Learning-Based Model Predictive Control under Signal Temporal Logic Specifications

Kyunghoon Cho and Songhwai Oh

Abstract— **This paper presents a control strategy synthesis method for dynamical systems with differential constraints while satisfying a set of given rules in consideration of their importances. A special attention is given to situations where all rules cannot be met in order to fulfill a given task. Such dilemmas compel us to make a decision on the degree of satisfaction of each rule including which rule should be maintained or not. In this work, we propose a learning-based model predictive control method in order to solve this problem, where a key insight is to combine a learning method and traditional control scheme so that the designed controller behaves close to human experts. A rule is represented as a signal temporal logic (STL) formula. A robustness slackness, a margin to the satisfaction of the rule, is learned from expert's demonstrations using Gaussian process regression. The learned margin is used in a model predictive control procedure, which helps to decide how much to obey each rule, even ignoring specific rules. In track driving simulation, we show that the proposed method generates human-like behavior and efficiently handles dilemmas as human teachers do.**

I. Introduction

Robotics has been widely used in broad areas including both civilian and industrial applications and is gradually appearing our everyday lives. Service robots continuously appear in public places, interacting with people and providing services. Especially, autonomous driving is one of topics of great interest in robotics and has been studied extensively. In many robotic applications, rules exist starting from simple collision avoidance in a navigation problem to complex traffic rules of autonomous driving. These rules are mainly for safety and robots, in most cases, need to satisfy the rules while fulfilling its tasks.

It is important to notice that rules do not have equal importance and there are rules that, depending on the situation, should be prioritized or, in some cases, ignored. For example, in autonomous driving, there may be a situation in which some rules must be disobeyed, such as changing lanes in heavy traffic, deciding what to do at a yellow traffic light or crossing a double yellow line to pass an illegally parked vehicles. Such dilemmas make it difficult for a robot to determine how well a robot will comply with the rules and it is a challenging problem to find control inputs of the robot in consideration of these constraints.

K. Cho and S. Oh are with the Department of Electrical and Computer Engineering and ASRI, Seoul National University, Seoul, Korea (e-mail: kyunghoon.cho@cpslab.snu.ac.kr, songhwai@snu.ac.kr).

This work was supported by Basic Science Research Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (NRF-2017R1A2B2006136), by the Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science and ICT (2017M3C4A7065926), and by the Brain Korea 21 Plus Project in 2017.

Model predictive control (MPC) has proven effective for autonomous control, which is known as online trajectory optimization [1]. The key insight of MPC is attributed to finding the optimal control inputs for a certain cost over inputs and expected future outputs through an accurate predictive model. MPC takes the optimal control framework with an objective term that describes the behavior of the robot and constraint terms to restrict unwanted or unsafe behaviors. MPC has been applied to various works and proven to be effective for many complex tasks, including full body control of humanoid robots [2].

The main difficulty in MPC is the design of controllers. Although practiced humans are able to control a robot well, it is difficult to design MPC to reflect such skillful behavior. For example, in autonomous track driving, expert drivers decides how to drive a car in various situations while considering traffic rules. The driver may have to choose an action between deceleration or shifting to another lane when a slow moving car is in front. A suitable set of MPC parameters should be found to cope with these various situations and a manual or exhaustive search is computationally burdensome.

Recently, imitation learning is emerging for robot learning problems. It seeks near-optimal control from demonstrations of human experts, without manually designing a policy or cost function. Imitation learning has strong strength for modeling a complex policy function with trading multiple desiderata off [3]. It learns weights of each desideratum from expert demonstration so that a robot can mimic the behavior of experts. Although imitation learning has great advantage over traditional approaches, it does not guarantee its performance. In the presence of safety rules such as collision avoidance, it is not easy to ensure that controls generated from imitation learning always satisfy these rules, while following the rules is of paramount importance for the safety of the robot and people nearby.

In this paper, we consider a control synthesis problem under the rules. The existence of priorities among the rules is assumed, and we want to design a controller which takes the priorities into account so that it can handle dilemmas mentioned above. Our approach is founded on the model predictive control scheme, rather than relying entirely on a learning method, which allows us to handle each rule constraint leading to performance guarantee. We model the rules as signal temporal logic (STL) [4], [5]. Temporal logic provides a mathematical formalism to specify desired behaviors of a system, and are utilized to specify robot task specifications [6]–[8]. Signal temporal logic has strength on the specification of properties of dense time, real-valued signals, which is appropriate for real-robotic applications. Instead of directly finding out priorities among rules, we

learn the lower bound of degrees of satisfactions from expert demonstrations. This allows us to figure out which rules to follow, rather than maintaining strict compliance with all rules. Our approach, which is a combination of machine learning and traditional model predictive control, guides a robot to act like human experts.

II. Related Work

Trajectory optimization or model predictive control under temporal logic specifications has been considered before in the context of linear temporal logic (LTL). Mixed-integer linear programming was proposed to generate trajectories for continuous systems with finite-horizon LTL specifications in [9], [10]. Authors in [11] encode a general LTL formula as mixed-integer linear constraints, which includes infinite run with periodic structure. Also, optimal path planning problem under synthetically co-safe LTL specification is considered in [12], which is based on sampling-tree and two-layered structure.

Recently, the MPC scheme has been applied to signal temporal logic (STL) [13], [14]. Authors in [13] frame MPC in terms of control synthesis from STL specifications, where a STL specification is encoded as mixed integer linear programming. The presented encoding method is capable of calculating open loop control signals that satisfy finite and infinite horizon STL properties and, in addition, generate signals that maximize quantitative (robust) satisfaction. In [14], a novel form of STL is used to allow a user to embed predictive models and associated uncertainties. The key concept of the framework is probabilistic predicates that take random variables as parameters, reasoning about safety under uncertainty.

The concept of combining MPC and machine learning approach was introduced in [15]–[17]. These works focus on solving the problem of system identification for MPC, which is different from our approach. Learning based MPC (LBMPC) was introduced in [15], which allows a designer to specify performance targets to optimize and explicitly incorporate online model updates to further improve performance. Authors in [16] try to model disturbances in a system with Gaussian process regression. Deep learning, which is an emerging topic in machine learning, has been applied in MPC in order to learn task-specific controls for complex tasks such as cutting food [17].

