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

Generating New Lower Abstract Task Operator using Grid-TLI

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

***Abstract*— We propose a method of subdividing robot tasks into new lower abstract tasks. The description of robot tasks in an abstract manner is effective for motion planning for complex tasks and teaching robot movements in various environments. However, a more efficient task description may be obtained by using a lower abstraction according to the work environment. We argue that a higher abstract task can be expressed as a new lower abstract subtasks by applying Grid-based Signal Temporal Inference (Grid-TLI). We show that a new task can be completed using the Signal Temporal Logic formula for each cluster. We demonstrated the efficiency of our method through computer simulations using a 2-D security robot task.**

I. INTRODUCTION

A. Background

Robots are widely used not only in industrial areas such as factories, but also in commercial areas. Even in the commercial area where robots have not been used in the past, the movement to introduce robots is becoming popular, but the difficulty of teaching tasks to robots in a real environment is a major barrier to introducing robots. The major factors are that it is not user-friendly to manually rewrite the motion plan according to the robot's work environment with conventional methods, difficulties increase in proportion to the complexity of the task, and it is necessary to conduct operation planning by considering safety.

One approach that simplifies the teaching tasks is describing a complicated task by dividing it into simple tasks such as Hierarchical Task and Motion Planning (TAMP) [1][2][3]. With this method, a task is expressed by a pre-condition and an end-condition after the task execution, and a complex task is completed by finding a task sequence that can reach a desired end condition from the current state. These methods can separate the task into high-level task planning and low-level controller design. They have the advantage that task planning becomes easier, and that complex tasks are represented by simple task sequences, making them easier for the user to understand.

However, how to set the pre-conditions and end-conditions of simple tasks is a problem. The problem depends on the degree of abstraction of the task. This degree indicates the complexity of the pre-conditions and end-conditions of the task (e.g. the number of conditions). A higher task abstraction has an advantage in that a robot can easily respond to changes in its work environment, but it requires low-level controllers for having higher performance. Usually high-performance controllers require long computation times and powerful computers. Reducing the abstraction level can reduce the scale of the problem by adding information about the work environment to the task conditions [4], thus reducing the performance required for the low-level controllers. Therefore, it is effective to prepare a task group with a high degree of abstraction in advance and systematically reduce the degree of abstraction of the task according to the environment.

[1]S.Tokuda, [1]M.Katayama, and [1]M.Yamakita are with the Department of Systems and Control Engineering, Tokyo Institute of Technology, 1-11-1 Oh-Okayama, Meguro, 151-8551, Japan tokuda@ac.sc.e.titech.ac.jp, katayama@ac.sc.e.titech.ac.jp, yamakita@ac.sc.e.titech.ac.jp

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

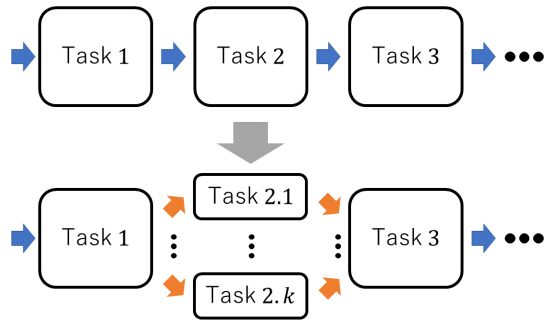

Fig. 1. Proposed method. Upper row is task sequence generated by high-level task planner. Lower row is task sequence using new lower abstract task generated with proposed method. It is effective to prepare task group with high degree of abstraction in advance and systematically reduce degree of abstraction of task according to environment.

B. Task Abstraction

There are cases in which the efficiency for operating tasks is poor when using the assumed task description. For example, consider a pick-and-place task for an object on a table. If there is a constraint in the work environment, for example, the left and right sides of the table are separated at the center of the table. The effects of the constraint on the task execution may differ depending on the arrangement of the objects. In this case, it is more efficient to divide the task into "pick the object on the right" and "pick the object on the left" with a lower abstract task description than to use the task with a higher abstract task description, "Picking an object".

C. Safety

Linear Temporal Logic (LTL) and Signal Temporal Logic (STL) have been proposed as task description methods to guarantee the safety of operations [5][6][7]. It is also expected that it will be possible to translate instructions in

natural language into LTL [8][9]. By describing tasks in LTL or STL, it is possible to ensure safety by checking in advance that rules stored as common sense and that newly acquired task rules are consistent with each other. However, checking that the acquired tasks are consistent is generally very computationally expensive [10].

D. Contributions

We propose a method for generating new lower abstract tasks from higher abstract tasks. There are two main contributions in this paper:

1) Generation of new tasks using Grid-based Signal Temporal Inference (Grid-TLI)
   Our method generates new tasks by defining the trajectory cluster as new tasks by using Grid-TLI [11].
2) Trajectory optimization using Grid-TLI
   We propose a trajectory optimization method using Grid-TLI for low-level controller.

