{OFF}}x + B^{\eta\text{OFF}}),$$

where $A^{\eta\text{ON}}, B^{\eta\text{ON}}$ are system dynamics when $\eta = 1$ and $A^{\eta\text{OFF}}, B^{\eta\text{OFF}}$ when $\eta = 0$. By choosing them, we can describe the jump behavior in equation (8). For example, it can be represented in the following form (see Fig. 5):

$$\begin{bmatrix} x'_{\text{obj1}} & \dot{x}'_{\text{obj1}} \end{bmatrix}'^{+} = \begin{bmatrix} x_{\text{obj1}} + \Delta T \dot{x}'_{\text{obj1}} & \sum_i \eta_{i1}\dot{x}'_{\text{arm}i} \end{bmatrix}, \tag{9}$$

where $\Delta T$ is the discretization step size, which is 1 in this paper.

Because picking should be done when the hand is inside the polytope representing the object, the constraint of $\eta_{ij}$ should satisfy the following proposition:

$$\eta_{ij} = 1 \rightarrow \text{“}x_{\text{arm}i} \in H_{ij}\text{”}. \tag{10}$$

Here, $H_{ij}$ is a polytope on the space of $x_{\text{arm}i}$ originating $x_{\text{obj}j}$ (so the dimensions of $x_{\text{arm}i}$ and $x_{\text{obj}j}$ must match), defined as

$$H_{ij} = \{x_{\text{arm}i} \in \mathbb{R}^{2 \text{ or } 3} : a'_{ij}(x_{\text{arm}i} - x_{\text{obj}j}) \leq a_{ij} \cdot o_{ij}\}, \tag{11}$$

where the dimensions of $a_{ij}$ and $x_{\text{arm}i}$ must match and we consider 2D or 3D space so the dimension of $x_{\text{arm}i}$ is 2 or 3. (10) is described by linear constraints in (7) by treating $\eta$ with an atomic proposition. If the $\eta_{ij}$ are fixed when the state is in $H_{ij}$ (i.e., $\eta_{ij} = 1 \leftrightarrow$ "$x_{\text{arm}i} \in H_{ij}$"), it becomes impossible to release objects once they are grabbed.

### B. Specifications

*1) LTL Formulation:* Here, in anticipation of section IV-C.1, we define an LTL formula that means an action to transport $m_o$ objects out of $n_o$ objects to a destination, where $n_o$ is the number of objects to be included in the calculation and $m_o$ is the number of objects to be delivered. The LTL formula consists of atomic propositions about the objects, i.e., $\theta_{j+m_o n_o} = x_{\text{obj}j} \in H_j$, for

$$H_j = \{x_{\text{obj}j} \in \mathbb{R}^{2 \text{ or } 3} : a'_j x_{\text{obj}j} \leq a_j \cdot o_j\}, \tag{12}$$

which means "object $i$ is in the destination region." Note that $\theta_1$ to $\theta_{m_o n_o}$ are the $\eta_{ij}$. Hence, "transport $m_o$ objects out of $n_o$ objects" is equivalents to the LTL formula specifying that "any $m_o$ variables of $\theta_{1+m_o n_o}, \ldots, \theta_{n_o+m_o n_o}$ are eventually true." That is, we have the following:

$$\phi_{\text{task}} = \bigvee_{cmb \in C} \bigwedge_{i \in cmb} \Diamond \theta_{i+m_o n_o}, \tag{13}$$

where $\mathbb{N}_{n_o} = \{1, \ldots, n_o\}$, and the set of all $m_o$-combinations of the set $\mathbb{N}_{n_o}$ is $C = \binom{\mathbb{N}_{n_o}}{m_o} \subset 2^{\mathbb{N}_{n_o}}$ [20].

*2) System Constraints for Picking and Carrying:* It is difficult for a real robot to grasp multiple objects, and grasping one object with both arms is not efficient. Moreover, we must prohibit a robot's arms from crossing, so we always incorporate the following as linear constraints of the system (i.e., $C, D_*, E$ in (1)). Recall here that $\eta =$ "hand $i$ grabs object $j$".