There have been attempts to handle dilemmas that we have introduced. Tumova et al. [18] and Castro et al. [19] consider the problem similar to ours. They focus on the dilemma of when all LTL rules cannot be satisfied in a path planning problem, and they try to find a minimally violated path. However, they require prior knowledge of weights among rules, while the proposed approach directly learns from demonstrations by experts. Driving dilemmas in urban environments are mainly considered in [20]. It solves the problem by applying inverse reinforcement learning so that the expert driving strategy can be learned. We did not directly compare the performance between our method and [20], but our approach can directly address the condition of the rule and has benefits when following a specific rule is important.

III. Preliminaries

A. System Model

We consider a continuous time dynamical system

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

where $\mathrm{x}_t \in \mathcal{X} \subset \mathbb{R}^{n_\mathrm{x}}$ is the state, $\mathrm{u}_t \in \mathcal{U} \subset \mathbb{R}^{n_\mathrm{u}}$ is the control input and f is a smooth (continuously differentiable) function of its variables. With a predefined time step dt, the continuous system (1) can be discretized into the following form:

$$\dot{\mathrm{x}}_{n+1} = f(\mathrm{x}_n, \mathrm{u}_n), \tag{2}$$

where n is discrete time step $n = \lfloor t/dt \rfloor$, and x_0 denotes an initial state. For a fixed horizon H, let $\mathbf{x}(\mathrm{x}_n, \mathbf{u}^{H,n})$ be a generated trajectory starting from x_n with control inputs $\mathbf{u}^{H,n} = \mathrm{u}_n, ..., \mathrm{u}_{n+H-1}$.

A *signal* is a sequence of states and controls, which is defined as:

$$\xi(\mathrm{x}_n, \mathbf{u}^{H,n}) = (\mathrm{x}_n, \mathrm{u}_n), \ldots, (\mathrm{x}_{n+H-1}, \mathrm{u}_{n+H-1}). \tag{3}$$

With a slight abuse of notation, $\xi(n)$ is a signal starting from time step n.

B. Signal Temporal Logic

Signal temporal logic (STL) is a logical formalism which allows us to specify the properties of real-value, dense-time signals and has been widely used for the analysis of continuous and hybrid systems [4], [5]. A predicate of a STL formula is defined as an inequality in the form of $\mu(\xi(t)) > 0$, where μ is a function of the signal ξ at time t. The truth value of the predicate μ is equivalent to $\mu(\xi(t)) > 0$. A STL formula is a composition of boolean and temporal operations on these predicates and the syntax of STL formulae φ is defined recursively as follows:

$$\varphi ::= \mu \mid \neg\mu \mid \varphi \wedge \psi \mid \mathrm{G}_{[a,b]}\psi \mid \varphi \mathrm{U}_{[a,b]}\psi,$$

where φ and ψ are STL formulas, G denotes the *globally* operator and U is the *until* operator. The validity of a STL formula φ with respect to signal ξ at time t is defined inductively as follows:

$$\begin{aligned}
(\xi, t) &\vDash \mu &&\Leftrightarrow \quad \mu(\xi(t)) > 0 \\
(\xi, t) &\vDash \neg\mu &&\Leftrightarrow \quad \neg((\xi, t) \vDash \mu) \\
(\xi, t) &\vDash \varphi \wedge \psi &&\Leftrightarrow \quad (\xi, t) \vDash \varphi \wedge (\xi, t) \vDash \psi \\
(\xi, t) &\vDash \varphi \vee \psi &&\Leftrightarrow \quad (\xi, t) \vDash \varphi \vee (\xi, t) \vDash \psi \\
(\xi, t) &\vDash \mathrm{G}_{[a,b]}\varphi &&\Leftrightarrow \quad \forall t' \in [t+a, t+b],\ (\xi, t') \vDash \varphi \\
(\xi, t) &\vDash \varphi \mathrm{U}_{[a,b]}\psi &&\Leftrightarrow \quad \exists t' \in [t+a, t+b] \,\text{s.t.} (\xi, t') \vDash \psi \wedge \\
& && \qquad \forall t'' \in [t, t'],\ (\xi, t'') \vDash \varphi.
\end{aligned}$$

The notation $(\xi, t) \vDash \varphi$ denotes that a signal ξ satisfies the STL formula φ at time t. For example, $(\xi, t) \vDash \mathrm{G}_{[a,b]}\varphi$ means that φ holds for the signal ξ between $t + a$ and $t + b$. For discrete-time systems, STL formulas consider intervals over discrete time values.

One great advantage of STL is that it is provided with a metric called *robustness degree* that measures how well a given signal ξ satisfies a STL formula φ. The robustness degree can be defined as a real-valued function of signal ξ

and t, whose value is calculated recursively with respect to the following *quantitative semantics*:

$$\begin{aligned}
\rho^{\mu}(\xi,t) &= \mu(\xi(t)) \\
\rho^{\neg\mu}(\xi,t) &= -\mu(\xi(t)) \\
\rho^{\varphi\wedge\psi}(\xi,t) &= \min(\rho^{\varphi}(\xi,t),\rho^{\psi}(\xi,t)) \\
\rho^{\varphi\vee\psi}(\xi,t) &= \max(\rho^{\varphi}(\xi,t),\rho^{\psi}(\xi,t)) \\
\rho^{\mathrm{G}_{[a,b]}\varphi}(\xi,t) &= \min_{t'\in[t+a,t+b]} \rho^{\varphi}(\xi,t) \\
\rho^{\varphi\mathrm{U}_{[a,b]}\psi}(\xi,t) &= \max_{t'\in[t+a,t+b]} \big(\min(\rho^{\psi}(\xi,t'), \\
&\quad \min_{t''\in[t,t']} \rho^{\varphi}(\xi,t''))\big).
\end{aligned}$$

In this work, we introduce a notation $(\xi,t) \vDash (\varphi,r)$ representing that the signal ξ satisfies the STL formula φ at time t with *robustness slackness* r, which can be defined as follows:

$$(\xi,t) \vDash (\varphi,r) \quad \equiv \quad \rho^{\varphi}(\xi,t) > r. \tag{4}$$

Eqn (4) states that the signal ξ satisfies φ at least with the minimum robustness degree r. The robustness slackness r acts as a margin to satisfaction of STL formula ρ^{φ}. As r increases, strong constraints for the signal ξ to satisfy φ at time t are given, while relaxed constraints are granted for small r. Especially, when $r<0$, it allows violation of φ.

