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

Learning-Based Distributionally Robust Motion Control with Gaussian Processes

Astghik Hakobyan Insoon Yang

***Abstract*— Safety is a critical issue in learning-based robotic and autonomous systems as learned information about their environments is often unreliable and inaccurate. In this paper, we propose a risk-aware motion control tool that is robust against errors in learned distributional information about obstacles moving with unknown dynamics. The salient feature of our model predictive control (MPC) method is its capability of limiting the risk of unsafety even when the true distribution deviates from the distribution estimated by Gaussian process (GP) regression, within an *ambiguity set*. Unfortunately, the distributionally robust MPC problem with GP is intractable because the worst-case risk constraint involves an infinite-dimensional optimization problem over the ambiguity set. To remove the infinite-dimensionality issue, we develop a systematic reformulation approach exploiting modern distributionally robust optimization techniques. The performance and utility of our method are demonstrated through simulations using a nonlinear car-like vehicle model for autonomous driving.**

I. Introduction

The adoption of learning-based decision-making tools for the intelligent operation of mobile robots and autonomous systems is rapidly growing because of advances in machine learning, sensing, and computing technologies. By learning its uncertain and dynamic environment, a robot can use additional information to improve the control performance. However, the accuracy of inference is often poor, as it is subject to the quality of the observations, statistical models, and learning methods. Employing inaccurately learned information in the robot's decision making may cause catastrophic system behaviors, in particular, leading to collision. The focus of this work is to develop an optimization-based method for safe motion control that is robust against errors in learned information about obstacles moving with unknown dynamics.

Learning-based control methods for mobile robots and autonomous systems can be categorized into two classes. The first class learns unknown system models, while the second class learns unknown environments. Control methods that learn unknown system dynamics typically use model predictive control (MPC) [1]–[5] and model-based reinforcement learning (RL) [6]–[8]. These tools employ various learning or inference techniques to update unknown system model parameters that are, in turn, used to improve control actions or policies. On the other hand, the methods in the second class put more emphasis on "learning the environment" rather than "controlling the robot". In particular, for learning the behavior (or intention) of obstacles or other vehicles, several methods have been proposed that use inverse RL [9]–[11], imitation learning [12], [13], and Gaussian mixture models [14], [15], among others.

Our method is classified as the second since it learns the movement of obstacles. However, departing from the previous approaches, we emphasize the importance of "control" in correcting potential errors in "learning". In our case, the key idea is to determine the motion control action of the agent that is robust against errors in learned information about the obstacles' motion. Specifically, our method uses Gaussian Process (GP) regression [16] to estimate the probability distribution of the obstacles' locations for future stages based on the current and past observations. To actively take into account the possibility that the learned distribution information may be inaccurate, we propose a novel MPC method that optimizes the motion control action subject to constraints on the *risk of unsafety* evaluated under the worst-case distribution in a so-called *ambiguity set*. Thus, the resulting control action will satisfy the risk constraints for safety even when the true distribution deviates from the learned one within the ambiguity set.

Unfortunately, the distributionally robust MPC (DR-MPC) problem is challenging to solve since the worst-case risk constraint involves an infinite-dimensional optimization problem over the ambiguity set of probability distributions. To resolve this issue, we propose a reformulation approach using (i) modern distributionally robust optimization techniques based on Kantorovich duality [17], (ii) the extremal representation of conditional value-at-risk, and (iii) a geometric expression of the distance to the union of half-spaces. The reformulated DR-MPC problem is finite-dimensional and can be efficiently solved by using existing nonlinear programming algorithms. Through simulations using a nonlinear car-like vehicle model for collision-avoidance racing, we empirically show that, unlike the standard non-robust version, our method preserves safety even with moderate errors in the results of GP regression.

The remainder of the paper is organized as follows. In Section II, we present a GP regression approach to learning the motion of obstacles. In Section III, we introduce the learning-based DR-MPC method with a tractable reformulation technique. The simulation results for collision-avoidance racing are presented in Section IV.

This work was supported in part by the Creative-Pioneering Researchers Program through SNU, the Information and Communications Technology Planning and Evaluation (IITP) grant funded by MSIT(2020-0-00857), and Samsung Electronics.

A. Hakobyan and I. Yang are with the Department of Electrical and Computer Engineering, Automation and Systems Research Institute, Seoul National University, Seoul 08826, South Korea, {astghikhakobyan, insoonyang}@snu.ac.kr

II. Learning the Movement of Obstacles Using Gaussian Processes

A. Obstacle Model

We consider a rigid body obstacle interfering with the motion of a mobile robot in $\mathbb{R}^{n_y}$. The obstacle state $\mathrm{x}_o(t) \in \mathbb{R}^{n_\mathrm{x}}$ is defined as the position and orientation of an arbitrary point on the obstacle. The obstacle state evolves with [1]:

$$\mathrm{x}_o(t+1) = \mathrm{x}_o(t) + T_o \mathrm{v}_o(\mathrm{x}_o(t)), \tag{1}$$

where $\mathrm{v}_o(\mathrm{x}_o(t)) \in \mathbb{R}^{n_\mathrm{x}}$ is the vector of the obstacle's velocity, and T_o is the sample time. For ease of exposition, we describe the case of a single obstacle, but our method is valid in multi-obstacle cases as well.