- Each arm does not grab the same object at the same time:

$$\sum_i^2 \eta_{ij}(k) \leq 1 \quad (j = 1, 2, \ldots, n_o, k \in \mathcal{T}). \tag{14}$$

- The arms do not grab other objects at the same time:

$$\sum_j^{n_o} \eta_{ij}(k) \leq 1 \quad (i = 1, 2, k \in \mathcal{T}). \tag{15}$$

- The $x$ position of the left arm is less than that of the right arm:

$$x_{\text{arm}2x} + 0.1 < x_{\text{arm}1x} \tag{16}$$

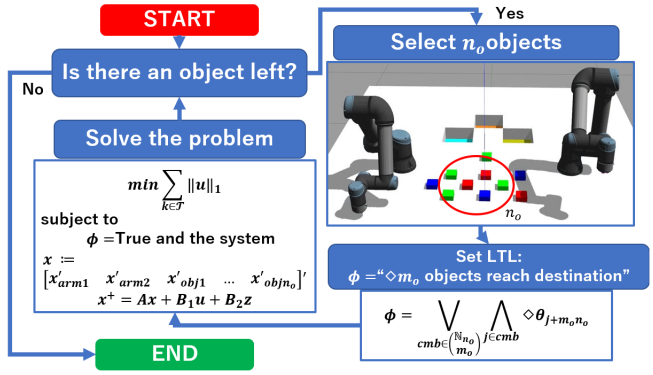


Fig. 6. Sequential transportation: To reduce the calculation time, the task of carrying several objects is repeated. Let $n_o$ be the number of objects to be included in the calculation and $m_o$ be the number of objects to be delivered ($m_o = 2$ is reasonable for the dual-arm case). Then, $\theta_{j+m_o n_o}$ is an atomic proposition that "object $j$ is at the goal." If there are objects left, choose $n_o$ and generate an LTL statement like "$\Diamond m_o$ objects reach their destination," e.g., $\phi = (\Diamond\theta_{1+2n_o} \wedge \Diamond\theta_{2+2n_o}) \vee \ldots \vee (\Diamond\theta_{2+2n_o} \wedge \Diamond\theta_{3+2n_o}) \vee (\Diamond\theta_{2+2n_o} \wedge \Diamond\theta_{4+2n_o}) \vee \ldots \vee (\Diamond\theta_{n_o-1+2n_o} \wedge \Diamond\theta_{n_o+2n_o})$. Then, solve the problem repeatedly. The LTL looks complex but can be generated very systematically.

Up to this point, we have seen the relationship between the continuous and discrete variables and how to determine the state of the system. Once the dynamics of the hand and the object are determined (double integrators in our work) , the system matrices $A, B_*, C, D_*, E$ are obtained according to the typical procedure of an MLD system [19].

### C. Algorithms

In this subsection, we describe the sequential transportation and pruning algorithms. The sequential transportation algorithm reduces the calculation time for the robot carrying objects sequentially, while the pruning algorithm reduce the time required for optimization.

*1) Sequential Transportation:* We first propose an algorithm that speeds up the computation for transportation. It takes much time to calculate the optimal solution when carrying all objects to the destination (i.e., $m_o = n_o$), because the numbers of calculation steps ($T$) and discrete variables become large. To solve this problem, we reduce the number of objects carried at one time (to $m_o = 2$, in this case) and repeatedly solve the optimization problem, as shown in Fig. 6.

The algorithm is listed below as Algorithm 1. $n_o$ can be equal to the number of all remaining objects, but in that case, the calculation takes more time.

**Algorithm 1** Sequential Transportation

1: **while** there exist objects to carry **do**
2: Select $n_o$ objects from the remaining objects
3: Define the system matrices, variables, and LTL from $n_o$ and $m_o$
4: $T$=predictT(system)
5: check_feasibility(system, $T$)
6: solve(system, $T$)
7: **end while**

*2) Pruning:* In general, because it takes time to solve a MILP problem, we must reduce the number of discrete variables as much as possible. As the value of $\theta_*$ can be predicted to some extent from the initial position of the objects or hand, it is used to reduce the number of discrete variables and speed up the calculation.

For $\eta_{ij}$, the time taken for hand $i$ to reach object $j$ by linear movement can be calculated, and for the remaining $\theta_{j+m_o n_o}$, the time required for object $j$ to reach the destination can be calculated as well. For example, in the case of one arm and one object, assume that it takes $T_\eta$ steps to grab object 1 and $T_\theta$ steps for the object to reach the destination. Then, we have $\theta_1(0 : (T_\eta - 1)) = 0$ and $\theta_2(0 : (T_\theta - 1)) = 0$. By giving this as a constraint, the calculation time can be reduced. $T_\theta$ may be used to predict the step $T$.

## V. IMPLEMENTATION AND CASE STUDIES

In this section, we apply the proposed algorithms in two case studies. First, we show that the pick and place problem can be solved with these algorithms. Second, we analyze the method's performance for different rules and various parameters. All computations were performed on an Intel® Xeon™ E3-1230, 3.30 GHz. All MILP problems were solved by using CPLEX[21] with MATLAB® as an interface.