C. Gaussian Process Regression

A Gaussian process (GP) is a collection of random variables which has a joint Gaussian distribution and is specified by its mean function $m(x)$ and covariance function $k(x,x')$ [21]. A Gaussian process $f(x)$ is expressed as:

$$f(x) \sim GP(m(x), k(x,x')).$$

Suppose that $x \in \mathbb{R}^n$ is an input and $y_i \in \mathbb{R}$ is an output. For a noisy observation set $D = \{(x_i,y_i)|i=1,...,n\}$, we can consider the following observation model:

$$y_i = f(x_i) + w_i,$$

where $w_i \in \mathbb{R}$ is a zero-mean Gaussian noise with variance σ_w^2. Then the covariance of y_i and y_j can be expressed as

$$cov(y_i,y_j) = k(x_i,x_j) + \sigma_w^2 \delta_{ij}, \tag{5}$$

where $\delta_{i,j}$ is the Kronecker delta function which is 1 if $i=j$ and 0 otherwise. $k(x_i,x_j) = \phi(x_i)\cdot\phi(x_j)$ is a covariance function based on some nonlinear mapping function ϕ, where k is known as kernel function. We can represent (5) in a matrix form as follows:

$$cov(y) = K + \sigma_w^2 I,$$

where $\mathbf{y} = [y_1 \dots y_n]^T$ and K is a kernel matrix such that $[K]_{i,j} = k(x_i,x_j)$. The conditional distribution of a new output y_* at a new test input $\mathbf{x}_*$ given D becomes

$$y_*|D,\mathbf{x}_* \sim \mathcal{N}(\bar{y}_*, \mathbb{V}(y_*)), \tag{6}$$

where the mean of y_* is

$$\bar{y}_* = k_*^T (K+\sigma_w^2 I)^{-1}\mathbf{y}, \tag{7}$$

and the covariance of y_* is

$$\mathbb{V}(y_*) = k(\mathbf{x}_*,\mathbf{x}_*) - k_*^T (K+\sigma_w^2 I)^{-1} k_*. \tag{8}$$

$k_* \in \mathbb{R}^n$ is a covariance vector between the new data x_* and existing data, such that $[k_*]_i = k(x_*,x_i)$. As for making a prediction given a training set, the computational cost of GP can be reduced by pre-calculating the inverse of a kernel matrix.

IV. Problem Formulation

We now formally state the main problem in this work. Let $\boldsymbol{\varphi} = [\varphi_1,\dots,\varphi_N]$ refer a given set of STL formulas, and $\bar{\boldsymbol{\varphi}} = \varphi_1 \wedge \dots \wedge \varphi_N$ is a conjunction of the STL formulas. A cost function J is defined over the state and control space, where $J(\mathbf{x},\mathbf{u})$ denotes the cost of trajectory $\mathbf{x}$ and control sequence $\mathbf{u}$. A control synthesis problem under a STL formula for model predictive control is stated as follows [13].

Problem 1: Given a system model of the form (2), initial state x_0, horizon length H, compute control input sequence $\mathrm{u}^{H,t}$ at each time step t, which minimizes $J(\mathbf{x}(\mathrm{x}_t,\mathbf{u}^{H,t}),\mathrm{u}^{H,t})$ satisfying $\bar{\boldsymbol{\varphi}}$, i.e.,

$$\begin{aligned}
\underset{\mathbf{u}^{H,t}}{\text{minimize}} \quad & J(\mathbf{x}(\mathrm{x}_t,\mathbf{u}^{H,t}),\mathbf{u}^{H,t}) \\
\text{subject to} \quad & (\xi(\mathrm{x}_t,\mathbf{u}^{H,t}),t) \vDash \bar{\boldsymbol{\varphi}}.
\end{aligned}$$

The above MPC formulation requires the strict satisfaction of the STL formula. In this paper, our goal is to find a control sequence under STL formulas, with consideration of margins to each rules. Let $\boldsymbol{r} = [r_1,...,r_N]$ be a set of robustness slackness, meaning how well to satisfy the STL formulas. Now the MPC formulation under the STL formulas with robustness slackness is stated as below:

Problem 2: Given a system model of the form (2), initial state x_0, horizon length H, and a set of robustness slackness, compute control input sequence $\mathrm{u}^{H,t}$ at each time step t by solving the following optimization problem:

$$\begin{aligned}
\underset{\mathbf{u}^{H,t}}{\text{minimize}} \quad & J(\mathbf{x}(\mathrm{x}_t,\mathbf{u}^{H,t}),\mathbf{u}^{H,t}) \\
\text{subject to} \quad & (\xi(\mathrm{x}_t,\mathbf{u}^{H,t}),t) \vDash (\varphi_1,r_1) \\
& \qquad\vdots \\
& (\xi(\mathrm{x}_t,\mathbf{u}^{H,t}),t) \vDash (\varphi_N,r_N).
\end{aligned}$$

This modified formulation allows flexible handling of STL constraints. It allows us to solve dilemma situations where all STL formulas can not be fulfilled with a properly given robustness slackness values. In this work, we learn the values of the robustness from demonstrations by experts with the assumption that the experts know what rules must precede and how well one has to satisfy each rule.

V. Proposed Method

The overall procedure is shown in Figure 1. The key concept of the proposed framework is that learning methods and STL constraints complement each other so that the designed model predictive controller becomes closer to human experts. Demonstrations are collected from experts, and margin (or robustness slackness) of each defined rules are learned from

demonstrations. Gaussian process regression is applied to predict the robustness slackness for an unseen situations. Based on the learned robustness slackness, the model predictive control method under the STL formulas creates a guaranteed control sequence with respect to specified rules. Linearized models are applied in order to handle nonlinear differential constraints of dynamical systems, although some errors may occur.

A. Learning Robustness Slackness from Demonstration

Let $\Xi = \{\xi^i\}_i^M$ be M demonstrated signals, where $\xi_n^i = (\mathrm{x}_n^i, \mathrm{u}_n^i)$ with x_n^i and u_n^i being the state and control input, respectively, at time step n. $d^{i,j}$ is the lowest value of robustness degree from the current time step n to the future time step $n+H-1$ for the demonstration ξ^i, which is defined as:

$$d_n^{i,j} = \min_{m \in [n, n+H-1]} \rho^{\varphi_j}(\xi^i, m), \tag{9}$$

where H is the control horizon length. $d^{i,j}$ coincides with robustness slackness for the signal with the length H starting from ξ_n^i, since it represents the minimum allowed lower bound of the robustness degree in the time horizon $[n, n+H-1]$.