Having the obstacle's states, as well as its geometric parameters, the region occupied by the obstacle at stage t can be modeled (or over-approximated if necessary) as a convex polytope defined by m number of half-spaces:

$$\mathcal{O}(t) := \{\mathbf{x} \in \mathbb{R}^{n_y} \mid G_t \mathbf{x} \leq g_t\}. \tag{2}$$

Here, $G_t \in \mathbb{R}^{m \times n_y}$ and $g_t \in \mathbb{R}^m$ are found from the geometry of the obstacle and the current state by $G_t = \mathrm{G}(\mathrm{x}_o(t))$ and $g_t = \mathrm{g}(\mathrm{x}_o(t))$.

For example, for a car-like obstacle in a 2D environment, the states can be chosen as the Cartesian coordinates and the orientation of an arbitrary point on the obstacle. However, by symmetry, the simplest motion pattern will be obtained for the three candidate states that are shown in Fig. 1. The region occupied by the obstacle is over-approximated as a rectangle, the parameters of which can be found using the geometry of the vehicle (e.g., the length and width of the car) and any of the three candidate states. To find the G_t and g_t, we need to know the exact expression of v_o. However, in practice it is impossible for a robot to have full knowledge of its environment, in particular, the behavior of the obstacle. For predicting the obstacle's motion, we use the GP regression approach introduced in the following subsection.

B. Gaussian Process Regression

GP regression is a nonparametric Bayesian approach to regression and infers a probability distribution over all possible values of a function given some training data [16]. A GP is a collection of random variables, any finite number of which have a joint Gaussian distribution. Because of its simplicity and good empirical performance, GP regression has been a popular tool in several learning-based control methods (e.g., [5], [18], [19]).

In this work, GP regression is used for predicting the noisy velocity function $\mathrm{v}_o(\mathrm{x}_o(t))$ from previous observations of the obstacle's behavior.

We choose the training input data as $\hat{\mathrm{x}} = \{\mathrm{x}_o(t-1), \mathrm{x}_o(t-2), \ldots, \mathrm{x}_o(t-M)\}$, consisting of the obstacle's state for M previous stages. The corresponding measured velocities $\hat{\mathrm{v}}$ are selected as the training output data. In reality, we do not have access to function values; instead, the following noisy observations are available: for the ith observation

$$\hat{\mathrm{v}}^{(i)} = \mathrm{v}_o(\hat{\mathrm{x}}^{(i)}) + \varepsilon, \quad i = 1, \ldots, M,$$

where $\hat{\mathrm{x}}^{(i)} := \mathrm{x}_o(t-i)$, and ε is an i.i.d. zero-mean Gaussian noise with covariance $\Sigma^\varepsilon = \mathrm{diag}([\sigma^2_{\varepsilon,1}\ \sigma^2_{\varepsilon,2}, \ldots, \sigma^2_{\varepsilon,n_\mathrm{x}}])$.

Since the velocities in different dimensions are assumed to be independent, each of them can be learned individually. The dataset for the jth dimension is thus constructed as

$$\mathcal{D}_j = \left\{\left(\hat{\mathrm{x}}^{(i)}, \hat{\mathrm{v}}^{(i)}_j\right),\ i = 1, \ldots, M\right\}.$$

For each dimension of output $\mathrm{v}_o(\cdot)$, we specify a GP prior with mean function $m_j(x)$ and kernel function $k_j(x, x')$. In this paper, we use an RBF kernel that is defined by

$$k_j(x, x') = \sigma^2_{f,j} \exp\Big[-\frac{1}{2}(x - x')^\top L_j^{-1}(x - x')\Big],$$

where L_j is a diagonal length scale matrix and $\sigma^2_{f,j}$ is the signal variance. The prior on the noisy observations is a normal distribution with mean function $m_j(\hat{\mathrm{x}}^{(i)})$ and covariance function $K_j(\hat{\mathrm{x}}, \hat{\mathrm{x}}) + \sigma^2_{\varepsilon,j} I$, where $K_j(\hat{\mathrm{x}}, \hat{\mathrm{x}}) \in \mathbb{R}^{M \times M}$ denotes the covariance matrix of training input data, i.e., $K_j^{(l,k)}(\hat{\mathrm{x}}, \hat{\mathrm{x}}) = k_j(\hat{\mathrm{x}}^{(l)}, \hat{\mathrm{x}}^{(k)})$. Throughout this work, we used a prior distribution with zero-mean.

It follows that the joint distribution of the training output data $\hat{\mathrm{v}}_j$ and the output $\mathbf{v}_j$ at an arbitrary test point $\mathbf{x}$ is given by

$$\begin{bmatrix} \hat{\mathrm{v}}_j \\ \mathbf{v}_j \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} m_j(\hat{\mathrm{x}}) \\ m_j(\mathbf{x}) \end{bmatrix}, \begin{bmatrix} K_j(\hat{\mathrm{x}}, \hat{\mathrm{x}}) + \sigma^2_{\varepsilon,j} I & K_j(\hat{\mathrm{x}}, \mathbf{x}) \\ K_j(\mathbf{x}, \hat{\mathrm{x}}) & k_j(\mathbf{x}, \mathbf{x}) \end{bmatrix} \right),$$

where $K_j^{(l)}(\hat{\mathrm{x}}, \mathbf{x}) = k_j(\hat{\mathrm{x}}^{(l)}, \mathbf{x})$, and $K_j(\mathbf{x}, \hat{\mathrm{x}}) = K_j(\hat{\mathrm{x}}, \mathbf{x})^\top$. As a result, the posterior distribution of the output in the jth dimension at an arbitrary test point $\mathbf{x}$ conditioned on the observed data is Gaussian, with the following mean and