With the proposed method, we apply Grid-TLI to the trajectories of a robot obtained by the task description with a higher abstraction and obtain the clusters of the trajectories represented by STL. We show that each resulting cluster can be represented as a new task; consequently, one task can be represented as multiple lower abstract task descriptions.

There are several algorithms of trajectory classification using STL [12][13], but Grid-TLI is suitable for the proposed method because the trajectory does not need to be labeled, unlike other algorithms.

Since the task is expressed as an STL condition, we can combine it with other STL conditions obtained as prior information. We argue that this can be used as an initial guess for warm-starting trajectory optimization using the obtained STL conditions, and a higher abstract task can be executed efficiently.

II. RELATED WORK

There have been many studies on robot task planning [1][2][14][15]. There are roughly two types of low-level controllers: the planning algorithm and feedback control. The planning algorithm solves a task as a trajectory optimization problem to obtain the trajectory of the system that can complete the task [1][2]. Feedback control, however, sets a policy in advance so that the task can be completed using the input obtained by that policy [14][15]. In contrast to feedback control, the planning algorithm incurs high computational cost; while guaranteeing the optimality of the trajectory. Therefore, it is important to generate a good initial guess using prior information and perform warm-start.

Related methods that define the behavior of robots using LTL or STL have been proposed [5][16][17][18][19][20]. In [16], the motion planner uses LTL, but the path of the robot is generated by rapidly expanding random tree (RRT), and selects the one that satisfies the definition of LTL. In [5], the trajectory is calculated from the LTL and the given system dynamics without the help of the planning algorithm. However, since the mixed integer programming problem (MILP) is used as the optimization method, the system needs to be converted to a discrete time linear system. Advisory Temporal Logic Inference [20] generates STL conditions from successful and failed trajectories when the robot is operated by a human to assist the execution of the motion. With this method, STL conditions are used and the time-varying constraint conditions, such as moving obstacles, can be expressed. However, this method requires the labeling of the trajectory in advance, similar to trajectory classification using other TLIs [12][13].

With the proposed method, trajectories can be clustered using only executable trajectories by Grid-TLI. Grid-TLI has the advantage that clustering trajectories and generating STL conditions for each cluster can be done simultaneously. Also, by using a new task with a lower abstract task, this new task can be executed efficiently by carrying out warm-start or limiting the areas of the solution.

III. PRELIMINARIES

A. Notation

We consider a continuous-time system of the form:

$$\dot{x} = f(x, u) \tag{1}$$

$$y = g(x, u) \tag{2}$$

where $x \in X \subseteq \mathbb{R}^{n_x}$ are the continues states, $u \in U \subseteq \mathbb{R}^{n_u}$ are control inputs, and $y \in Y \subseteq \mathbb{R}^m$ are outputs. For $t_1,\, t_2 \in \mathbb{R}$, we write the interval $[t_1,\, \infty)$ as $\mathbb{R}_{\geq t_1}$ and $[t_1,\, t_2]$ as $\{t \in \mathbb{R} \,|\, t_1 \leq t,\, t \leq t_2\}$. We denote a trajectory of a system as $s \in S$, $s : \mathbb{R}_{\leq 0} \to \mathbb{R}^n$ and the set of them as S.

B. Task Description

As in a previus study [14], we represent a high-level task using three conditions: entry L_P, runnable L_R, and expected L_E. The L_P condition is necessary for operating a task, L_R continues the task motion, L_E is expected to be satisfied after task execution. For example, the "$Picking$" task for a manipulator is described as

$Picking$:

$$\begin{aligned} L_P :&\ \ Object\ Exists,\ Hand\ is\ Empty \\ L_R :&\ \ Object\ Exists,\ Hand\ is\ Empty \\ L_E :&\ \ Object\ is\ in\ Hand. \end{aligned}$$

C. Optimization-based Motion Planning

Optimization-based Motion Planning is a method of using a trajectory optimization method as a low-level controller. The L_p, L_R, and L_E conditions of each task are converted into the following inequality constraint:

$$L_P \quad \to \quad h_P(x(0)) = 0,\, g_P(y(0)) \leq 0 \tag{3}$$

$$L_R \quad \to \quad {}^{\forall}t,\, h_R(y(t)) = 0,\, g_R(y(t)) \leq 0 \tag{4}$$

$$L_E \quad \to \quad h_E(y(t_E)) = 0,\, g_E(y(t_E)) \leq 0, \tag{5}$$

where t_E is the terminal time of the task, h_P, h_R, h_E are functions of equality constraint that return vectors, and g_P, g_R, g_E are functions of inequality constraint that return vectors.