We define a feature function which maps a signal into a feature vector $\phi : \mathbb{R}^{n_x+n_u} \rightarrow \mathbb{R}^{n_f}$. Generally, a feature vector is designed to represent the current state of the system. For example, in autonomous driving, distances to other cars or the difference between heading of the vehicle and the direction of the lane can be a feature vector. From demonstrated signals Ξ, let D^j be outputs from (9) for a STL formula φ_j. We define Φ as feature vectors mapped from the demonstrated signal Ξ. For a new input feature ϕ_*, Gaussian process regression is applied to predict the lower bound of robustness degree for horizon H, which is robustness slackness. The conditional distribution of a new output d_* at a test input feature ϕ_*, given Φ, D^j becomes

$$d_* | \Phi, D^j, \phi_* \sim \mathcal{N}(\mu_j(\phi_*), \mathbb{V}_j(\phi_*)),$$

where $\mu_j(\phi_*), \mathbb{V}_j(\phi_*)$ denote the mean and variance of the new output, which can be computed using (7) and (8).

B. Model Predictive Control Synthesis

Prior work [13] shows that MPC optimization with STL constraints can be posed as a mixed integer linear program (MILP). It has shown two different encoding styles: one that focuses on whether a STL formula is satisfied or not and the other, which is called 'robustness-based encoding', considering robustness degree of STL formula. In our problem formulation, each STL formula is handled according to the defined robustness slackness, and the robustness-based encoding method is used for each STL formula. Let C_{φ_j, r_j} be encoded constraints for the STL formula φ_j with a robustness slackness r_j. All encoded constraints are combined as follows:

$$z^{\varphi} = \bigwedge_{j=1}^{N} z^{\varphi_j} \iff z^{\varphi} \leq z^{\varphi_j}, \quad z^{\varphi} \geq 1 - N + \sum_{j}^{N} z^{\varphi_j},$$

where $z^{\varphi}, z^{\varphi_j} \in [0, 1]$ are boolean variables with z^{φ} for satisfaction of the all STL constraints and z^{φ_j} for an individual STL formula φ_j. Note that $z^{\varphi_j} = 1$ only if $\rho^{\varphi_j} - r_j > 0$, otherwise $z^{\varphi_j} = 0$.

The proposed algorithm is shown in Algorithm 1. Inputs to the algorithm are a set of STL formulas $\varphi_1, \ldots, \varphi_N$, the time of interest $\tau = [t_0, t_1]$, discretization timestep dt, a control horizon H, an initial signal state ξ_{init} and demonstrated signals Ξ. First, feature vectors and robustness slackness (the lowest robustness degree for the horizon H) are precomputed from demonstrations (line 1). The closed-loop algorithm, finding the optimal strategy at every time step, is run in the time interval $\tau = [t_0, t_1]$. Nonlinear dynamics are linearized with respect to the current signal state (line 4). The robustness slackness of STL formula φ_j for the input feature $\phi(\xi_{cur})$ is estimated through Gaussian process regression (line 6). μ_j and $\mathbb{V}_j$ are the predictive mean and variance, respectively. We update a robustness slackness r_j of each STL formula from the estimated values (line 7). In most cases, the found mean μ_j becomes r_j. However, there are some cases which require modification so that the estimated value is well incorporated into our control scheme: (i) μ_j is out of range, (ii) variance at the tested input is larger than the threshold value, and (iii) current robustness degree d_j is less than μ_j. In case (i), the range on robustness slackness given by the user prohibits the unexpected behavior of the controller, and a proper value in the range is selected for r_j. For the second case (ii), a high variance represents the test input is far from training data, implying large test error. In such a case, we set r_j as 0. The last case (iii) gives infeasible constraint. In order to relax this constraint, we define r_j as a function of timestep which increases from the current robustness degree d_j to μ_j for the horizon H. Based on the updated robustness slackness r_j, each STL formula φ_j is converted into mixed integer programming constraints C_{φ_j, r_j} by robustness-based encodings method (line 8), where C_{φ_j, r_j} consisting of binary variables and linear predicates. In consideration of all STL constraints, dynamic constraints and past trajectory, optimal control sequence is computed with time horizon H with user-defined cost function (line 12). The above procedure repeatedly continues for the time interval τ.

VI. Experimental Results

We implemented the proposed algorithm in Matlab with Yalmip [22] and Gurobi [23] as its optimization engine. The two scenarios in autonomous driving are evaluated in the experiments. A unicycle model is used to define the dynamics of vehicles in the track. We let the state of the system at time t be $\mathrm{x}_t = [x_t, y_t, \theta_t, v_t]^T$, where x_t, y_t denote the position of the vehicle, θ_t is the heading, and v_t is the linear velocity. Further, the control inputs of the system is $\mathrm{u}_t = [w_t, a_t]^T$, where w_t is the angular velocity, and a_t is the acceleration. The dynamic of the vehicle is stated as below:

$$\dot{x}_t = v_t cos(\theta_t),\ \dot{y}_t = v_t sin(\theta_t),\ \dot{\theta}_t = v_t \kappa_1 w_t,\ \dot{v}_t = \kappa_2 a_t,$$

where κ_1, κ_2 are constants. In order to solve the optimization, we linearize the dynamic about the reference point $\hat{\mathrm{x}} =$