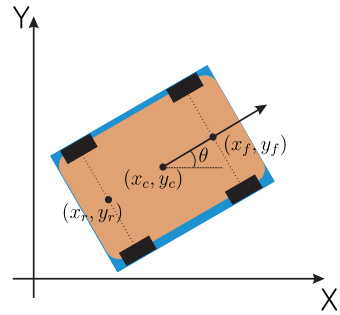


Fig. 1: Car-like obstacle in 2D environment. By symmetry, the simplest motion pattern will be obtained for the following three candidates of state: $[x_r, y_r, \theta]^\top$, $[x_f, y_f, \theta]^\top$ and $[x_c, y_c, \theta]^\top$, where (x_r, y_r), (x_f, y_f) and (x_c, y_c) are the coordinates of the center of the rear axle, front axle, and center of mass, respectively, with θ as the heading angle. The region occupied by the vehicle is over-approximated by the blue rectangle.

[1]For simplicity, we use a single integrator kinematics even though a double integrator can be used for enhancing accuracy.

covariance:

$$\boldsymbol{\mu}_{\mathrm{v}}^{j}(\mathbf{x}) := m_j(\mathbf{x}) + K_j(\mathbf{x},\hat{\mathrm{x}})(K_j(\hat{\mathrm{x}},\hat{\mathrm{x}}) + \sigma_{\varepsilon,j}^2 I)^{-1}(\hat{\mathrm{v}}_j - m_j(\hat{\mathrm{x}})), \quad (3)$$

$$\boldsymbol{\Sigma}_{\mathrm{v}}^{j}(\mathbf{x}) = k_j(\mathbf{x},\mathbf{x}) - K_j(\mathbf{x},\hat{\mathrm{x}})(K_j(\hat{\mathrm{x}},\hat{\mathrm{x}}) + \sigma_{\varepsilon,j}^2 I)^{-1}K^j(\hat{\mathrm{x}},\mathbf{x}). \quad (4)$$

The resulting GP approximation of v_o is then given by

$$\mathbf{v}(\mathbf{x}) \sim \mathcal{GP}(\boldsymbol{\mu}_{\mathrm{v}}(\mathbf{x}), \boldsymbol{\Sigma}_{\mathrm{v}}(\mathbf{x})),$$

where $\boldsymbol{\mu}_{\mathrm{v}}(\mathbf{x}) = [\boldsymbol{\mu}_{\mathrm{v}}^1(\mathbf{x}), \ldots, \boldsymbol{\mu}_{\mathrm{v}}^{n_{\mathrm{x}}}(\mathbf{x})]^\top$, and $\boldsymbol{\Sigma}_{\mathrm{v}}(\mathbf{x}) = \mathrm{diag}([\boldsymbol{\Sigma}_{\mathrm{v}}^1(\mathbf{x}), \ldots, \boldsymbol{\Sigma}_{\mathrm{v}}^{n_{\mathrm{x}}}(\mathbf{x})])$.

C. Prediction of Obstacle's Motion

Assuming that $\mathrm{x}_o(0) \sim \mathcal{N}(\mu_{\mathrm{x}}(0), \mathbf{0})$, it is straightforward to check that $\mathrm{x}_o(t)$ is normally distributed at each stage t with mean $\mu_{\mathrm{x}}(t)$ and covariance $\Sigma_{\mathrm{x}}(t)$ to be specified. Having the posterior of the velocity vector, the state of the obstacle at the next stage can be predicted by considering the following joint distribution of the state and velocity vectors [5]:

$$\begin{bmatrix} \mathrm{x}_o(t) \\ \mathbf{v}(t) \end{bmatrix} \sim \mathcal{N}\Big(\begin{bmatrix} \mu_{\mathrm{x}}(t) \\ \mu_{\mathrm{v}}(t) \end{bmatrix}, \begin{bmatrix} \Sigma_{\mathrm{x}}(t) & \Sigma_{\mathrm{xv}}(t) \\ \Sigma_{\mathrm{vx}}(t) & \Sigma_{\mathrm{v}}(t) \end{bmatrix} \Big).$$

Following procedures in [20] and [21] and applying the first-order Taylor approximation to (3) and (4) with Gaussian input $\mathrm{x}_o(t) \sim \mathcal{N}(\mu_{\mathrm{x}}(t), \Sigma_{\mathrm{x}}(t))$ yields the following approximate mean and covariance functions:

$$\begin{aligned} \tilde{\mu}_{\mathrm{v}}(t) &= \boldsymbol{\mu}_{\mathrm{v}}(\mu_{\mathrm{x}}(t)) \\ \tilde{\Sigma}_{\mathrm{v}}(t) &= \boldsymbol{\Sigma}_{\mathrm{v}}(\mu_{\mathrm{x}}(t)) + \nabla\boldsymbol{\mu}_{\mathrm{v}}(\mu_{\mathrm{x}}(t))\Sigma_{\mathrm{x}}(t)\nabla\boldsymbol{\mu}_{\mathrm{v}}(\mu_{\mathrm{x}}(t))^\top \\ \tilde{\Sigma}_{\mathrm{xv}}(t) &= \Sigma_{\mathrm{x}}(t)\nabla\boldsymbol{\mu}_{\mathrm{v}}(\mu_{\mathrm{x}}(t))^\top. \end{aligned} \quad (5)$$