We can obtain the trajectory that completes the task by solving the following trajectory optimization problem:

$$\min_{x,\,u,\,t} \quad \left\{ \int_0^T c(x,\,u,\,t)dt + c_E(x(t_E)) \right\} \tag{6}$$

s.t.

$$\begin{aligned} &\dot{x} = f(x,u) \\ &y = g(x,u) \\ &h_P(y(0)) = 0,\, g_P(y(0)) \leq 0 \\ &h_R(y(t)) = 0,\, g_R(y(t)) \leq 0 \\ &h_E(y(t_E)) = 0,\, g_E(y(t_E)) \leq 0 \end{aligned} \tag{7}$$

where $c(x,\,u,\,t)$ is a stage cost function and $c_E(x(t_E))$ is a terminal cost function.

D. Signal Temporal Logic

1) Preliminary Definitions: we consider STL formulas defined recursively according to the following grammar:

$$\phi := \pi^\mu \,|\, \neg\,\mu \,|\, \phi_1 \,\wedge\, \phi_2 \,|\, \phi_1 \,\vee\, \phi_2 \,|\, \mathbf{G}_{[a,b]}\,\phi \,|\, \phi_1\,\mathbf{U}_{[a,b]}\,\phi_2 \tag{8}$$

where π^μ is an atomic predicate $X \to \mathbb{B}$ whose truth value is determined by the sign of a function $\mu : X \to \mathbb{R}$, and ϕ_1, ϕ_2 are STL formulas. The fact that the signal s satisfies an STL formula ϕ is denoted as $s \models \phi$. The $s \models \phi_1\,\mathbf{U}_{[a,b]}\,\phi_2$ if ϕ_1 holds every time before ϕ_2 holds, and $s \models \mathbf{G}_{[a,b]}\,\phi$ if ϕ holds every time between a and b. Formally, the validity of ϕ with respect to s at time t, denoted as $(s,\,t)$ is defined inductively as follows

$$(s,\,t) \models \pi^\mu \iff \mu(x(t)) > 0 \tag{9}$$

$$(s,\,t) \models \neg\phi \iff \neg(s,\,t) \models \phi \tag{10}$$

$$(s,\,t) \models \phi_1 \wedge \phi_2 \iff (s,\,t) \models \phi_1 \wedge (s,\,t) \models \phi_2 \tag{11}$$

$$(s,\,t) \models \phi_1 \vee \phi_2 \iff (s,\,t) \models \phi_1 \vee (s,\,t) \models \phi_2 \tag{12}$$

$$(s,\,t) \models \mathbf{G}_{[a,b]}\,\phi \iff {}^\forall t_l \in [t+a,\,t+b]\,,\,(s,t_l) \models \phi \tag{13}$$

$$\begin{aligned} (s,\,t) \models \phi_1\,\mathbf{U}_{[a,b]}\,\phi_2 \iff \\ {}^\exists t' \in [a+t,\,t+b]\ s.t.\ (s,t') \models \phi_2 \\ \wedge {}^\forall t'' \in [t,\,t']\,,\,(s,t'') \models \phi_1. \end{aligned} \tag{14}$$

The robust degree for STL is defined as a real-valued function ρ of s and t such that $(s,t) \models \phi \equiv \rho^\phi(x,t) > 0$ [21]. We call ρ^ϕ the rubustness function of ϕ. This is computed recursively. The complete robust semantics is defined as follows:

$$\begin{aligned} &\rho^\mu(s,t) = \mu(s(t)) \\ &\rho^{\neg\mu}(s,t) = -\mu(s(t)) \\ &\rho^{\phi_1 \wedge \phi_2}(s,t) = \min(\rho^{\phi_1}(s,t), \rho^{\phi_2}(s,t)) \\ &\rho^{\phi_1 \vee \phi_2}(s,t) = \max(\rho^{\phi_1}(s,t), \rho^{\phi_2}(s,t)) \\ &\rho^{\mathbf{G}_{[a,\,b]}\phi}(s,t) = \min_{t' \in [t+a,t+b]} \rho^\phi(s,t') \\ &\rho^{\phi_1 \mathbf{U}_{[a,\,b]} \phi_2}(s,t) = \max_{t' \in [a+t,\,b+t]} (\min(\rho^{\phi_1}(s,t'), \end{aligned} \tag{15}$$

$$\min_{t'' \in [t,\,t']} \rho^{\phi_2}(s,t'')). \tag{16}$$

E. Grid-based Signal Temporal Logic Inference

Grid-TLI [11] is an algorithm that generates STL formulas from a given set of training signals. We explain Grid-TLI for multi-dimensional signals.