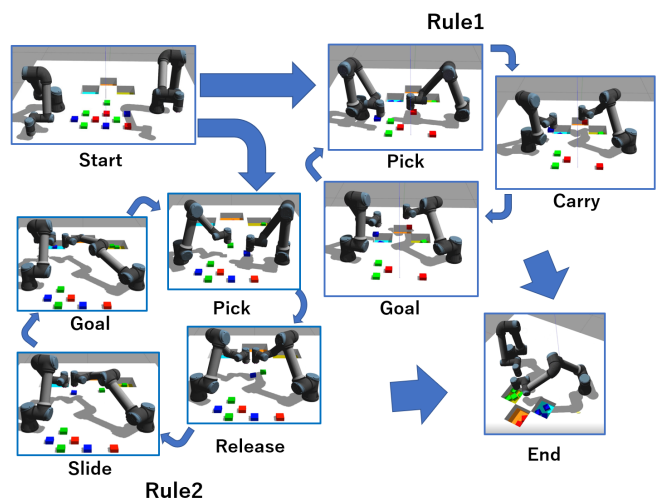


Fig. 7. Overall picture: The upper left is the initial state and the lower right is the end state. In rule 1, grasping continues to the goal. The object is first gripped and then transported, finally reaching the goal. In rule 2, the arm does not reach the goal. First, the blue and green objects are gripped, then released immediately, causing them to slide and be thrown to the goal. The important point is that the trajectory is automatically obtained without any instructions from the user, even when the arm does not reach the goal. Equations (18-20) represent safety instructions and do not indicate reachability, so we can calculate the trajectory automatically without them.

### A. Settings

We consider the task of transporting objects, as mentioned in section I-D, in a 2D system consisting of two arms, 10 objects, and 3 goals. The colors of the objects and goals match, as shown at the end of Fig. 7.

We consider input $u$ as the force applied to the hands, with the following cost function [22]:

$$J = \sum_{k \in \mathcal{T}} (||u_1(k)||_1 + ||u_2(k)||_1), u_1, u_2 \in \mathbb{R}^2. \tag{17}$$

### B. Common Sense Rules of Transporting

As mentioned in section I-B, it is a strength of definition in LTL that it is possible to describe safety or handling rules and have them memorized as common sense. Real robots have difficulty in throwing objects, and sometimes we want robots to carry fragile objects, so the action of not throwing objects is considered as common sense. Therefore, we represent this kind of common sense via the following safety rules written in LTL (18-20). We then confirm that the robot's operation can be changed with these concise LTL expressions.

*1) Rule 1:* Throwing is not allowed (i.e., arms continue to grasp objects to the goal region).

$$\phi_{\text{nothrow}} = \Box(\eta_{ij} \wedge \bigcirc\neg\eta_{ij} \ \rightarrow \ \bigcirc\theta_{j+2n_o}). \tag{18}$$

This means the following: Always, if grasping ($\eta_{ij}$) now and then releasing ($\bigcirc\neg\eta_{ij}$), then the object is in the goal ($\bigcirc\theta_{j+2n_o}$).

*2) Rule 2:* Arms can throw objects but we limit some behaviors.

- No juggling (an arm does not give objects to another arm):

$$\phi_{\text{nojuggle}} = \Box((\eta_{1j} \rightarrow \bigcirc\neg\eta_{2j}) \wedge (\eta_{2j} \rightarrow \bigcirc\neg\eta_{1j})). \tag{19}$$

- No swapping (the arm must not grab and release at the same step):

$$\phi_{\text{noswap}} = \Box(\eta_{ij} \ \rightarrow \ \bigwedge_{l \neq j}^{n_o} \bigcirc\neg\eta_{il}). \tag{20}$$

Here, (20) means the following: Always, if object $j$ is grasped now, then releasing $j$ and grasping $l(\neq j)$ simultaneously is not allowed.

### C. Validation of Proposed Algorithm

For this validation, Table I lists the positions of the arms, objects, and goals, and the hand movement range. We applied the limitations described in section V-B.2, because the arms could not reach the goal regions. The LTL parameters were $m_o = 2$, $n_o = 6$, giving the following LTL specification for each step:

$$\begin{aligned} \phi &= \phi_{\text{task}} \wedge \phi_{\text{nojuggle}} \wedge \phi_{\text{noswap}} \\ \phi_{\text{task}} &= \Diamond \text{“any 2 objs of 6 objs reach destination”}, \end{aligned} \tag{21}$$

where the exact LTL representation of $\phi_{\text{task}}$ followed (13). We obtained optimal trajectories, and snapshots taken during the task are shown for Rule 2 in Fig. 7. The calculation times for each step were $2.38s$, $5.76s$, $6.07s$, $2.22s$, and $0.39s$.