Now, it follows from (1) that the obstacle's state at the next stage is also normally distributed with the following mean and covariance:

$$\begin{aligned} \mu_{\mathrm{x}}(t+1) &= \mu_{\mathrm{x}}(t) + T_o\mu_{\mathrm{v}}(t) \\ \Sigma_{\mathrm{x}}(t+1) &= \Sigma_{\mathrm{x}}(t) + T_o^2\Sigma_{\mathrm{v}} + T_o(\Sigma_{\mathrm{xv}} + \Sigma_{\mathrm{vx}}). \end{aligned} \quad (6)$$

Using (5) and (6), the approximate mean and variance of $\mathrm{x}_o(t)$ can be updated.

Having the inferred or predicted obstacle state $\mathrm{x}_o(t)$, it is straightforward to obtain g_t and G_t in (2) as $g_t = \mathrm{g}(\mathrm{x}_o(t))$ and $G_t = \mathrm{G}(\mathrm{x}_o(t))$. An example of predicting the motion of an obstacle is shown in Fig. 2, where a car-like vehicle is chosen as the obstacle with unknown dynamics. GP regression is used to predict the trajectory of the vehicle for the next 10 stages. As shown in Fig. 2a, the predicted mean in an early stage ($t = 5$) deviates from the actual trajectory, as there were no observations available. As more data are collected, the robot better learns the motion pattern of the car-like obstacle. As a result, in Figures 2b the difference between the predicted mean and the actual one is small.

However, in practice, the motion predicted by GP regression can be quite different from the actual movement of an obstacle, for example, when it abruptly changes the heading angle, as in the case of Fig. 2c. To guarantee safety even when learning fails, we propose a distributionally robust motion control tool in the following section.

III. Learning-Based Distributionally Robust Motion Control

Consider a mobile robot navigating in $\mathbb{R}^{n_y}$ according to the following discrete-time dynamics:

$$\xi(t+1) = f(\xi(t), u(t)) \quad (7)$$

$$y(t) = h(\xi(t), u(t)), \quad (8)$$

where $\xi(t) \in \mathbb{R}^{n_\xi}$ and $u(t) \in \mathbb{R}^{n_u}$ are the robot's state and control inputs, respectively, and $y(t) \in \mathbb{R}^{n_y}$ is the robot's current position in the n_y-dimensional configuration space. At stage t, the robot is subject to the following state and control constraints:

$$\xi(t) \in \Xi(t), \quad u(t) \in \mathcal{U}(t), \quad (9)$$

where $\Xi(t) \subseteq \mathbb{R}^{n_\xi}$ and $\mathcal{U}(t) \subseteq \mathbb{R}^{n_u}$.

The robot's environment changes over time as the obstacle moves according to its unknown dynamics. As introduced in our previous work [22], the *safe region* regarding the obstacle is defined by the complement of the region occupied by it, i.e.

$$\mathcal{Y}(t) := \mathbb{R}^{n_y} \setminus \mathcal{O}^o(t) \quad \forall t \geq 0,$$

where $\mathcal{O}^o$ denotes the interior of $\mathcal{O}(t)$. Our goal is to control the robot while keeping it in the *safe region*, even when the GP-based prediction results are inaccurate.

A. Risk Constraint for Safety

To systematically measure the risk of collision, we use the notion of *safety risk* introduced in our previous work [23]. We first define the *loss of safety* as the deviation of the robot's position from the safe region $\mathcal{Y}(t)$:

$$\mathrm{dist}(y(t), \mathcal{Y}(t)) := \min_{a \in \mathcal{Y}(t)} \|y(t) - a\|_2. \quad (10)$$

For safety, it is ideal to force the robot to stay inside the safe region. However, due to the uncertain movement of the obstacle, such a deterministic approach is often too conservative or infeasible. Instead, we employ the conditional value-at-risk (CVaR) [24] to define the safety risk at stage t as

$$\mathrm{CVaR}_\alpha[\mathrm{dist}(y(t), \mathcal{Y}(t))],$$

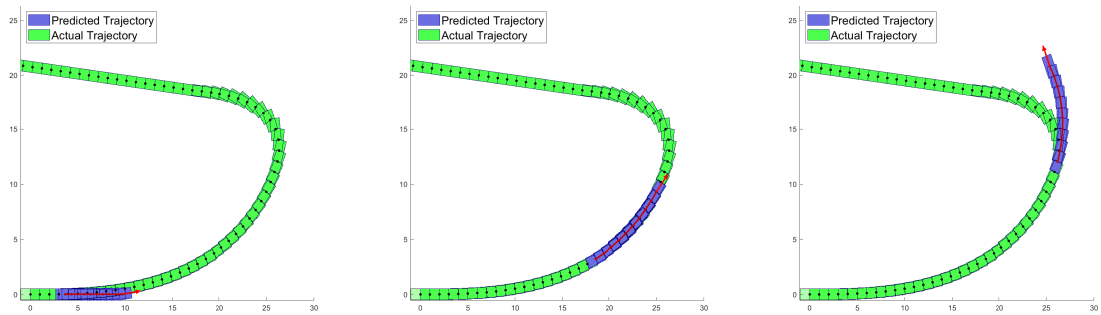
where $\mathrm{CVaR}_\alpha(X) := \min_{z\in\mathbb{R}} \mathbb{E}\big[z + (X - z)^+/(1-\alpha)\big]$.[2] The safety risk quantifies the average loss of safety beyond the confidence level α. Note that CVaR is a *coherent* risk measure in the sense of Artzner *et al.* [25] and thus satisfies *axioms* that risk metrics in robotics applications should respect for rationally assessing risk [26]. More importantly, CVaR is able to distinguish the worst-case tail events, which is crucial for quantifying rare but unsafe events.