1) Notations: The notation R is an $n+1$ -dimensional closed rectangular region given signal value codomain $\mathbb{R}^n$ and time domain $\mathbb{R}_{\geq 0}$. Region R is bounded by $[x^1_{\min},\, x^1_{\max}] \times \cdots \times [x^n_{\min},\, x^n_{\max}] \times [0, t_{\max}] \subset \mathbb{R}^n \times \mathbb{R}_{\geq 0}$, where $x^i_{\min},\, x^i_{\max} \in \mathbb{R},\, x^i_{\min} < x^i_{\max},\, i = 1, 2 \ldots, n$, and $t_{\max} \in R_{>0}$. Region R is partitioned into a grid made of rectangular cells. As in [11], we use the object oriented dot notation (".") to reference the properties. For example, the properties of R are described as $R.x_{\max},\, R.x_{\min},\, R.t_{\max},\, x_{\max} = (x^1_{\max},\, \ldots,\, x^n_{\max}),\, x_{\min} = (x^1_{\min},\, \ldots,\, x^n_{\min})$.

A cell g is a rectangular bounding box in $\mathbb{R}^n \times \mathbb{R}_{\geq 0}$ represented with a tuple $(x,\,t,\,x_{inc},\,t_{inc})$, where $(x,t) \in \mathbb{R}^n \times \mathbb{R}_{\geq 0}$ is the minimum point and minimum time defined as $x = (x^1,\,\ldots,\,x^n)$, s.t. ${}^\forall x',\, {}^\exists j,\, x^j < x'^j,\, t = min_{\tau \in g.I_T}\,\tau,\, g.I_T := [t,\,t+t_{inc}]$, and $x_{inc} \in \mathbb{R}^n_{>0}$ and $t_{inc} \in \mathbb{R}_{>0}$ are a set of the lengths of the sides of the box along the value and one of time dimensions. Therefore, g is bounded between $[x^i,\, x^i + x^i_{inc}] \times \ldots [x^n,\, x^n + x^n_{inc}]$ along the value dimensions and between $[t, t+t_{inc}]$ along the time dimension. As with R, the properties of g are described as $g.x,\, g.y,\, g.x_{inc},\, g.t_{inc},\, g.I_T$.

Given a g and a set of signals S, we define "g is covered by S" if and only if at least one signal $s \in S$ has a non-empty intersection with g and described as $g(s) \models \top$. The properties of s is described as $s.t_{\max},\, s.g$, where $s.t_{\max}$ is the end time of s and $s.g$ is the set of cells covered by s. We define the following property of an s, ${}^\forall t \geq s.t_{\max},\, s(t) := s(s.t_{\max})$.We denote the set of cells as Γ.

2) Algorithm of Grid-TLI: With Grid-TLI [11], all signal exists until $t_{\max}$. In our case, each signal has different end times. Therefore, we use an extended version of Grid-TLI [11].

Our Grid-TLI has four parameters: $x_t,\, t_t,\, c_x,\, c_t$, where $x_t = (x^1_t,\, \ldots, x^n_t) \in \mathbb{R}^n,\, t_t \in \mathbb{R}$ are the signal and time thresholds and define minimum values of $g.x_{inc},\, g.t_{inc}$ for all cells. The $c_x = (c_{x^1},\, \ldots, c_{x^n}) \in \mathbb{R}^n,\, c_t \in \mathbb{R}$ are the cluster threshold and affect the number of clusters.

Grid-TLI has four main steps: 1) find cells covered by each signal 2) cluster signals, 3) simplify the STL formula for each cluster, and 4) map the clusters to the final STL formula.

In the first step, we set the properties of R and unit cell g_u where $g_u(x,\,t,\,x_t,\,t_t)$, and find the cells coverd by each signal.

In the second step, we cluster the signals in S to k subset $(S_1,\,\ldots,S_k)$ by cluster thresholds $c_x,\, c_t$. We define "two signals s_1 and s_2 are in same cluster S_i" as that s_1 is in S_i and there exists $s' \in S_i$ such that s_2 and s' satisfy the following relations, for all $t \in [0,\, s_2.t_{\max}]$, there exists $g_p,\, g_q$ such that $t \in g_p.I_T,\, g_p(s_2) \models \top,\, t \in g_q.I_T,\, g_q(s') \models$

$\top$, and

$$\begin{cases} g_p.x^i - (g_q.x^i + g_q.x^i_{inc}) \leq c^i_x & \text{if } g_p.x^i \geq g_q.x^i, \\ g_q.x^i - (g_p.x^i + g_p.x^i_{inc}) \leq c^i_x & \text{if } g_q.x^i > g_p.x^i. \end{cases} \quad (17)$$

$$|s_2.t_{\max} - s'.t_{\max}| \leq c_t. \quad (18)$$

In the third and fourth steps, we use Algorithm 3 in [11]. Finally, we obtain STL formula $\Phi_i = \bigwedge_j G_{[\tau_{0j},\, \tau_{1j}]}(s(t) \in g_j)$ for each cluster and the complete formula assembled by disjunction of the formula obtained for each cluster: $\Phi = \Phi_1 \vee \Phi_2 \vee \cdots \vee \Phi_k$.

