Available online at www.sciencedirect.com

ScienceDirect

IFAC Papers Online CONFERENCE PAPER ARCHIVE

IFAC-PapersOnLine 48-27 (2015) 141–146

Temporal Logic Inference with Prior Information: An Application to Robot Arm Movements ⋆

Zhe Xu * **Calin Belta** ** **Agung Julius** *

* *Department of Electrical, Computer, and Systems Engineering, Rensselaer Polytechnic Institute, Troy, NY 12180, USA. Email: xuz8,juliua2@rpi.edu.*

** *Department of Mechanical Engineering and the Division of Systems Engineering, Boston University, Boston, MA 02215, USA. Email: cbelta@bu.edu.*

Abstract: Temporal logics are widely used to express (desired) system properties in controller synthesis and verification. In linear temporal logics, the semantics of the formulae are defined on the execution trajectories of the system. Recently, there have been a lot of interest in using dense-time linear temporal logic, such as Signal Temporal Logic (STL) in characterizing system trajectories. In this paper, we present a new method to derive an STL formula that characterizes the motion of a robot arm. Our work generalizes earlier work in this area by (i) allowing the use of polyhedral predicates, and (ii) incorporating a priori knowledge about the predicates. The formula is defined by a set of parameters, whose values are determined by minimizing a cost function that balances the trade-off between the formula's match with the trajectories and the the similarity between its predicate and an a priori known predicate. We apply our algorithm on experimental trajectories generated using a PHANToM Omni robot.

© 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.

Keywords: Signal Temporal Logic, Learning, Optimization

1. INTRODUCTION

Formal methods have been used in robotics because they provide verification and synthesis tools for robotics control systems (see Fainekos et al. (2005); Finucane et al. (2010); Fainekos et al. (2007)). Temporal logic is an effective tool to express complex high-level control specifications, for which formal controller synthesis and verification can be performed. Unlike traditional robot motion with a simple specification such as "*go from A to B while avoiding obstacles*", temporal logic can express more complex specifications and precise temporal properties that often occur in reality such as *"Go to A and B and then C in different time intervals. Don't go to D and E in certain time intervals."* There has been rich literature on how to design controllers to meet temporal logic specifications (see Kress-Gazit et al. (2009); Raman et al. (2012); Lin et al. (2002); Smith et al. (2010); Ulusoy et al. (2011)). The temporal logic formulae are typically predefined as a guideline for the behaviors of the system.

Recently, there has been a growing interest in identifying dense-time temporal logic formulae from system trajectories. Algorithms for this purpose are enabled by the robust semantics of dense-time temporal logics. For example, Fainekos and Pappas introduced a robust semantics for Metric Temporal Logic (MTL) based on how far a given trajectory stands, in space and time, from satisfying or violating a temporal logic property (see Fainekos and Pappas (2009)). With Signal Temporal Logic (STL), Donze and Maler also defined several variants of robustness measures (see Donzé and Maler (2010), and more details in Sec. 2). They also presented a method to compute these robustness measures as well as their sensitivity to the parameters of the system or parameters appearing in the formula. STL inference can be performed roughly in two stages (see the work in Kong et al. (2014)). In the first stage, the structure of the formula is chosen. The structure of the formula refers to how the atomic propositions are combined using temporal and logical operators. Kong et al. showed that the formulae admit a partial order of complexity, based on which the structure can be chosen. Once the structure is chosen, the formula is characterized by some parameters that modify the atomic predicates and the temporal operators. In the second stage, these parameters are determined using non-convex optimization. In the context of specification mining for control systems, Jin et al. (2013) also solved the problem of identifying parameters for STL formulae from simulation traces.

In this paper, we present a new algorithm for STL formula inference that takes into account a priori information about the atomic predicates. The motivation for our work is to enable robots to learn from human demonstrations and generate temporal logic specifications from the demonstration data. We seek to infer STL formulae that classify different robot arm movements, predict sequential robot

⋆ Corresponding Author: Zhe Xu. We acknowledge the support of the National Science Foundation through grants number CNS-0953976, CNS-1218109, and NRI-1426907, and the Office of Naval Research through grant number N00014-14-1-0554 for the research reported in this paper.

2405-8963 © 2015, IFAC (International Federation of Automatic Control) Hosting by Elsevier Ltd. All rights reserved.
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2015.11.166

arm movements and achieve multi-goals while avoiding different obstacles during different time intervals. For these purposes, it is natural to assume that a priori information about (some of) the atomic predicates involved in the formulae is available. For example, we can naturally assign a subset of the state-space to the predicate "*the arm is stretched upright*". By including such prior information in the inference process, we can ensure that the STL formula are composed of atomic predicates with a priori assigned meaning.

The rest of this paper is structured as follows. Section 2 reviews the framework of Signal Temporal Logic and the robustness degree. Section 3 describes the types of temporal logic inference tasks that we consider in this paper. Section 4 outlines the algorithms that we propose to complete the tasks mentioned above. Section 5 describes the implementation of the algorithms on trajectories generated using the PHANToM Omni haptic device. Finally, some conclusions are presented in Section 6.