The desired level of safety can be reached by limiting the safety risk by a pre-specified risk tolerance parameter δ:

$$\mathrm{CVaR}_\alpha[\mathrm{dist}(y(t), \mathcal{Y}(t))] \leq \delta. \quad (11)$$

This risk constraint is adopted in our MPC for safe motion control in the following subsection.

[2] We let $(\boldsymbol{x})^+ := \max\{\boldsymbol{x}, 0\}$ throughout this paper.



(a) $t = 5$ (b) $t = 20$ (c) $t = 32$

Fig. 2: Predicted mean trajectories of a car-like obstacle for the next 10 stages.

B. Wasserstein Distributionally Robust GP-MPC

The safe region in (11) depends on g_t and G_t, which define the region occupied by the obstacle at stage t. Unfortunately, the distribution of these two parameters is unknown and challenging to directly identify in practice. However, having sample data $\{\tilde{\mathrm{x}}_o^{(1)}(t), \tilde{\mathrm{x}}_o^{(2)}(t), \ldots, \tilde{\mathrm{x}}_o^{(N)}(t)\}$ generated according to the learned distribution of $\mathrm{x}_o(t)$, it is possible to obtain a sample of g_t and G_t using

$$\tilde{g}_t^{(i)} := \mathrm{g}(\tilde{\mathrm{x}}_o^{(i)}(t)), \quad \tilde{G}_t^{(i)} := \mathrm{G}(\tilde{\mathrm{x}}_o^{(i)}(t)). \tag{12}$$

We can then use the sample data to approximate the safety risk in (11). However, making such an approximation using limited data may lead to the violation of the original risk constraint (11). Instead of directly using the learning result of GP regression, we proposed a motion control method that is robust against errors in the estimated distribution.

For a concrete MPC formulation, we first rewrite the loss of safety (10) in an equivalent form using the definition of the safe region, which is a union of half-spaces.

Lemma 1. *Suppose that the region occupied by the obstacle is given by* (2)*. Then, the loss of safety* (10) *can be expressed as*

$$\mathrm{dist}(y(t), \mathcal{Y}(t)) = \min_{j=1,\ldots,m} \left\{ \frac{\left(g_{t,j} - G_{t,j} y(t)\right)^+}{\|G_{t,j}\|_2} \right\}, \tag{13}$$

where $g_{t,j}$ is the jth element of g_t, and $G_{t,j}$ is the jth row of G_t.

Proof. The proof is similar to the proof of [27, Lemma 1], which we briefly summarize here. Since the safe region is a union of half-spaces, the distance can be written as the shortest distance to the all half-spaces that define the safe region:

$$\mathrm{dist}(y(t), \mathcal{Y}(t)) = \min_{j=1,\ldots,m} \mathrm{dist}(y(t), \mathcal{Y}_j(t)) \tag{14}$$

where $\mathcal{Y}_j(t) = \{\mathbf{x} \mid G_{t,j}\mathbf{x} \geq g_{t,j}\}$. The distance to each half-space can then be expressed in its dual form as

$$\mathrm{dist}(y(t), \mathcal{Y}_j(t)) = \left(\frac{g_{t,j} - G_{t,j} y(t)}{\|G_{t,j}\|_2} \right)^+ \tag{15}$$

using an argument similar to the proof of [27, Lemma 1]. Strong duality follows from the fact that the primal problem is feasible and the inequality constraints are linear. By substituting (15) into (14), the result follows. □

We now let

$$c_{t,j} := -\frac{G_{t,j}}{\|G_{t,j}\|_2}, \quad d_{t,j} := \frac{g_{t,j}}{\|G_{t,j}\|_2}. \tag{16}$$

Using the sample data (12) of $\tilde{g}_t^{(i)}$ and $\tilde{G}_t^{(i)}$, we can then generate a sample $\{(\tilde{c}_{t,j}^{(i)}, \tilde{d}_{t,j}^{(i)})\}_{i=1}^N$ of $(c_{t,j}, d_{t,j})$ according to the definition above. Let Q_t be the joint empirical distribution of $(c_t, d_t) \in \mathbb{W} \subseteq \mathbb{R}^{m(n_y+1)}$ constructed using the sample data, i.e., $\mathrm{Q}_t := \sum_{i=1}^N \boldsymbol{\delta}_{(\tilde{c}_t^{(i)}, \tilde{d}_t^{(i)})}$, where $\boldsymbol{\delta}_{\boldsymbol{x}}$ denotes the Dirac delta measure concentrated at $\boldsymbol{x}$. However, the accuracy of the empirical distribution is subject to errors in the learning results. To satisfy the risk constraint (11) even under distribution errors, we instead impose the following distributionally robust risk constraint:

$$\sup_{\mathrm{P}_t \in \mathbb{D}_t} \mathrm{CVaR}_\alpha^{\mathrm{P}_t}[\mathrm{dist}(y(t), \mathcal{Y}(t))] \leq \delta. \tag{17}$$