IV. PROPOSED METHOD

In this section, we present a proposed method for subdividing tasks. The flow of the proposed method is shown in Fig. 2. It consists of three steps. In the first step, we apply task planner to existing higher abstract tasks, and the trajectory for each task is generated using the low-level controller for various situations using the task sequence. In the second step, we apply clustering using Grid-TLI to the trajectories of each task generated in the first step. Finally, in the third step, the task is separated into new tasks using the STL conditions of the cluster obtained in the second step.

1) High-Level Task Planner and Low-Level Controller: The task planner uses a method in which tasks are represented with a higher abstraction. For example, suppose that tasks are represented in the Planning Domain Definition Language (PDDL) [22], LTL, and STL. The method for generating task sequence can use a method such as Stanford Research Institute Problem Solver (STRIPS) [23].

A. Clustering Task Trajectory

We apply Grid-TLI to the trajectories obtained in step 1. The number of clusters and the length of STL formula for each cluster depends on the parameters of Grid-TLI.

The STL condition for the obtained cluster Γ_j is $\Phi_j = \phi_{j1} \wedge \cdots \wedge \phi_{jn_j}$, $\phi_{jl} = G_{[\tau^j_{0l},\, \tau^j_{1l}]}\,(y \in g_l)$.

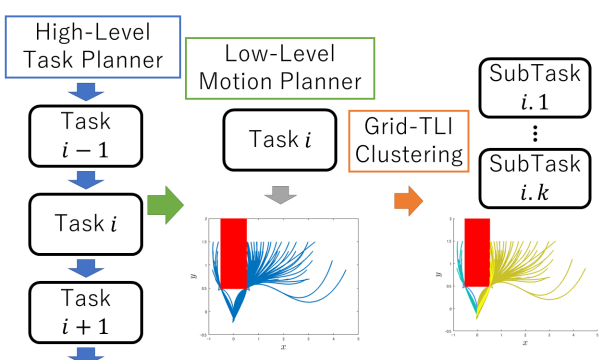

Fig. 2. Flow of the proposed method. The proposed method consists of three steps: 1) we apply task planner to existing higher abstract tasks, and trajectory for each task is generated using low-level controller for various situations using task sequence, 2) we apply clustering using Grid-TLI to trajectories of each task generated in first step, 3) task is separated into new tasks using STL conditions of cluster obtained in second step.

B. Using Subtask

Using the STL formula obtained in Step 2, the preconditions of the new task are expressed as follows:

$$\psi_j^{L^i_p} = \psi^{L^i_p} \wedge \phi_{j_1} \quad (19)$$

where $\psi^{L^i_p}$ is the precondition of task i. We define a set of initial condition of subtask in task i as $\Psi^i = [\psi_1^{L^i_p}, \ldots, \psi_k^{L^i_p}]$.

Algorithm 1 shows the optimization-based motion planning algorithm using the newly generated subtask. The flow is to first check which the subtask satisfies the initial condition, and obtain the trajectory of the task by solving the trajectory optimization problem using the central trajectory of the cluster of the subtask as the initial guess.

First, at line 1 of Algorithm 1, check whether the initial state belongs to the cluster from the initial condition of Eq. (19). Algorithm 2 shows Select_SubTask. t_0, y_0 is the initial time and initial output of the $Task$. Output of Select_SubTask γ^i can be considered in two cases. The first is when the number of elements of γ^i is 1 or more. In this case, the central trajectory of the cluster in γ_i with the largest value of robustness function $\rho_{set}(i)$ is used as the inital guess for the trajectory optimization problem. The second is when γ^i is 0. When γ^i is 0, y^i_0 means that the initial condition of any subtask is not satisfied. In this case, solve the trajectory optimization problem with the central trajectory of the cluster with the largest value of robustness function $\rho_{set}(i)$ as the inital guess. This allows us to efficiently obtain the trajectory that for completing the original task.