2. SIGNAL TEMPORAL LOGIC (STL)

In this section, we briefly review the Signal Temporal Logic (STL) (see Donzé and Maler (2010)). We start with the Boolean semantics of STL. The state of the system we are studying is described by a set of n variables that can be written as a vector $x = [x_1, x_2, \ldots, x_n]^T$. The domain of x is denoted by $\mathbb{X} = \mathbb{X}_1 \times \mathbb{X}_2 \times \cdots \times \mathbb{X}_n$. The domain $\mathbb{B} = \{\text{True}, \text{False}\}$ is the Boolean domain and the time set is $\mathbb{T} = R_{\geqslant 0}$. With a slight abuse of notation, we define trajectory (or signal or behavior) x describing an evolution of the system as a function from $\mathbb{T}$ to $\mathbb{X}$. Therefore, x_i refers to both the name of the i-th state variable and its valuation in $\mathbb{X}$. A set $\Pi = \{\pi_1, \pi_2, \ldots \pi_n\}$ is a set of atomic propositions, each of which can be either true or false. The atomic propositions can express properties such as "*the robot is inside the safe area 1*", or "*the robot arm is lifting*". The syntax of STL is defined recursively as follows:

$$\phi := \top \mid \pi \mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \mathcal{U}_I \phi_2$$

where $\top$ stands for the Boolean constant True, π is an atomic proposition, $\neg$ (negation), $\wedge$(conjunction), $\vee$ (disjunction) are standard Boolean connectives, $\mathcal{U}$ is a temporal operator representing "until", I is an interval of the form $I = (i_1, i_2), (i_1, i_2], [i_1, i_2)$ or $[i_1, i_2] (i_1, i_2 \in \mathbb{T})$. We can also derive two useful temporal operators from "until"$(\mathcal{U})$, which are "eventually" $\Diamond\phi = \top\mathcal{U}\phi$ and "always" $\Box\phi = \neg\Diamond\neg\phi$.

We use (x, t) to represent the trajectory x at time t; $(x, t) \models \pi$ means the trajectory x satisfies π at time t, $\pi \in \Pi = \{\pi_1, \pi_2, \ldots \pi_n\}$, or equivalently, if $p \subset \mathbb{X}$ is the predicate of the proposition π, i.e. the set of states that satisfy the proposition π, $x(t) \in p$.

The rest of the Boolean semantics of STL are defined recursively as follows:

$$\begin{aligned}
(x,t) \models \neg\phi \quad &\text{iff} \quad (x,t) \not\models \phi \\
(x,t) \models \phi_1 \wedge \phi_2 \quad &\text{iff} \quad (x,t) \models \phi_1 \quad \textit{and} \quad (x,t) \models \phi_2 \\
(x,t) \models \phi_1 \vee \phi_2 \quad &\text{iff} \quad (x,t) \models \phi_1 \quad \textit{or} \quad (x,t) \models \phi_2 \\
(x,t) \models \phi_1 \mathcal{U}_{[a,b)} \phi_2 \quad &\text{iff} \quad \exists t' \in [t+a, t+b) \\
&\qquad s.t. (x,t') \models \phi_2, (x,t'') \models \phi_1 \\
&\qquad \forall t'' \in [t+a, t')
\end{aligned}$$

The robustness degree of a trajectory x with respect to an STL formula ϕ at time t is given as $r(x, \phi, t)$, where r can be calculated recursively via the quantitative semantics (see Donzé and Maler (2010)). It should be noted that in computing the robustness degree with a computer, the data is actually time-stamped. So only the time-stamped points are used for calculation.

$$\begin{aligned}
r(x, x_i \geq b, t) &= x_i(t) - b, \\
r(x, x_i < b, t) &= b - x_i(t), \\
r(x, \neg\phi, t) &= -(r(x, \phi, t)), \\
r(x, \phi_1 \wedge \phi_2, t) &= \min(r(x, \phi_1, t), r(x, \phi_2, t)), \\
r(x, \phi_1 \vee \phi_2, t) &= \max(r(x, \phi_1, t), r(x, \phi_2, t)), \\
r(x, \Box_{[\tau_1, \tau_2)}\phi, t) &= \min_{t+\tau_1 \leq t' < t+\tau_2} r(x, \phi, t'), \\
r(x, \Diamond_{[\tau_1, \tau_2)}\phi, t) &= \max_{t+\tau_1 \leq t' < t+\tau_2} r(x, \phi, t').
\end{aligned}$$

The robustness degree of the entire trajectory ω is denoted as

$$\begin{aligned}
r(x, \Box_{[\tau_1, \tau_2)}\phi) &= \min_{\tau_1 \leq t < \tau_2} r(x, \phi, t) \\
r(x, \Diamond_{[\tau_1, \tau_2)}\phi) &= \max_{\tau_1 \leq t < \tau_2} r(x, \phi, t)
\end{aligned}$$