Here, the left-hand side of the inequality represents the worst-case CVaR when the joint distribution P_t of (c_t, d_t) lies in a given *ambiguity set* $\mathbb{D}_t$. Thus, any motion control action that satisfies (17) can meet the original risk constraint under any distribution error characterized by $\mathbb{D}_t$. In this work, we use the following *Wasserstein ambiguity set*:

$$\mathbb{D}_t := \{\mathrm{P} \in \mathcal{P}(\mathbb{W}) \mid W(\mathrm{P}, \mathrm{Q}_t) \leq \theta\}, \tag{18}$$

where $\mathcal{P}(\mathbb{W})$ denotes the set of Borel probability measures on the support $\mathbb{W}$. Here, $W(\mathrm{P}, \mathrm{Q})$ is the Wasserstein distance (of order 1) between P and Q, defined by

$$W(\mathrm{P}, \mathrm{Q}) := \min_{\kappa \in \mathcal{P}(\mathbb{W}^2)} \left\{ \int_{\mathbb{W}^2} \|w - w'\|_2 \, \mathrm{d}\kappa(w, w') \mid \Pi^1\kappa = \mathrm{P}, \Pi^2\kappa = \mathrm{Q} \right\},$$

where $\Pi^i\kappa$ denotes the ith marginal of κ for $i = 1, 2$. The Wasserstein distance between two probability distributions represents the minimum cost of redistributing mass from one to another using a non-uniform perturbation. Using the Wasserstein metric in distributionally robust optimization and control has recently drawn a great deal of interest because it provides a tractable solution with superior statistical

properties such as a probabilistic out-of-sample performance guarantee [17], [28]–[32].

Using the distributionally robust risk constraint (17), we formulate the following MPC problem:

$$\inf_{\mathbf{u},\boldsymbol{\xi},\mathbf{y}} J(\xi(t),\mathbf{u}) := \sum_{k=0}^{K-1} r(\xi_k,u_k) + q(x_K) \tag{19a}$$

$$\text{s.t. } \xi_{k+1} = f(\xi_k,u_k) \tag{19b}$$

$$y_k = h(\xi_k,u_k) \tag{19c}$$

$$\xi_0 = \xi(t) \tag{19d}$$

$$\sup_{\mathrm{P}_k\in\mathbb{D}_k} \mathrm{CVaR}_\alpha^{\mathrm{P}_k}[\mathrm{dist}(y_k,\mathcal{Y}_k)] \le \delta \tag{19e}$$

$$\xi_k \in \Xi,\ u_k \in \mathcal{U}, \tag{19f}$$

where $\mathbf{u} := (u_0,\dots,u_{K-1})$, $\boldsymbol{\xi} := (\xi_0,\dots,\xi_K)$, $\mathbf{y} := (y_0,\dots,y_K)$, constraint (19b) and $u_k \in \mathcal{U}$ in (19f) should hold for $k = 0,\dots,K-1$, (19c) should hold for $k = 0,\dots,K$, and all the remaining constraints should be satisfied for $k = 1,\dots,K$. Note that the problem can be extended to consider L obstacles by repeating the constraints (19e) L times.

The distributionally robust MPC (DR-MPC) problem with GP is defined in a receding horizon manner for each stage. The cost function can be chosen in a way that would guide the robot so it follows a reference trajectory y^{ref} generated by, for example, RRT* [33]:

$$J(\xi(t),\mathbf{u}) := \|y_K - y_K^{ref}\|_P + \sum_{k=0}^{K-1} \|y_k - y_k^{ref}\|_Q + \|u_k\|_R, \tag{20}$$

where $Q \succeq 0$, $R \succ 0$ are the state and control weighting matrices, respectively; and $P \succeq 0$ is chosen in a way to ensure stability. The constraints (19b) and (19c) are used for computing the robot state and output over the MPC horizon, specifying the initial state ξ_0 as the current state $\xi(t)$ in the constraint (19d). Most importantly, (19e) corresponds to the distributionally robust risk constraint, thereby limiting the safety risk by a pre-specified tolerance even when the actual distribution deviates from the distribution estimated by GP regression within $\mathbb{D}_k$. Here, the Wasserstein ambiguity set $\mathbb{D}_k$ is constructed from the joint empirical distribution Q_k of (c_k,d_k) at each time step k. The joint distribution is obtained from GP regression, by learning the obstacle's velocity vector and evolving the obstacle's state according to (6) from $\mathrm{x}_o(t)$ to $\mathrm{x}_o(t+K)$. Finally, (19f) are the state and control constraints given in (9).

C. Tractable Reformulation

Unfortunately, solving the DR-MPC problem (19) is a challenging task because the risk constraint (19e) involves an infinite-dimensional optimization problem over the ambiguity set of probability distributions. To resolve this issue, we reformulate the DR-MPC problem in a computationally tractable form.