C. Approximation Grid-TLI

Expressing the STL condition as a continuous function is effective when used for control. The $\phi = G_{[\tau_{0j},\, \tau_{1j}]}\,(y \in g_j)$ is approximated using the following continuous function and constraints:

$$\begin{aligned} (t,\, y) &\models \phi \\ &\approx \mathcal{G}^{\phi}(t,\, y) \geq \delta > 0, \end{aligned} \quad (20)$$

$$\mathcal{G}^{\phi}(t,\, y) = S(\tau_{0j},\, \tau_{1j},\, \epsilon_t,\, t) \prod_{i=1}^{m} S(y_{l_i},\, y_{u_i},\, \epsilon_{y_i},\, y_i), \quad (21)$$

$$S(c_1,\, c_2,\, \epsilon,\, x) = \frac{1}{1 + e^{(-\alpha(x-(c_1-\epsilon)))}} \frac{1}{1 + e^{(\alpha(x-(c_2+\epsilon)))}}, \quad (22)$$

where ϵ_t, ϵ_{y_i}, δ are relaxation parameters. This function is $(t,\, y) \models G_{[\tau_{0j},\, \tau_{1j}]}\,(y \in g_j)$ with $(t,\, y)$, and $\mathcal{G}^{\phi}(t,\, y)$ is a function that returns ≈ 1. Using this function, the STL condition for Γ_i can be written as

$$\mathcal{G}^{\Phi_i}(t,\, y) = \sum_{j=1}^{n} \mathcal{G}^{\phi_{ij}}(t,\, y). \quad (23)$$

V. IMPLEMENTATION AND SIMULATION

We applied the proposed method to simulations of security robot tasks. In the simulations, we implemented the ICLOCS2 [24] framework in MATLAB® to solve the trajectory optimization problem, and used IPOPT [25].

Algorithm 1 Optimization-based Motion Planning Algorithm

Require: t_0, y_0, $Task$, Ψ, Γ
Ensure: x^*, u^*

$[\gamma, \rho_{set}] =$ Select_SubTask(t_0, y_0, Ψ)
if γ is empty **then**
  $index_{\rho_{\max}} = \arg\max_{i \in \{1, \ldots, k\}} (\rho_{set,i})$
else
  $index_{\rho_{\max}} = \arg\max_{i \in \gamma} (\rho_{set,i})$
end if
$\tilde{y} =$ Initial_Guess($\Gamma_{index_{\rho_{\max}}}$)
Slove x^*, u^* using the trajectory optimization warm-started by $\tilde{y}$ for $Task$

Algorithm 2 Select_SubTask

Input: t_0, y_0, Ψ^i
Output: γ, ρ_{set}

for $l := 1$ to L_i^k **do**
  if $y_0 \models \psi_j^{L_p}$ **then**
    $\gamma.insert(l)$
  end if
  $\rho_{set}.insert(\rho^{\phi_{j1}}(y_0, t_0))$
end for

A. System of Security Robots

Consider the task when a security robot finds a suspicious individual or object. When a suspicious object appears on the premises, the security robot needs to approach the object to confirm its identity and possibly eliminate it. Security robots often have areas into which they are prohibited from entering, such as a private area. We show the work environment of considering these situations in Fig. 3.

The security robots are indicated as blue triangles and suspicious objects are indicated as green triangles. The red rectangular area is the private area into which security robots cannot enter. The dynamics of the security robot are described as

$$\dot{x} = -v \sin(\theta) \tag{24}$$
$$\dot{y} = v \cos(\theta) \tag{25}$$
$$\dot{\theta} = \omega \tag{26}$$
$$\dot{v} = u_1 \tag{27}$$
$$\dot{\omega} = u_2, \tag{28}$$

where x, y are the coordinates of the robot, θ is the posture of the robot, v is speed of the robot, and ω is the angular velocity of the attitude angle of the robot.

Considering the case in which a suspicious object moves, the dynamics of the suspicious object are set as

$$\dot{x}_o = v_1 \tag{29}$$
$$\dot{y}_o = v_2, \tag{30}$$

where x_o, y_o are the coordinates of the object and v_1, v_2 are the speeds of the object, which were constant during simulation. We assumed $x_o(0) = -1$, $y_o(0) \in [0.5, 1]$, $v_1 \in [0, 1]$, $v_2 = 0$.

B. Task

We define the task of the security robot as follows:

$Catch$:

$$\begin{aligned} L_P &: \ Object\ Exists, \neg Enter\ Private\ Area \\ L_R &: \ \neg Enter\ Private\ Area \\ L_E &: \ Contact\ Object, \end{aligned}$$

where $\neg Enter\ Private\ Area$ in L_R indicates that the security robot does not always enter the private area, and L_E means that the security robot has caught a suspicious object at the end of the task. In other words, if a suspicious object is in the private area, the security robot cannot catch it.

C. Sampling Trajectory

Using the trajectory optimization method as a low-level controller, we generated the trajectory of the task when $y(0)$, v_1 was changed. The initial values of other states were $(x, y, \theta, v, \omega)(0) = (0, 0, 0, 0, 0)$, $x_o(0) = -1$, $v_2 = 0$. Since the optimization problem for this task tends to converge to the local infeasible point, it is necessary to use global search. Therefore, we repeatedly solved the optimization problem while changing the initial guess until a solution was obtained for the initial state. We needed to solve the trajectory optimization problem twice on average. The generated trajectory is shown in Fig. 4.