In this paper, we consider atomic predicates that are polyhedra, i.e. not necessarily rectangular sets as in Jin et al. (2013); Kong et al. (2014). For this purpose, we extend the quantitative semantics of STL by defining

$$r(x, a^T x > b, t) = a^T x - b, \ a \in \mathbb{R}^n, \ b \in \mathbb{R}. \tag{1}$$

The set of states for each predicate p that we consider in this paper is expressed as a conjunction of linear inequalities

$$p : \left(x \in \mathbb{X} \mid \left(\bigwedge_{k=1}^{m} a_k^T x > b_k \right) \right), a_k \in \mathbb{R}^n, b_k \in \mathbb{R}, \tag{2}$$

where the vector a_k and number b_k are parameters that define the predicate, and m is the number of linear inequalities in the predicate. As, for example, the formulae $(x_1 + 2x_2 > 4)$ and $(2x_1 + 4x_2 > 8)$ are essentially the same, in order to reduce redundancy and expedite the searching process, we constraint $\|a_k\|_2 = 1$.

3. PROBLEM DESCRIPTION

In this paper, we intend to infer STL formulae to achieve the following three kinds of tasks. The first two tasks are also considered in the paper (Kong et al. (2014)):

Task 1: Classification of different movements. Given two sets of trajectories $\mathcal{A}$ and $\mathcal{B}$, we seek to find an STL formula ϕ of the form $\Box_{[\tau_1, \tau_2]}\pi$ or $\Diamond_{[\tau_3, \tau_4]}\pi$ that is satisfied by the trajectories in $\mathcal{A}$ and violated by the trajectories in $\mathcal{B}$.

Task 2: Identification of sequential movements The formula $\Box_{[\tau_1,\tau_2]}\pi_1 \Rightarrow \Diamond_{[\tau_3,\tau_4]}\pi_2$ reads "*If* π_1 *is always true in the time interval* $[\tau_1,\tau_2]$ *then eventually* π_2 *is true in the time interval* $[\tau_3,\tau_4]$". In the context of robotics, such formula can be naturally interpreted as a prescription for a sequential movement. In this task, we seek to identify formulae with this structure from the trajectories. As $\Box_{[\tau_1,\tau_2]}\pi_1 \Rightarrow \Diamond_{[\tau_3,\tau_4]}\pi_2$ is equivalent to $\neg\left(\Box_{[\tau_1,\tau_2]}\pi_1\right) \vee \Diamond_{[\tau_3,\tau_4]}\pi_2$, by searching for the formula with this structure, the implication relationship can be obtained.

Task 3: Identification of goals and obstacles In the context of robotics, the formula $\Box_{[\tau_1,\tau_2]}\neg\pi_1 \wedge \Diamond_{[\tau_3,\tau_4]}\pi_2 \wedge \Diamond_{[\tau_5,\tau_6]}\pi_3$ expresses the property that the robot will eventually reach π_2 during the time interval of $[\tau_3,\tau_4]$, and eventually reach π_3 during the time interval of $[\tau_5,\tau_6]$, and it must avoid π_1 during the time interval of $[\tau_1,\tau_2]$. Thus, we can naturally interpret π_1 as an obstacle, and $\pi_{2,3}$ as (sub)goals of the motion. In this task, we seek to identify formulae of this structure. That is, the goal areas or obstacles can be characterized by formulae of the form $\Box_{[\tau_1,\tau_2]}\neg\pi_1$ and $\Diamond_{[\tau_3,\tau_4]}\pi_2$, respectively.

In addition to the description of the tasks above, we also have alternative version in which the computed predicate is required to be close to an a priori defined predicate. The notion of distance between two predicates are expressed as the Hausdorff distance between them, as explained in the next section.

Note that Task 1 is fundamentally different from Tasks 2 and 3. Task 1 is essentially a supervised learning problem for classification, where the identified STL formula ϕ is required to separate a priori known sets $\mathcal{A}$ and $\mathcal{B}$. In Tasks 2 and 3, the objective is to identify a formula ϕ that matches a set of trajectories $\mathcal{A}$ optimally. Unlike in Task 1, there is not a second set of trajectories that needs to be separated from $\mathcal{A}$.

4. SOLUTION

The core of our proposed method is a non-convex optimization problem for finding the best parameters that describe the formula. Denote all parameters that define the formula as α. Take the case of $\Box_{[\tau_1,\tau_2]}(\bigwedge_{k=1}^{m} a_k^T x > b_k)$ for example. As $\|a_k\|_2 = 1$, $a_k^T x$ can be represented as $\cos(\theta_{k,1})x_1 + \sin(\theta_{k,1})\cos(\theta_{k,2})x_2 + \sin(\theta_{k,1})\sin(\theta_{k,2})x_3 + \ldots$. Thus, one simple way to represent a_k is to use the trigonometric parameters $\theta_{k,1}, \theta_{k,2}, \ldots$. For each atomic proposition, $\tau_1, \tau_2, \theta_{k,1}, \theta_{k,2}, \ldots$ will be the elements of α. The lower bound and upper bound of the angles are set to be $[-\pi, \pi]$.