The significant point is that which object to carry with which arm and the timing of grasping and releasing were successfully calculated. Furthermore, as seen in Fig. 7, the "throw" action was automatically calculated even if the arms could not reach the goal regions. Actually, the system (V-B.2) consisted of (19) and (20), which do not directly imply throwing action, but that action was automatically generated.

As you can see in the attached video, sometimes the point of release looks not consistent. The point of release and which object to be grasped, are determined by the cost function which only includes input $u_1, u_2$ and the states follow the Newton's law, therefore sometimes it seems strange. To deal with this, we should use a cost function like $J = \sum(||u|| + ||x||)$ but for easy implementation, we use (17) in our work.

### D. Performance Analysis

Object positions were generated via a uniform random number within $[-0.25, 0.25]^2$. Table I lists the goal regions, which the arms could reach. We solved the problem for both rules (V-B.1 and V-B.2) with various parameters $m_o, n_o$. Table II lists the average total computational time and cost function value for runnning the simulation 10 times. The proposed algorithms were effective even if the $n_o$ parameters changed. As mentioned in section IV-C.1, however, $m_o$ becoming close to $n_o$ made $T$ large and increased the calculation time.

TABLE I
SETTINGS

| Initial positions of hands, objects and arm base | | | |
|---|---|---|---|
| Item | Position | Item | Position |
| left arm base | $(-0.3, 0.3)$ | object4 | $(-0.2, 0.2)$ |
| right arm base | $(0.3, 0.3)$ | object5 | $(0.0, 0.2)$ |
| left hand | $(-0.1, 0.0)$ | object6 | $(0.2, 0.2)$ |
| right hand | $(0.1, 0.0)$ | object7 | $(0.2, 0.0)$ |
| object1 | $(0.1, 0.1)$ | object8 | $(-0.2, 0.0)$ |
| object2 | $(-0.1, 0.1)$ | object9 | $(-0.3, 0.1)$ |
| object3 | $(0.0, 0.0)$ | object10 | $(0.0, 0.4)$ |

| Movable range of the hand $((x1, x2) \times (y1, y2))$ |
|---|
| $(-0.9, 0.9) \times (0, 0.9)$ |

| Goals | |
|---|---|
| For V-C | |
| Goal color | Range |
| red | $(0.225, 0.575) \times (1.15, 1.45)$ |
| green | $(-0.175, 0.175) \times (1.35, 1.65)$ |
| blue | $-(0.575, 0.225) \times (1.15, 1.45)$ |
| For V-D | |
| Goal color | Range |
| red | $(0.225, 0.575) \times (0.65, 0.95)$ |
| green | $(-0.175, 0.175) \times (0.85, 1.15)$ |
| blue | $-(0.575, 0.225) \times (0.65, 0.95)$ |

TABLE II
MEAN SUMS OF COST FUNCTION VALUE AND CALCULATION TIME

| ($m_o$,$n_o$) | Rule 1 | | Rule 2 | |
|---|---|---|---|---|
| | cost function | time (s) | cost function | time (s) |
| (2,5) | 9.03 | 2.52e+00 | 8.79 | 1.13e+01 |
| (2,6) | 9.70 | 2.54e+00 | 9.07 | 1.17e+01 |
| (2,10) | 9.80 | 2.78e+00 | 9.04 | 1.69e+01 |
| (3,5) | 12.68 | 2.70e+03 | 8.96 | 1.97e+01 |
| (3,6) | 12.51 | 3.64e+03 | 8.81 | 2.64e+01 |

## VI. CONCLUSIONS

In this paper, we have shown how to automatically generate pick and place motion commands as an example of handling operations with abstract motion definition using LTL. First, by using a hybrid system, it is possible to define simple LTL operations. Second, by modeling picking, it is possible to simulate gripping and releasing correctly. Finally, a sequential transport algorithm enables calculation within a reasonable time.

It is important that the specifications are indirect. In the conventional method, the behaviors of controllable states are indicated by LTL. In this paper, however, by treating a state of a target object as a system state and configuring it as a hybrid system, the target of a state that is not necessarily controllable all the time can be indicated by simple LTL rules, and the problem is solved by "thinking." This method differ from the method in which the hand coordinates of a manipulator are programmed manually. In fact, we found that, even when the hands could not reach the destination area, the action of sliding an object to deliver it was automatically generated.