Algorithm 1 Learning-Based Controller Synthesis under STL constraints

$\Phi, D^j \leftarrow$ Initialize(Ξ)
$\xi_{cur} \leftarrow \xi_{init},\ \xi_{past} \leftarrow \emptyset$
for $t = t_0 : dt : t_1$ **do**
  $f_{lin} \leftarrow$ Linearize(f, ξ_{cur})
  for $j = 1 : 1 : N$ **do**
    $\mu_j, \mathbb{V}^j \leftarrow$ GP-regression$(\Phi, D^j, \phi(\xi_{cur}))$
    $r_j \leftarrow$ UpdateValue$(\mu_j, \mathbb{V}^j)$
    $C_{\varphi_j, r_j} \leftarrow$ EncodeSTLConstraints(φ_j, r_j)
  end for
  $C_{STL} \leftarrow C_{\varphi_1, r_1} \wedge \ldots \wedge C_{\varphi_N, r_N}$
  $C \leftarrow C_{STL} \wedge f_{lin} \wedge [\xi(t_0, \cdots, t - dt) = \xi_{past}]$
  $\mathbf{u}^{H,t} \leftarrow$ Optimize$(J(\xi^H), C)$
  $\mathrm{x}_{next} = f(\mathrm{x}_{cur}, \mathbf{u}^{H,t}(t))$
  $\xi_{past} \leftarrow [\xi_{past}\ \xi_{cur}]$
  $\xi_{cur} \leftarrow (\mathrm{x}_{next}, \mathbf{u}^{H,t}(t))$
end for

$[\hat{x}, \hat{y}, \hat{\theta}, \hat{v}]^T$. The resulting linear system is a first-order Taylor approximation of the non-linear dynamic, which can be written as follows:

$$\mathrm{x}_{n+1} = A_n \mathrm{x}_n + B_n \mathrm{u}_n + C_n,$$

$$A_n = \begin{bmatrix} 1 & 0 & -\hat{v}sin(\hat{\theta})dt & cos(\hat{\theta})dt \\ 0 & 1 & \hat{v}cos(\hat{\theta})dt & sin(\hat{\theta})dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix},$$

$$B_n = \begin{bmatrix} 0 & 0 \\ 0 & 0 \\ \kappa_1 \hat{v} dt & 0 \\ 0 & \kappa_2 dt \end{bmatrix}, \quad C_n = \begin{bmatrix} \hat{v}sin(\hat{\theta})\hat{\theta}dt \\ -\hat{v}cos(\hat{\theta})\hat{\theta}dt \\ 0 \\ 0 \end{bmatrix}.$$

Throughout the section, we call the vehicle that is in our control as the *ego* vehicle. To learn robustness slackness efficiently, we use different features for each rule. Squared exponential kernel is used in Gaussian process regression.

A. Scenario 1

In the first scenario, the track environment in a two-way two-lane road with a double yellow center-line is considered. There exists an obstacle interrupting the movement of the ego vehicle. Five rules are defined as follows, and each rule is represented as STL formula.

1) Lane keeping: Do not cross the center line.

$$\varphi_1 = (y \geq y_{l,min}) \wedge (y \leq y_{l,max})$$

2) Collision avoidance (Car).

$$\varphi_2 = (x \leq x_{c,min}) \vee (x \geq x_{c,max}) \vee (y \leq y_{c,min}) \vee (y \geq y_{c,max})$$

3) Collision avoidance (Obstacle).

$$\varphi_3 = (x \leq x_{o,min}) \vee (x \geq x_{o,max}) \vee (y \leq y_{o,min}) \vee (y \geq y_{o,max})$$

4) Speed limit: Linear velocity should be less than the the threshold.

$$\varphi_4 = v_t \leq v_{max}$$

5) Track keeping: Do not leave the track.[1]

$$\varphi_5 = (y \geq y_{t,min}) \wedge (y \leq y_{t,max})$$

Figure 2 shows a curved track with two lanes, where an obstacle shown in a stripped box is located in the lane of the ego vehicle (blue). Regression results of robustness slackness are shown in heatmaps for rules φ_1, φ_4, and φ_5, with varying position of another vehicle (red) which is in the opposite direction. Notice that different robustness slackness is obtained with respect to other car's state. A high robustness slackness, which is greater than 0, expresses that the rule condition must be strongly satisfied. A low robustness slackness, which is less than 0, means that the rule condition can be relaxed. In order to advance on, the ego vehicle needs to cross the center line which can be dangerous due to another approaching vehicle. Therefore, it is important to learn when to change the lane, which is related with the rule φ_1.

In Figure 2(a), the second and third columns show high robustness slackness for the states between the ego vehicle and the obstacle (dashed circle), which is greater than 0. It means that the ego vehicle needs to keep its lane rather than changing it, which is reasonable since the the other vehicle is approaching. However, in the first and fourth columns, low robustness slackness values less than 0 are earned, telling us that the ego vehicle can disobey the rule φ_1, which allows the vehicle to change its lane to move forward.

Regarding to the rule φ_4, robustness slackness values are shown in Figure 2(b). Relative high robustness slackness for area in the dashed circle is shown in the second and third columns, while opposite results are presented in the first and fourth columns. This informs that the ego vehicle should decelerate in front of the obstacle until another vehicle passes, and it can speed up when another vehicle passes or is quite far away to reach the ego vehicle. For the rule φ_5, we obtain positive robustness slackness for the majority of the states in the 4 cases, suggesting that the ego vehicle does not leave the track (see Figure 2(c)).

We briefly compare the proposed algorithm with two existing approaches, which are standard MPC under STL constraints [13] and behavioral cloning. As a behavioral cloning method, Gaussian process regression is used for mapping the feature of state to a control input. For the proposed method, the minimum allowable robustness slackness value of φ_3 is set as 0 so that the vehicle can evade from the collision with the obstacle. The results are shown in Figure 3. The trajectory of the ego vehicle (blue) is shown in a blue solid line, while the red line is the trajectory of the other vehicle (red). Figure 3(c) shows a failed case from behavioral cloning, where the ego car collides with the obstacle. In comparison with standard MPC under STL constraints, the proposed method

[1]We assume the track is horizontal and rules related with the track consider y-coordinate of the state. If the track is near vertical, then the STL formulas can be redefined according to x-coordinate of the state.