4.1 Classification Problem

To complete Task 1, we essentially follow the approach given in Jin et al. (2013); Kong et al. (2014). Mathematically, Task 1 can be expressed as follows. Suppose that a formula with a given structure is parameterized by α as discussed above. We denote the formula as $\phi(\alpha)$. Given two sets of trajectories $\mathcal{A} = \{\omega_1, \cdots, \omega_{N_A}\}$ and $\mathcal{B} = \{\omega_{N_A+1}, \cdots, \omega_N\}$, find α that minimizes the following cost function:

$$J_1(\alpha) = \sum_{i=1}^{N_A} g(r(\omega_i, \phi(\alpha))) + \sum_{i=N_A+1}^{N} g(-r(\omega_i, \phi(\alpha))), \quad (3)$$

where $r(\omega_i, \phi(\alpha)$ is the robustness degree of the formula $\phi(\alpha)$ for the i-th trajectory. The function $g(\cdot)$ is a penalty function defined as

$$g(x) \triangleq \begin{cases} \epsilon - x & \text{if } x \leq \epsilon\ , \\ 0 & \text{if } x > \epsilon. \end{cases} \quad (4)$$

The parameter ϵ is an appropriate positive number, which is chosen so as to avoid overly specific formula.

4.2 Identification Problem

Mathematically, the identification problem in Tasks 2 and 3 can be expressed as follows. Given a set of trajectories $\mathcal{A} = \{\omega_1, \cdots, \omega_{N_A}\}$, find α that minimizes the following cost function:

$$J_2(\alpha) = \sum_{i=1}^{N_A} f(r(\omega_i, \phi(\alpha))). \quad (5)$$

$$f(x) \triangleq \begin{cases} \rho & \text{if } x \leq \epsilon\ , \\ x^2 & \text{if } x > \epsilon. \end{cases} \quad (6)$$

The parameter ϵ is an appropriate positive number which is chosen so as to avoid overly specific formula and the parameter ρ is a large positive number to penalize negative robustness degrees. Intuitively, $f(\cdot)$ is designed to penalize cases where the robustness degree is (i) negative (corresponding to misclassification), (ii) positive but less than ε (corresponding to overly specific formula that can only represent the given trajectories without much robustness to disturbances, or (iii) positive and very large (corresponding to overly general formula that may overlook essential features of the trajectories). As explained in the previous section, the cost function essentially penalizes misclassification, overfitting, and underfitting. The identified formula $\phi(\alpha^*)$, where α^* minimizes J_2, is essentially the formula that describes the trajectories in $\mathcal{A}$. If $\phi(\alpha)$ is designed to have certain structures, as explain in Sec. 3, then we can complete Tasks 2 and 3, accordingly.

4.3 Incorporating a priori Predicates

Suppose that the predicate in $\phi(\alpha)$ is denoted by the set $p(\alpha) \subset \mathbb{X}$, and we are given a priori predicates $X_1, \cdots, X_p \subset \mathbb{X}$. We want to associate $p(\alpha)$ with one of the predicates in $X_{1,\cdots,p}$. To do so, we define a cost function

$$J_3(\alpha, k) = \sum_{i=1}^{N_A} f(r(\omega_i, \phi(\alpha))) + \lambda\, d_{\mathrm{H}}(X_k, p(\alpha)), \quad (7)$$

where $d_{\mathrm{H}}(X_k, p(\alpha))$ is the Hausdorff distance between the a priori predicate region X_k and $p(\alpha)$, and λ is a positive weighting factor. Larger value of λ means more emphasis in keeping $p(\alpha)$ close to X_k. If (α^*, k^*) minimizes J_3, then $\phi(\alpha^*)$ is the formula that best describes the trajectories in $\mathcal{A}$, and the predicate $p(\alpha^*)$ in it is associated with the predicate X_{k^*}.

The Hausdorff distance is an important tool to measure the similarity between two sets of points (see Atallah (1983)). It is defined as the largest distance from any point in one of the sets, to the closest point in the other set. Let X

and Y be two non-empty subsets of a metric space $(\mathbb{X}, d)$. Their Hausdorff distance $d_{\mathrm{H}}(X,Y)$ is defined as follows

$$d_{\mathrm{H}}(X,Y) = \max\{\sup_{x\in X}\inf_{y\in Y} d(x,y), \sup_{y\in Y}\inf_{x\in X} d(x,y)\}.$$

The expression $\sup_{x\in X}\inf_{y\in Y} d(x,y)$ when both X and Y are polyhedra can be performed as follows:

Step 1: Calculate all vertices of the polyhedron X. Denote them as $\psi_1, \psi_2, \cdots, \psi_{N_X}$.

Step 2: Calculate the distance between ψ_i to Y for each $i \in \{1, \cdots, N_X\}$. This is a convex quadratic optimization problem.

Step 3: Find the maximum of the distances calculated in Step 2.