Algorithm 1: Learning-based DR-MPC at stage t

Input: $\xi(t), \mathrm{x}_o(t), \hat{\mathrm{x}}^{(i)}, \hat{\mathrm{v}}^{(i)}$, $i = 1,\dots,M$;
$\mathcal{D}_j := \{(\hat{\mathrm{x}}^{(i)}, \hat{\mathrm{v}}_j^{(i)}),\ i = 1,\dots,M\}$, $j = 1,\dots,n_{\mathrm{x}}$;
Initialize $\mu_{\mathrm{x}}(0) := \mathrm{x}_o(t)$, $\Sigma_{\mathrm{x}}(0) := \mathbf{0}$;
for $k = 0 : K-1$ **do**
  Compute $\tilde{\mu}_{\mathrm{v}}(k)$, $\tilde{\Sigma}_{\mathrm{v}}(k)$ and $\tilde{\Sigma}_{\mathrm{xv}}$ from (5);
  Update $\mu_{\mathrm{x}}(k+1)$ and $\Sigma_{\mathrm{x}}(k+1)$ from (6);
  Generate a sample $\{\tilde{\mathrm{x}}_o^{(1)}(k+1),\dots,\tilde{\mathrm{x}}_o^{(N)}(k+1)\}$ from $\mathcal{N}(\mu_{\mathrm{x}}(k+1),\Sigma_{\mathrm{x}}(k+1))$;
  Compute $\tilde{c}_{k+1}^{(i)}$ and $\tilde{d}_{k+1}^{(i)}$, $i = 1,\dots,N$ using (12) and (16);
end
Solve (22) to obtain $\mathbf{u}^*$;
return $u(t) = u_0^*$;

To begin with, we make use of Lemma 1 to rewrite the safety risk as

$$\begin{aligned}\mathrm{CVaR}_\alpha[\mathrm{dist}(y,\mathcal{Y})] &= \min_{z\in\mathbb{R}} \mathbb{E}\left[z + \frac{(\mathrm{dist}(y,\mathcal{Y}) - z)^+}{1-\alpha}\right] \\ &= \min_{z\in\mathbb{R}} \left\{z + \mathbb{E}\left[\frac{\max\{\min_j(c_j y + d_j) - z, -z, 0\}}{1-\alpha}\right]\right\}.\end{aligned} \tag{21}$$

Next, the following proposition can be used to reformulate the distributionally robust risk constraint (19e) in a conservative manner, which is suitable for our purpose of limiting the risk of unsafety:

Proposition 1. *Suppose that $\mathbb{W} = \mathbb{R}^n$, where $n := m(n_y + 1)$ is the dimension of (c_t, d_t). Then, the following inequality holds:*

$$\begin{aligned}&\sup_{\mathrm{P}_t\in\mathbb{D}_t} \mathrm{CVaR}_\alpha^{\mathrm{P}_t}[\mathrm{dist}(y(t),\mathcal{Y}(t)] \\ &\le \inf_{z,\lambda,s,\rho} z + \frac{1}{1-\alpha}\left[\lambda\theta + \sum_{i=1}^N s_i\right] \\ &\quad \text{s.t. } \langle \rho_i, \tilde{c}_t^{(i)} y(t) + \tilde{d}_t^{(i)}\rangle \le s_i + z \\ &\quad\quad s_i + z \ge 0 \\ &\quad\quad s_i \ge 0 \\ &\quad\quad \sum_{j=1}^m \rho_{i,j}^2 \Big(\sum_{l=1}^{n_y} y_l^2 + 1\Big) \le \lambda^2 \\ &\quad\quad \lambda \ge 0 \\ &\quad\quad \langle \rho_i, e\rangle = 1 \\ &\quad\quad \rho_i \ge 0 \\ &\quad\quad z \in \mathbb{R},\end{aligned}$$

where all the constraints hold for $i = 1,\dots,N$, and $e \in \mathbb{R}^m$ is a vector of all ones. $\rho_{i,j}$ represents the jth element of ρ_i and y_l is the lth element of y.

Its proof follows directly from Lemma 2, [27, Proposition 1], and [17, Theorem 4.2]. The assumption that $\mathbb{W} = \mathbb{R}^n$ can be relaxed using [27, Proposition 1]. Note that

the optimization problem on the right-hand side is finite-dimensional, unlike the original one on the left-hand side. Thus, by limiting this upper-bound of the distributionally robust safety risk instead of (19e), we can completely remove the infinite-dimensionality issue inherent in the original DR-MPC problem (19).

Specifically, according to Proposition 1, the DR-MPC problem (19) can be reformulated as follows:

$$\inf_{\substack{\mathbf{u},\boldsymbol{\xi},\mathbf{y},\mathbf{z},\\ \lambda,s,\rho}} J(\xi(t),\mathbf{u}) := \sum_{k=0}^{K-1} r(\xi_k,u_k) + q(\xi_K) \tag{22a}$$

$$\text{s.t. } \xi_{k+1} = f(\xi_k,u_k) \tag{22b}$$

$$y_k = h(\xi_k,u_k) \tag{22c}$$

$$\xi_0 = \xi(t) \tag{22d}$$

$$z_k + \frac{1}{1-\alpha}\left[\lambda_k\theta + \frac{1}{N_k}\sum_{i=1}^{N_k} s_{k,i}\right] \le \delta \tag{22e}$$