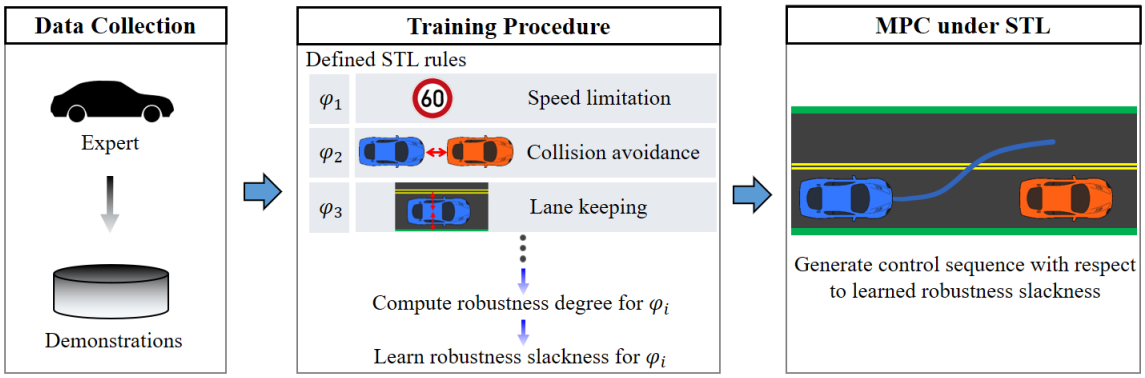

Fig. 1. Overall procedure for the proposed learning-based framework. Demonstrations of experts are acquired and robustness slackness (lower bound for robustness degree) is learned through Gaussian process regression. Based on learned values, model predictive control method computes control sequences in consideration of the STL rules.

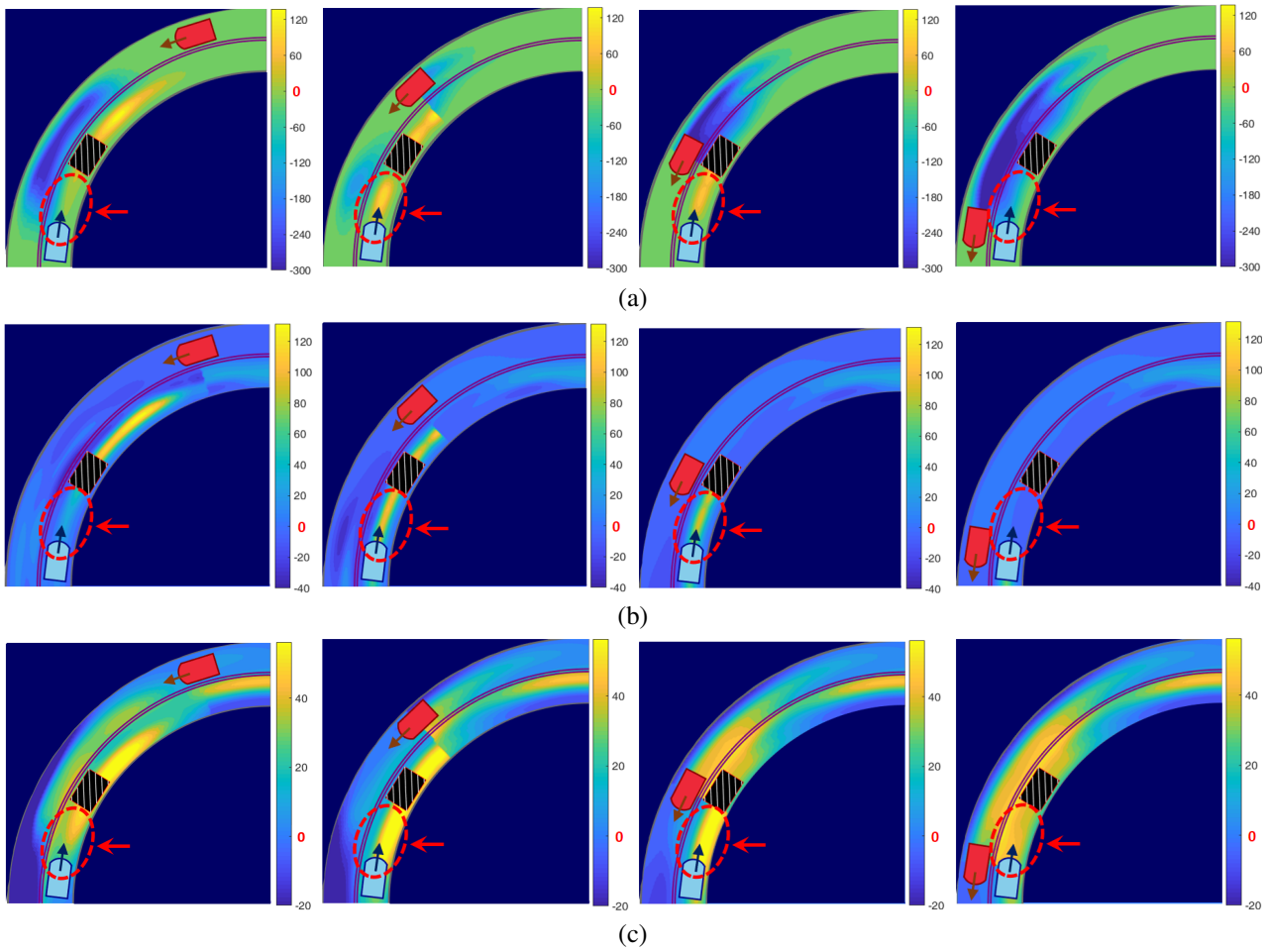

Fig. 2. Learned robustness slackness for (a) φ_1, (b) φ_4 and (c) φ_5 in Scenario 1. The heatmap is the predictive mean of Gaussian process regression according to state of the ego vehicle. The obstacle shown in the black box with the other car marked red

behaves more like human expert due to learned robustness slackness from demonstrations. In case of behavior cloning, it generates control inputs similar to proposed methods in most cases. However, the learning method itself, in this case Gaussian process regression, cannot guarantee the avoidance of obstacles, while the proposed can enforce the rule constraint by managing allowable robustness slackness.

B. Scenario 2

In the second scenario, a single-direction track with multiple lanes is considered. The ego vehicle needs to change its lanes in order to pass the other vehicle in front. When the vehicle attempts to change its lane, it has two options either to move the left lane or the right lane. Five different rules are defined with the corresponding STL formulas.

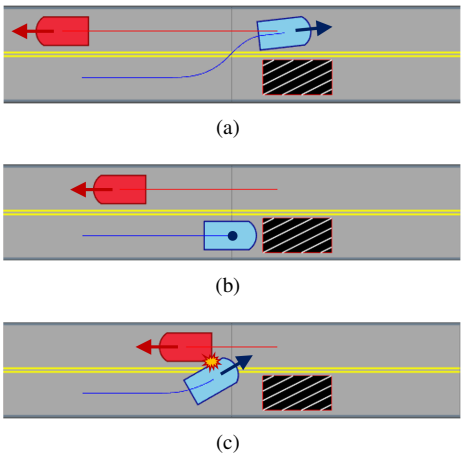

Fig. 3. Generated trajectory of (a) the proposed method, (b) standard MPC under STL constraints, and (c) behavioral cloning. For the behavioral cloning method, the failure case is shown.