We use Particle Swarm Optimization (PSO) (see Zhang et al. (2013)) to optimize the cost functions defined above. In each iteration, the parameter α is updated as a swarm of particles move in the parameter space to find the global minimum. Upon convergence of the PSO algorithm, the optimal α is set to be the steady-state global best.

5. IMPLEMENTATION

We apply the proposed method on trajectories generated from a PHANToM Omni haptic device. PHANToM Omni devices have been used in many applications, such as entertainment, teleoperation and medical applications of diagnostic and rehabilitation (see Sansanayuth et al. (2012); Silva et al. (2009)). We use three degrees of freedom of the robot arm, and record the 3D position of the end effector.

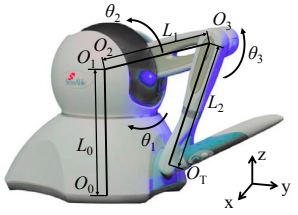

Fig. 1. PHANToM Omni haptic device

The kinematics chain of the PHANToM Omni haptic devices and its representation of variables and constants is shown in Figure 1. The zero configuration is set as O_2O_3 perpendicular to O_0O_1 and O_3O_T perpendicular to O_2O_3. Using the forward kinematics model, the position of the end effector O_T can be obtained as

$$\begin{aligned} x &= -L_1\sin(\theta_1)\cos(\theta_2) - L_2\sin(\theta_1)\sin(\theta_2+\theta_3);\\ y &= L_1\cos(\theta_1)\cos(\theta_2) + L_2\cos(\theta_1)\sin(\theta_2+\theta_3);\\ z &= L_0 + L_1\sin(\theta_2) - L_2\cos(\theta_2+\theta_3); \end{aligned}$$

where $L_0 = 133.35mm$, $L_1 = 133.35mm$, $L_2 = 133.35mm$. The variables $\theta_{1,2,3}$ are the joint angles. As we move the robot arm in a certain manner, the time-stamped location and velocity data of the trajectories can be recorded.

By discretizing the continuous-time location values, the velocities of the end-effector can be obtained as follows:

$$\begin{bmatrix} v_x \\ v_y \\ v_z \end{bmatrix}(k) = \frac{1}{\Delta T}\left(\begin{bmatrix} x \\ y \\ z \end{bmatrix}(k+1) - \begin{bmatrix} x \\ y \\ z \end{bmatrix}(k)\right). \quad (8)$$

where $[x\ y\ z]^T(k)$ and $[v_x\ v_y\ v_z]^T(k)$ are the location vector and velocity vector at time instant k, respectively, and ΔT is the sampling period.

Example 1: Classification Problem In the first example, we intend to classify two kinds of robot arm movements. In its zero configuration, the robot arm is described as folded. We steer the robot arm manually to follow two kinds of trajectories with some small variations each time. We demonstrate first by lifting the folded arm(as the desired movement) 10 times and then lifting while extending the arm (as the undesired movement) 10 times [1]. In this manner, we generate 10 trajectories for both the desired movement and the undesired movement. We search for a classifying STL formula of the form $\Box_{[\tau_1,\tau_2]}\pi$ or $\Diamond_{[\tau_3,\tau_4]}\pi$, where π corresponds to a polyhedral predicate in $[x\ y\ z]^T$ or $[v_x\ v_y\ v_z]^T$as discussed in the previous section. The best two formulae that we found are as follows:

$$\Box_{[0.01,4)} 0.90254 * y - 0.43061 * z > -17.6699$$
$$\Box_{[1.56,4)} z < 255.2673$$

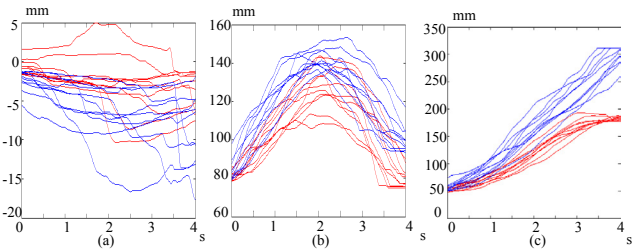

Fig. 2. Trajectories in the first example(red dots for the desired movement, blue dots for the undesired movement): (a)x coordinate values; (b)y coordinate values;(c)z coordinate values

The x, y and z coordinate values of the two kinds of movements are depicted in Figure 2. It can be seen that indeed the y and z coordinate values are the ones that can best differentiate the two kinds of movements.
Similarly, we can classify the two kinds of robot arm movements from the velocity perspective and obtain the following formulae:

$$\Box_{[0.76,2.12]} v_z < 102.3272$$
$$\Diamond_{[0.01,2.91]} 0.76756 * v_x + 0.64097 * v_z < 9.1841$$

Thus, the velocity classification provides an alternative way to differentiate between the two kinds of movements.

Example 2: Identification Problem In the second example, in order to discover sequential movements, we generate 10 trajectories by steering the robot arm to first stretch, and then yaw, and then lift with some small variations each time.