D. Clustering using Grid-TLI

We applied clustering using Grid-TLI to the sampled trajectories. The output was defined as $[x, y, \theta, v, \omega, x_o, y_o, v_1, v_2]$. Grid-TLI parameters were manually determined based on number of clusters and the number of STL conditions. The parameters used are shown in Table. I. We show the trajectories after clustering in Fig. 4. The trajectories were classified into eight clusters using these parameters. Then we checked the STL formula for each cluster. For example, the entry condition to v_1 of the yellow trajectory's cluster was $G_{[0,\,0.5]}(0.2 \leq v_1 \leq 1)$ and that of the brown trajectory's cluster was $G_{[0,\,0.5]}(0.0 \leq v_1 \leq 0.2)$. It shows that "the security robot could catch the suspicious object before the suspicious object reached the private area when the suspicious object was moving slowly, but when the suspicious object was fast, the security robot needed to catch the suspicious object after the object passed the private area". Therefore, we can consider new lower abstract tasks "catch before the object reaches private area" and "catch after the object passed the private area".

E. Using Subtask

The trajectory generated from Algorithm 1 using the obtained subtask is shown in Fig. 5 and supplemental video. The rectangular area shows the area that satisfies the STL condition of each subtask. A trajectory that completed the task is almost generated by solving the trajectory optimization problem once.

F. Using L_R of Subtask

We can consider an STL formula for each cluster as an L_R condition of the subtask. We compared Algorithm 1 and that using approximated Grid-TLI constraints of L_R conditions of the subtask. The average computation times of solving the trajectory optimization problem when generating initial state of y and v_1 at random were 6.21s and 9.15s, and the success rates to solve the trajectory optimization problem were 97% and 62%. The performance when using L_R depends on the relaxation parameters ϵ_t, ϵ_{y_i}, δ because approximated Grid-TLI constraints cause computational complexity.

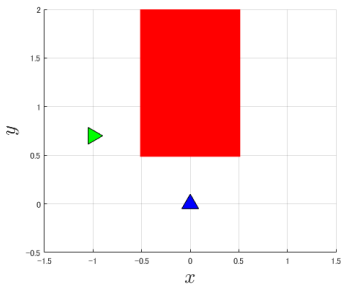

Fig. 3. Work environment of security robot. Security robot is indicated with blue triangle. Suspicious object is indicated with green triangle. Red rectangular area is private area into which security robots cannot enter.

TABLE I
PARAMETERS FOR GRID-TLI

t_{inc}	x_{inc}								
0.5	0.1	0.1	π/180	0.1	0.1	0.2	0.2	0.1	0.1
c_t	c_x								
4	0.2	0.2	π/18	0.2	0.2	1	1	1	1

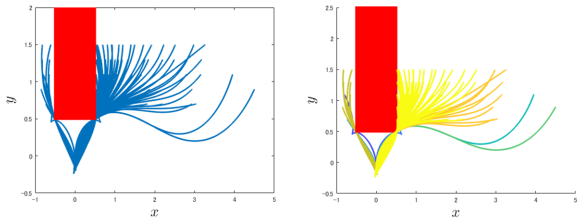

Fig. 4. Left : trajectories generated by solving optimization problem of higher abstract task. Right : trajectories after clustering using Grid-TLI. Trajectories are clustered into 8 clusters by using Grid-TLI. Trajectories of same color belong to same cluster.

VI. CONCLUSIONS

We showed that higher abstract tasks are subdivided into new lower abstract tasks by using the STL condition of each cluster generated by clustering the trajectories using Grid-TLI. The proposed method is effective for tasks that require a search of a global solution, such as the case study in discussed in Section 5. The proposed method may be also effective in situations where multiple agents execute tasks simultaneously. When multiple agents execute tasks at the same time, it is important to find the tasks of each agent efficiently.

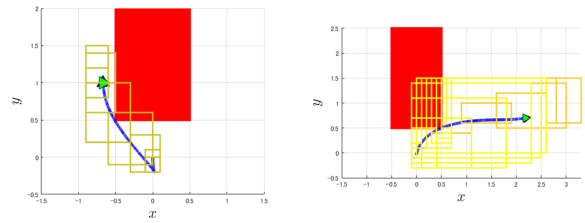

Fig. 5. Trajectories generated by solving warm-started optimization problem. Rectangles are STL conditions for cluster of subtasks. Left : trajectory that can be considered as new task "catch before the object reaches private area". Right : trajectory that can be considered as new task "catch after the object passes by private area". Initial output satisfies pre-conditions of two subtasks.