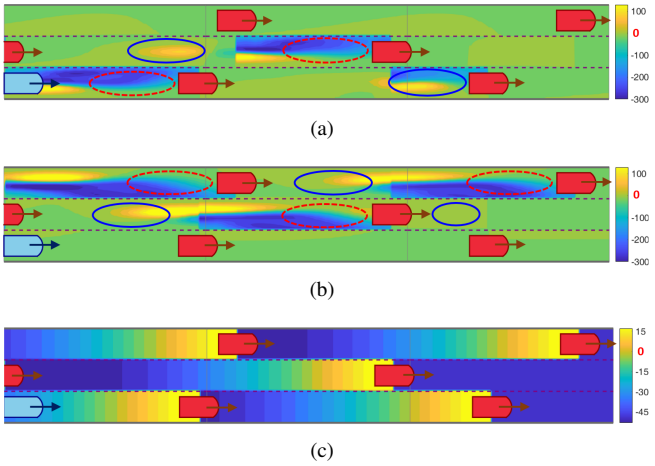

Fig. 4. Estimated mean of robustness slackness from Gaussian process regression for (a) φ_1, (b) φ_2 and (c) φ_5 in Scenario 2. For φ_1 and φ_2, the locations where the rule can be violated are marked as red dashed circles, while robustness degree in the blue solid circles states that the rule must be satisfied.

1) Lane keeping: Do not move to the left lane.

$$\varphi_1 = y_t \leq y_{l,max}$$

2) Lane keeping: Do not move to the right lane.

$$\varphi_2 = y_t \geq y_{l,min}$$

3) Collision avoidance (Car).

$$\varphi_3 = (x_t \leq x_{c,min}) \vee (x_t \geq x_{c,max}) \vee (y_t \leq y_{c,min}) \vee (y_t \geq y_{c,max})$$

4) Speed limit: Linear velocity should be less than the the threshold.

$$\varphi_4 = v_t \leq v_{max}$$

5) Slow down before the other vehicle.

$$\varphi_5 = (v_t \leq v_{th}) \mathrm{U}_{[t_a,t_b]} (x_t \leq x_{c,min})$$

The last rule φ_5 states that the ego vehicle needs to decelerate when it becomes close to another car in the same lane.

Learned robustness slackness is shown in the Figure 4. From the regression results of rules φ_1, φ_2, One interesting thing is that we can find out that the ego vehicle can move to the other lane if it satisfies the following conditions.

(i) There is other vehicle in front, which is close to the ego vehicle.
(ii) There exists an enough space in the lane that the ego vehicle plans to go.
(iii) The ego vehicle does not leave the track.

These conditions seem reasonable in the expert point of view. In Figure 4(a) and (b), the section in the red dashed circles satisfy the above conditions leading to low robustness slackness, while area in the blue solid circles does not satisfy the condition (ii). Notice that robustness slackness in the leftmost lane in Figure 4(a) is almost nonnegative, representing that the ego vehicle does not move to the left lane which does not exist. A similar result is observed in the rightmost lane in Figure 4(b), and these observations support the condition (iii). Also, as the distance between the ego vehicle and the preceding other vehicle decreases, the rule φ_5 requires strong satisfaction (Figure 4(c)).

We also conducted a realistic simulation by using the public Next-Generation Simulation (NGSIM) dataset [24]. The data in [24] contains the vehicle trajectory data on the US Highway 101 with the track information, which covers an area in Los Angeles approximately 640m in length with five mainline lanes and a sixth auxiliary lane for highway entrance and exit. Expert demonstrations are collected again in the environment which has similar scale with the US Highway 101. We put the ego vehicle on the US 101 highway, and run the proposed algorithm.

The resulting controlled output is shown in Figure 5. The ego vehicle (blue) is controlled in consideration of the five STL rules that we define in Scenario 2, and its resulting trajectory is shown in the blue solid line. The snapshot of how the ego vehicle behaves with the robustness slackness of rules defined in Scenario 2 is presented at the specific points P1, P2, P3 and P4. At the point P1, all robustness slackness values are positive, which leads the ego vehicle to keep the current lane with a moderate speed. At P2, the negative robustness slackness of φ_2 allows the ego vehicle to move to the right lane and it is a natural choice since there are other vehicles on the front and left of the ego vehicle. Rules φ_4, φ_5 are disobeyed at the point P3. The reason for this is likely to be the fast approaching vehicle at the rear, resulting negative robustness slackness. At the point P4, the ego vehicle move to the left lane with high speed, which is acceptable because of negative robustness slackness values of φ_1, φ_4, φ_5.

VII. Conclusion

In this paper, we have presented a model predictive control method to control a dynamic system while satisfying a set of STL rules. Rather than strict compliance with all rules, the proposed method follows each rule more efficiently, including disobeying certain rules so that it can solve dilemma situations when all rules can not be satisfied. The proposed