We search for an STL formula with the structure of $\neg\left(\Box_{[\tau_1,\tau_2]}\pi_1\right) \vee \Diamond_{[\tau_3,\tau_4]}\pi_2$, where $\pi_{1,2}$ are polyhedral predicates in $[x\ y\ z]^T$. The following formula is obtained,

$$\Box_{[4,5]} 0.3329 * x + 0.94296 * z < 199.9623 \Rightarrow$$
$$\Diamond_{[5,10.66]} 0.2758 * x - 0.4516 * y + 0.8485 * z > 127.393$$

[1] Demostration videos of Example 1, 2 and 3 can be found at `https://www.youtube.com/watch?v=g8U3ZjwhvPs&feature=youtu.be`

Example 3: Identification Problem with A Priori Predicates In the third example,the robot arm is intentionally steered to reach different a priori regions and avoids certain regions that are set as obstacles. The goals and obstacles predicates, as well as the temporal properties are to be identified. However, we also want to be able to match these predicates with a priori predicates, which are depicted in Figure 3. We call these a priori predicates Regions 1 to 6. To make the formula more specific in different separate time frames, we divide the time frame into three time intervals, the search process is conducted in these three time intervals consecutively. To obtain predicates that have the shape of bounded polyhedra, at least 4 linear inequalities are needed for each of them, so we set $m = 4$ in (2). Besides, outer boundaries are given to guarantee that the sets associated to the predicates are bounded.

We run the experiment twice with different values of λ in the cost function J_3. In the first experiment, $\lambda = 1$. In the second one, $\lambda = 100$. Essentially, we expect to see that the higher λ results in predicates that are close to the a priori ones. The search process takes about 700 seconds on a laptop computer. For $\lambda = 1$, we obtain the following formula:

$$\Box_{[0.01,4]}\neg((0.9992*y+0.040015*z > 266.8318)\wedge(0.42688*x + 0.88941*y + 0.16349*z < 276.8318)\wedge(0.98106*x + (-0.10825)*y+(-0.16062)*z > -43.6728)\wedge(0.57956*x+(-0.69056)*y+0.43271*z < 3.7659))\wedge\Box_{[4.05,7]}\neg((0.9921*x + 0.11232*y + (-0.055925)*z > -222.63)\wedge(0.99817*x+0.023393*y+0.05585*z < -212.63)\wedge((-0.33752)*y+(-0.94132)*z > -221.5122)\wedge(0.86946*x + (-0.48563)*y+(-0.090522)*z < 32.3959))\wedge\Box_{[9.76,10.91]}\neg((0.048447*x + 0.84246*y + 0.53658*z > 234.7687)\wedge((-0.36106)*y + (-0.93254)*z < 276.8318)\wedge(z > 266.8318)\wedge((-1)*z < 276.8318))\wedge\Diamond_{[0.01,3.98]}((0.031996*y + 0.99949*z > -222.63)\wedge(0.079656*x + (-0.81569)*y + 0.57297*z < 276.8318)\wedge(0.82275*x+(-0.5684)*z > -28.0976)\wedge(0.67792*x+(-0.73514)*z < 15.8572))\wedge\Diamond_{[4,7]}((0.86698*x + 0.059055*y + 0.49483*z > -222.63)\wedge(0.90751*x + 0.25487*y + 0.33388*z < 7.631)\wedge(0.96411*y + 0.2655*z > -18.9939)\wedge(0.5444*x + 0.83883*y < -5.9479))\wedge\Diamond_{[7,10.24]}((0.78801*x+0.61546*y+0.015727*z > -73.2865)\wedge((-1)*z < 210.343)\wedge(z > 266.8318)\wedge(0.31673*x + 0.086968*y + 0.94452*z < 276.8318))$$

After matching with the a priori predicates, the matched formula can be expressed as:

$$\Box_{[0.01,4]}(\neg obtained\ region4)\wedge\Box_{[4.05,7]}(\neg obtained\ region5)\wedge\Box_{[9.76,10.91]}(\neg obtained\ region6)\wedge\Diamond_{[0.01,3.98]}(obtained\ region1)\wedge\Diamond_{[4,7]}(obtained\ region2)\wedge\Diamond_{[7,10.24]}(obtained\ region3).$$

The comparisons of the a priori regions and the obtained regions can be seen in Figure 3. The obtained predicates have different shapes and sizes from those of the a priori predicates, therefore they can provide new knowledge and help modify our a priori knowledge about the regions.