Future work is to automatically generate and optimize the parameters of Grid-TLI. And in the case study of this paper, the performance of proposed method with approximated Grid-TLI constraints of L_R is worse than without. Therefore we need to use more computationally efficient arroximation for L_R like [17]. In this paper, we used the trajectory optimization method as the low-level controller. We will also verify a case using feedback controller.

ACKNOWLEDGMENTS

We would like to thank NEC Corporation for support in this research.

REFERENCES

[1] L. P. Kaelbling, T. Lozano-Pérez, "Hierarchical task and motion planning in the now," IEEE International Conference on Robotics and Automation, 2011.
[2] M. Toussaint, "Logic-Geometric Programming: An Optimization-Based Approach to Combined Task and Motion Planning," International Joint Conferences on Artificial Intelligence, 2015.
[3] S. Srivastava, E. Fang, L. Riano, R. Chitnis, S. Russell, P. Abbeel, "Combined Task and Motion Planning Through an Extensible Planner-Independent Interface Layer," IEEE International Conference on Robotics and Automation, 2014.
[4] T. Lozano-Pérez, L. P. Kaelbling, "A constraint-based method for solving sequential manipulation planning problems," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2014.
[5] S. Karaman, R. G. Sanfelice and E. Frazzoli, "Optimal Control of Mixed Logical Dynamical Systems with Linear Temporal Logic Specifications," Proc. of 47th IEEE Conference on Decision and Control, 2008.
[6] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems. Springer, pp.152-166, 2004.
[7] S. Sadraddini and C. Belta, "Robust Temporal Logic Model Predictive Control," Proc. of 5 th Annual Allerton Conference, Allerton House, UIUC, Illinois, 2015.
[8] 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.
[9] S.Ghosh et. al., "SRSENAL:Automatic Requirements Specification Extraction from Natural Language," arXiv:1403.3142v3[cs.CL], 2016.
[10] 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.
[11] T. Lozano-Pérez, L. P. Kaelbling, "Grid-Based Temporal Logic Inference," IEEE 56th Annual Conference on Decision and Control, 2017.

[12] Z. Kong, A. Jones, A. I. M. Ayala, E. A. Gol, C. Belta, "Temporal Logic Inference for Classification and Prediction from Data," HSCC '14: Proceedings of the 17th international conference on Hybrid systems: computation and control, pp.273-282, 2014.

[13] Z. Xu, C. Belta, A. Julius, "Temporal Logic Inference with Prior Information: An Application to Robot Arm Movements," in Proc. IFAC Conf. Anal. Design Hybrid Syst. (ADHS), vol.48. no.27, pp.141-146, 2015.

[14] C. Paxton, N. Ratliff, C. Eppner, D. Fox, "Representing Robot Task Plans as Robust Logical-Dynamical Systems," IEEE/RSJ International Conference on Intelligent Robots and Systems, 2019.

[15] E. Pignat and S. Calinon, "Bayesian Gaussian Mixture Model for Robotic Policy Imitation," in IEEE Robotics and Automation Letters, vol. 4, no. 4, pp. 4452-4458, Oct. 2019.

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

[17] K. Garg and D. Panagou, "Control-Lyapunov and Control-Barrier Functions based Quadratic Program for Spatio-temporal Specifications", 58th IEEE Conference on Decision on Control, 2019.

[18] J. A. DeCastro, V. Raman, H. Kress-Gazit, "Dynamics-driven adaptive abstraction for reactive high-level mission and motion planning," IEEE International Conference on Robotics and Automation, 2015.

[19] Y. V. Pant, H. Abbas, R. Quaye, R. Mangharam, "Fly-by-Logic: Control of Multi-Drone Fleets with Temporal Logic Objectives," 9th ACM/IEEE International Conference on Cyber-Physical Systems, 2018.

[20] Z. Xu, S. Saha, B. Hu, A. A. Julius, "Advisory Temporal Logic Inference and Controller Design for Semiautonomous Robots," in IEEE Transactions on Automation Science and Engineering, vol.16, pp.459-477, 2019.

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

[22] M. Fox, D. Long, "PDDL2.1 : An Extension to pddl for Expressing Temporal Planning Domains," Journal of Artificial Intelligence Research 20, pp.61-124, 2003.

[23] R. E. Fikes, N. J. Nilsson, "STRIPS: A new approach to the application of theorem proving to problem solving," Artificial intelligence, vol.2, no.3-4, pp.189-208, 1971.

[24] Y. Nie, O. Faqir, E. C. Kerrigan, "ICLOCS2: Try this Optimal Control Problem Solver Before you Try the Rest," UKACC 12th International Conference on Control (CONTROL), 2018.

[25] A. Wächter, L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," Mathematical Programming, vol.106, pp.25-57, 2006.