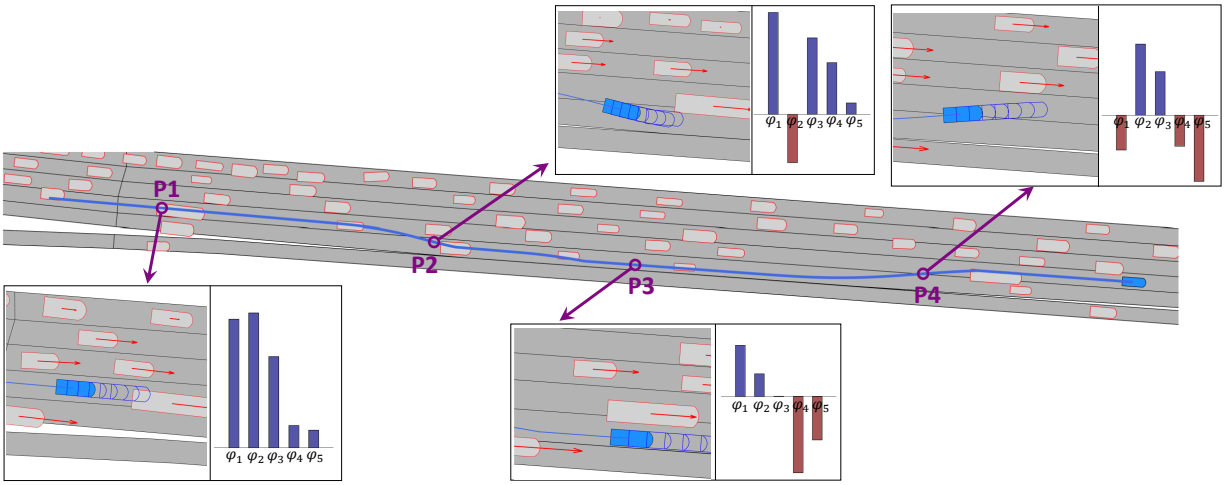

Fig. 5. Simulation result on the US Highway 101. The ego vehicle is marked as blue, and its trajectory is shown in the blue solid line. At specific spots on the trajectory P1, P2, P3, and P4, the snapshot of how the ego vehicle is controlled is shown with the robustness slackness of rules defined in Scenario 2. Positive robustness slackness is marked in blue, while negative robustness slackness in red.

method learns the robustness slackness, which is the lower bound of robustness degree, from demonstrations by experts. Based on learned robustness slackness, the proposed controller controls a robot while valuing the rules differently under different situations, similar to human experts. Our novelty lies in the fact that learning is performed on the satisfaction measure of a rule such that a robot can learn the value systems of humans.

REFERENCES

[1] E. F. Camacho and C. B. Alba, *Model predictive control*. Springer Science & Business Media, 2013.

[2] T. Erez, K. Lowrey, Y. Tassa, V. Kumar, S. Kolev, and E. Todorov, "An integrated system for real-time model predictive control of humanoid robots," in *Humanoid Robots (Humanoids), 2013 13th IEEE-RAS International Conference on*, 2013, pp. 292–299.

[3] P. Abbeel and A. Y. Ng, "Apprenticeship learning via inverse reinforcement learning," in *Proc. of the twenty-first international conference on Machine learning*. ACM, 2004, p. 1.

[4] O. Maler and D. Nickovic, "Monitoring temporal properties of continuous signals," in *FORMATS/FTRTFT*, vol. 3253. Springer, 2004, pp. 152–166.

[5] A. Donzé and O. Maler, "Robust satisfaction of temporal logic over real-valued signals." in *FORMATS*, vol. 6246. Springer, 2010, pp. 92–106.

[6] G. E. Fainekos, H. Kress-Gazit, and G. J. Pappas, "Temporal logic motion planning for mobile robots," in *Proc. of the IEEE International Conference on Robotics and Automation*, Apr. 2005.

[7] S. Karaman and E. Frazzoli, "Complex mission optimization for multiple-uavs using linear temporal logic," in *Proc. of the IEEE American Control Conference*, Jun. 2008.

[8] T. Wongpiromsarn, U. Topcu, and R. M. Murray, "Receding horizon temporal logic planning for dynamical systems," in *Proc. of the IEEE Conference on Decision and Control*, Dec. 2009.

[9] S. Karaman, R. G. Sanfelice, and E. Frazzoli, "Optimal control of mixed logical dynamical systems with linear temporal logic specifications," in *Proc. of the IEEE Conference on Decision and Control*, Dec. 2008, pp. 2117–2122.

[12] K. Cho, J. Suh, C. J. Tomlin, and S. Oh, "Cost-aware path planning under co-safe temporal logic specifications," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2308–2315, 2017.

[10] Y. Kwon and G. Agha, "Ltlc: Linear temporal logic for control," *Hybrid Systems: Computation and Control*, pp. 316–329, 2008.

[11] E. M. Wolff, U. Topcu, and R. M. Murray, "Optimization-based control of nonlinear systems with linear temporal logic specifications," in *Proc. of the IEEE Conference on Robotics and Automation*, 2014, pp. 5319–5325.

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

[14] D. Sadigh and A. Kapoor, "Safe control under uncertainty," *arXiv preprint arXiv:1510.07313*, 2015.

[15] A. Aswani, H. Gonzalez, S. S. Sastry, and C. Tomlin, "Provably safe and robust learning-based model predictive control," *Automatica*, vol. 49, no. 5, pp. 1216–1226, 2013.

[16] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Learning-based nonlinear model predictive control to improve vision-based mobile robot path-tracking in challenging outdoor environments," in *Robotics and Automation (ICRA), 2014 IEEE International Conference on*. IEEE, 2014, pp. 4029–4036.

[17] I. Lenz, R. A. Knepper, and A. Saxena, "Deepmpc: Learning deep latent features for model predictive control." in *Robotics: Science and Systems*, 2015.

[18] Z. Kong, A. Jones, A. Medina Ayala, E. Aydin Gol, and C. Belta, "Temporal logic inference for classification and prediction from data," in *Proceedings of the 17th international conference on Hybrid systems: computation and control*. ACM, 2014, pp. 273–282.

[19] L. I. R. Castro, P. Chaudhari, J. Tumova, S. Karaman, E. Frazzoli, and D. Rus, "Incremental sampling-based algorithm for minimum-violation motion planning," in *Decision and Control (CDC), 2013 IEEE 52nd Annual Conference on*. IEEE, 2013, pp. 3217–3224.

[20] S.-H. Lee and S.-W. Seo, "A learning-based framework for handling dilemmas in urban automated driving," in *Proc. of the IEEE Conference on Robotics and Automation*. IEEE, 2017, pp. 1436–1442.

[21] C. E. Rasmussen and C. K. Williams, *Gaussian processes for machine learning*. MIT press Cambridge, 2006, vol. 1.

[22] J. Lofberg, "Yalmip: A toolbox for modeling and optimization in matlab," in *Computer Aided Control Systems Design, 2004 IEEE International Symposium on*. IEEE, 2004, pp. 284–289.

[23] G. Optimization, "Inc.,gurobi optimizer reference manual, 2014," *URL: http://www. gurobi. com*, 2014.

[24] J. Colyar and J. Halkias, "Us highway 101 dataset," *US Highway 101 Dataset, FHWA-HRT-07-030*, 2007.