For $\lambda = 100$, the obtained formula is as follows:

$$\Box_{[0.01,3.95]}\neg((0.70848*x + (-0.70573)*z > -182.5494)\wedge((-1)*z < -172.5494)\wedge(y > 182.2917)\wedge(0.76453*x + (0.64458)*z < 225.1621))\wedge\Box_{[4,7]}\neg(((-1)*z > -90.4979)\wedge(x < 177.494)\wedge((-1)*y > -144.543)\wedge(x < -131.0609))\wedge\Box_{[8.27,10.08]}\neg((x > -94.6454)\wedge((-1)*z < 43.1152)\wedge(z > 168.9758)\wedge(y < 188.4168))\wedge\Diamond_{[0.01,4]}((y > 100.8097)\wedge(z < 110.8097)\wedge(x > -47.7208)\wedge((-1)*z < 46.576))\wedge\Diamond_{[4,7]}(((-1)*z > -196.5922)\wedge(x < -186.5922)\wedge((-1)*y > -136.9438)\wedge(y < 276.8318))\wedge\Diamond_{[7.22,11]}((z > 234.1874)\wedge(0.86995*y+0.49313*z < 276.8318)\wedge((-1)*y > -222.63)\wedge(x < -112.5924))$$

After matching with the obtained formula, the matched formula is

$$\Box_{[0.01,3.95]}(\neg obtained\ region4)\wedge\Box_{[4,7]}(\neg obtained\ region5)\wedge\Box_{[8.27,10.08]}(\neg obtained\ region6)\wedge\Diamond_{[0.01,4]}(obtained\ region1)\wedge\Diamond_{[4,7]}(obtained\ region2)\wedge\Diamond_{[7.22,11]}(obtained\ region3)$$

It can be seen from Figure 3 that with a larger λ, the obtained predicates in the formula are closer to the a priori ones, as expected.

6. CONCLUSION

In this paper, we presented a method for finding Signal Temporal Logic formulae that describe a set of given system trajectories. Our work generalizes earlier work in this area by (i) allowing the use of polyhedral predicates and (ii) incorporating a priori knowledge about the predicates. We apply the method on trajectories describing the motion of the end effector of a robot arm. We consider three tasks, (i) finding a formula that classifies two sets of trajectories, (ii) identifying a formula that describes sequential motion, and (iii) identifying a formula that describes a Reach-Avoid type of motion. The formula is defined by a set of parameters, whose values are determined by minimizing a non-convex cost function. The cost function is generally based on the robustness degree of the formula and the similarity between the a priori predicates and the obtained predicates in the formula. We apply the method on experimental trajectories generated with a PHANToM Omni robot. We demonstrate that the algorithm can identify STL formulae for each of the tasks above. In robotics, our results are potentially useful in deriving the temporal logic formula that specified a task from human demonstration or human defined trajectories.

REFERENCES

Atallah, M.J. (1983). A linear time algorithm for the hausdorff distance between convex polygons. *Information Processing Letters*, 17(4), 207 – 209. doi:http://dx.doi.org/10.1016/0020-0190(83)90042-X.

Donzé, A. and Maler, O. (2010). Robust satisfaction of temporal logic over real-valued signals. In *Proceedings of the 8th International Conference on Formal Modeling and Analysis of Timed Systems*, FORMATS'10, 92–106. Springer-Verlag, Berlin, Heidelberg.