As future work, further reduction of the calculation time will be necessary. In this paper, objects are carried sequentially, but from the viewpoint of overall optimization, it is better to calculate all trajectories in one calculation. Therefore, further reduction in the number of discrete variables will also be required and can be achieved by improving the model and incorporating learning. In addition, confirmation of effectiveness of our method in situations where dual-arm manipulation is more required is included.

## ACKNOWLEDGMENTS

We thank NEC Corporation, and Mr. Yano at Tokyo Institute of Technology for helping to improve the robot simulation environment.

## REFERENCES

[1] S. Kawamura, "Soft Robotics for Industrial Handling", Plenary talk, AIM 2019.

[2] R. Zollner et. Al, "Towards Cognitive Robots: Building Hierarchical Task Representations of Manipulatons from Human Demonstration", Proc. of ICRA, 2005.

[3] T. Abbas, B. A. MacDonald, "Generalizing Topological Task Graphs From Multiple Symolic Demonstration in Programming by Demonstration (PbD)Process", Proc. of ICRA, 2011.

[4] J. Orkin, "Applying Goal-Oriented Action Planning to Games", In: AI Game Programming Wisdom 2. Charles River Media, 2003.

[5] H. Hoang, S. Lee-Urban and H. Muñoz-Avila, "Hierarchical Plan Representations for Encoding Strategic Game AI", Proc. of AIIDE, 2005.

[6] F. Kolbe, "Goal Oriented Task Planning for Autonomous Service Robot", Master Thesis, Dept. of Information Science, Hamburg University of Applied Sciences, 2013.

[7] S. Karaman, R. G. Sanfelice and E. Frazzoli, "Optimal Control of Mixed Logical Dynamical Systems with Linear Temporal Logic Specifications", Proc. of 17 th IEEE Conference on Decision and Control, pp.2117-2122, 2008.

[8] T. Nishi and Y. Mori, "Energy Efficient Motion Planning of Dual-Armed Robots with Pickup Point Determination for Transportation Tasks", Proc. of IEEE International Conference on Industrial Engineering and Engineering Management (IEEM), pp.1401-1405, 2018.

[9] S. Sadraddini and C. Belta, "Robust Temporal Logic Model Predictive Control", Proc. of 5 th Annual Allerton Conference, Allerton House, UIUC, Illinois, 2015.

[10] A. P. Nikora, and G. Balcom, "Automated Identification of LTL Patterns in Natural Language Requirements", Proc. of 10 th Int. Symposium on Software Reliability Engineering, 2009.

[11] S.Ghosh et. al., "SRSENAL:Automatic Requirements Specification Extraction from Natural Language", arXiv:1403.3142v3[cs.CL], 2016.

[12] Michael Bauland et.al., "The Tractability of Model-Checking for LTL:The Good, the Bad, and the Ugly Fragments", arXiv:0805.0498 [cs.LO], 2008.

[13] S. Dathathri, and R. M. Murray, "Decomposing GR(1) Games with Singleton Liveness Guarantees for Efficient Synthesis", Proc. 56 th IEEE Conference on Decision and Control, pp.2117-2122, 2017.

[14] C. Belta, B. Yordanov and E. A. Gol, "Formal Methods for Discrete-Time Dynamical Systems", Studies in Systems, Decision and Control, Springer, 2017.

[15] V. Nenchev, C. Belta, and J. Raisch, "Optimal motion planning with temporal logic and switching constraints", Proc. 14 th ECC, 2015.

[16] T. Ushio and A. Sakakibara, "Formal Control System Design-Temporal Logic Specification and Game Theoretical Approach" system/control/information, Vol.62, No.6, pp.215 - 220, 2018.(Japanese)

[17] P. Nilsson and A. D. Ames "Barrier Functions: Bridging the Gap between Planning from Specifications and Safety-Critical Control", Proc. of 57 th IEEE Conference on Decision and Control, 2018.

[18] E. M. Wolff and R. M. Murray "Optimal Control of Mixed Logical Dynamical Systems with Long-Term Temporal Logic Specifications", Tech. rep., California Institute of Technology, 2013.

[19] A. Bemporad and M. Morari, "Control of systems integrating logic, dynamics, and constraints", 35 th Automatica, pp.407 - 427, 1999.

[20] R. P. Stanley, "Enumerative Combinatorics", Vol. I, second edition, pp.13-15, 2011.

[21] "IBM ILOG CPLEX Optimization Studio CPLEX User's Manual", Version 12, Release 6, 2015.

[22] M. G. Earl and R. D'Andrea, "Iterative MILP Methods for Vehicle-Control Problems", IEEE Transactions on Robotics, Vol. 21, No.6, pp.1160-1161, December 2005.