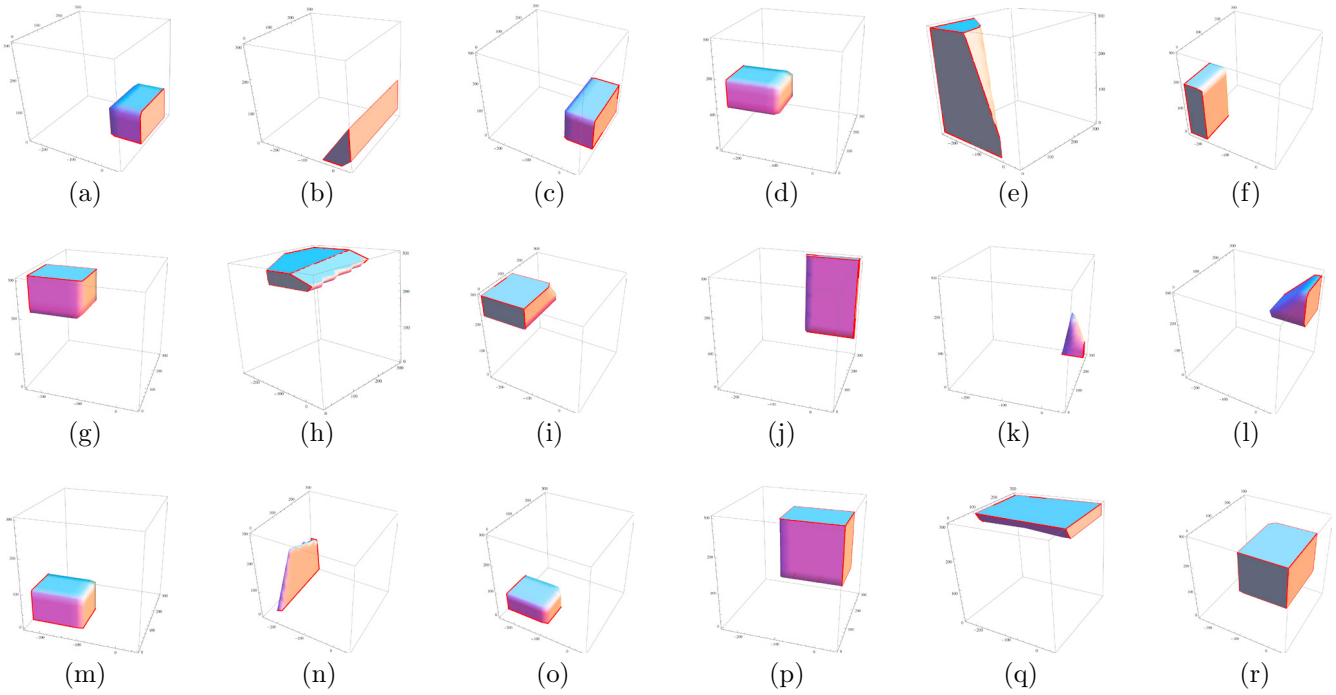

Fig. 3. A priori regions and obtained regions with λ being 1 and 100: (a) a priori region 1; (b) obtained region 1 (λ=1); (c) obtained region 1 (λ=100); (d) a priori region 2; (e) obtained region 2 (λ=1); (f) obtained region 2 (λ=100); (g) a priori region 3; (h) obtained region 3 (λ=1); (i) obtained region 3 (λ=100); (j) a priori region 4; (k) obtained region 4 (λ=1); (l) obtained region 4 (λ=100); (m) a priori region 5; (n) obtained region 5 (λ=1); (o) obtained region 5 (λ=100); (p) a priori region 6; (q) obtained region 6 (λ=1); (r) obtained region 6 (λ=100);

Fainekos, G.E. and Pappas, G.J. (2009). Robustness of temporal logic specifications for continuous-time signals. *Theoretical Computer Science*, 410(42), 4262–4291.

Fainekos, G., Kress-Gazit, H., and Pappas, G. (2005). Hybrid controllers for path planning: A temporal logic approach. In *Decision and Control, 2005 and 2005 European Control Conference. CDC-ECC '05. 44th IEEE Conference on*, 4885–4890. doi:10.1109/CDC.2005.1582935.

Fainekos, G.E., Girard, A., Kress-Gazit, H., and Pappas, G.J. (2007). Temporal logic motion planning for dynamic robots.

Finucane, C., Jing, G., and Kress-Gazit, H. (2010). Ltlmop: Experimenting with language, temporal logic and robot control. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 1988–1993. doi:10.1109/IROS.2010.5650371.

Jin, X., Donze, A., Deshmukh, J.V., and Seshia, S.A. (2013). Mining requirements from closed-loop control models. In *Proc. Int. Conf. Hybrid Systems: Computation and Control*, 43–52.

Kong, Z., Jones, A., Medina Ayala, A., Aydin Gol, E., and Belta, C. (2014). Temporal logic inference for classification and prediction from data. In *Proceedings of the 17th International Conference on Hybrid Systems: Computation and Control*, HSCC '14, 273–282. ACM, New York, NY, USA. doi:10.1145/2562059.2562146.

Kress-Gazit, H., Fainekos, G., and Pappas, G. (2009). Temporal-logic-based reactive mission and motion planning. *Robotics, IEEE Transactions on*, 25(6), 1370–1381. doi:10.1109/TRO.2009.2030225.

Lin, K.H., Lam, K.M., and Siu, W.C. (2002). A new approach using modified hausdorff distances with eigenface for human face recognition. In *Control, Automation, Robotics and Vision, 2002. ICARCV 2002. 7th International Conference on*, volume 2, 980–984 vol.2. doi:10.1109/ICARCV.2002.1238557.

Raman, V., Finucane, C., and Kress-Gazit, H. (2012). Temporal logic robot mission planning for slow and fast actions. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, 251–256. doi:10.1109/IROS.2012.6385935.

Sansanayuth, T., Nilkhamhang, I., and Tungpimolrat, K. (2012). Teleoperation with inverse dynamics control for phantom omni haptic device. In *SICE Annual Conference (SICE), 2012 Proceedings of*, 2121–2126.

Silva, A., Ramirez, O., Vega, V., and Oliver, J. (2009). Phantom omni haptic device: Kinematic and manipulability. In *Electronics, Robotics and Automotive Mechanics Conference, 2009. CERMA '09.*, 193–198. doi:10.1109/CERMA.2009.55.

Smith, S., Tumova, J., Belta, C., and Rus, D. (2010). Optimal path planning under temporal logic constraints. In *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*, 3288–3293. doi:10.1109/IROS.2010.5650896.

Ulusoy, A., Smith, S., Ding, X.C., Belta, C., and Rus, D. (2011). Optimal multi-robot path planning with temporal logic constraints. In *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*, 3087–3092. doi:10.1109/IROS.2011.6094884.

Zhang, F., Cao, J., and Xu, Z. (2013). An improved particle swarm optimization particle filtering algorithm. In *Communications, Circuits and Systems (ICCCAS), 2013 International Conference on*, volume 2, 173–177. doi:10.1109/ICCCAS.2013.6765312.