$$\langle \rho_{k,i}, \tilde{c}_k^{(i)} y_k + \tilde{d}_k^{(i)} \rangle \le s_{k,i} + z_k \tag{22f}$$

$$s_{k,i} + z_k \ge 0 \tag{22g}$$

$$s_{k,i} \ge 0 \tag{22h}$$

$$\sum_{j=1}^{m} \rho_{k,i,j}^2 \Big(\sum_{l=1}^{n_y} y_{k,l}^2 + 1 \Big) \le \lambda_k^2 \tag{22i}$$

$$\lambda_k \ge 0 \tag{22j}$$

$$\langle \rho_{k,i}, e \rangle = 1 \tag{22k}$$

$$\rho_{k,i} \ge 0 \tag{22l}$$

$$z_k \in \mathbb{R}, \tag{22m}$$

$$\xi_k \in \Xi,\ u_k \in \mathcal{U}, \tag{22n}$$

where (22b) and $u_k \in \mathcal{U}$ in (22n) should hold for $k = 0,\dots,K-1$, (22c) should hold for $k = 0,\dots,K$, and all the other constraints should be satisfied for $k = 1,\dots,K$ and $i = 1,\dots,N$. As desired, the reformulated problem is finite-dimensional unlike the original one (19). However, it is a nonconvex optimization problem due to the constraints (22f) and (22i) even when the system dynamics and the output equation are affine and the cost function is convex. A locally optimal solution to this problem can be efficiently computed by using existing nonlinear programming algorithms such as interior-point methods (e.g., [34]).

The overall learning-based DR-MPC at stage t is shown in Algorithm 1. At each stage, the current states of the robot and the obstacle as well as M past observations $\{(\hat{\mathrm{x}}^{(i)}, \hat{\mathrm{v}}^{(i)})\}_{i=1}^{M}$ of the obstacle's position and velocity are taken as the input data. Then, the obstacle's movement for future stages is learned by GP regression and is used in the DR-MPC problem (22). The first element of the locally optimal solution $\mathbf{u}^*$ is taken as the motion control action for the robot at the current stage. Note that at stage $t = 0$, the dataset $\mathcal{D}$ consists of all zeros. As time goes on, new observations are added to the dataset for GP regression. During the update, old observations are removed so that only M latest data are stored.

IV. Experiment Results

In this section, we present simulation results to demonstrate the performance of our motion control method. In our experiments, we consider a car-like vehicle navigating a 2D environment with the following bicycle dynamics [35]:

$$\begin{aligned} x^v(t+1) &= x^v(t) + T_s v^v(t)\cos(\theta^v(t)+\beta^v(t)) \\ y^v(t+1) &= y^v(t) + T_s v^v(t)\sin(\theta^v(t)+\beta^v(t)) \\ \theta^v(t+1) &= \theta^v(t) + T_s v^v(t)\frac{\sin(\beta^v(t))}{l_r} \\ \beta^v(t+1) &= \beta^v(t) + T_s \tan^{-1}\Big(\frac{l_r}{l_r+l_f}\tan\delta^v(t)\Big), \end{aligned} \tag{23}$$

where $x^v(t)$ and $y^v(t)$ are the coordinates of the vehicle's center of gravity, $\theta^v(t)$ is the heading angle, $\beta^v(t)$ is the current velocity angle. The control inputs are velocity $v^v(t)$ and steering angle $\delta^v(t)$. The coefficients l_f and l_r represent the distances from the center of gravity to the front and rear wheels, respectively. Throughout the simulations, we assume that $l_f = l_r = 2$. We also impose the following control constraints:

$$v^v(k) \in [0,30], \quad \delta^v(k) \in [-\pi/6, \pi/6] \quad \forall k.$$

The vehicle is controlled to follow the centerline of the track while avoiding two dynamic obstacles. The centerline is thus taken as the reference trajectory y^{ref} in (20). The two obstacles are rectangular car-like vehicles with size 2×1. It is straightforward to check that for both obstacles g_k and G_k are easily found from the state that consists of the vehicle's center of mass and its heading angle. In our experiments, we set $Q = P = I$ and $R = 0.01I$. The sampling time T_s and T_o are set to be 0.01, and the MPC horizon is chosen as $K = 5$. The risk tolerance level and the confidence level were selected as $\delta = 0.01$ and $\alpha = 0.95$, respectively.

To evaluate the performance of learning-based DR-MPC, we compare it to its non-robust counterpart obtained by sample average approximation (SAA) [22]. All the simulations were conducted on a PC with 3.70 GHz Intel Core i7-8700K processor and 32 GB RAM. The optimization problem was modeled in AMPL [36] and solved using interior-point method-based solver IPOPT [37].

Fig. 3 shows the resulting trajectories for different sizes of the Wasserstein ambiguity set compared to the SAA version (SAA-MPC) with $N = 50$ samples. At each stage, the dataset for GP regression is updated to keep only the latest $M = 20$ observations.

In the early stages, the robotic vehicle follows the centerline while predicting the future motion of the obstacles. As shown in Fig. 3a, when reaching one obstacle that abruptly changes its heading angle at $t = 13$, the vehicle tries to avoid it. In the case of SAA-MPC, the vehicle collides with the obstacle because the distributional information learned by GP regression is inaccurate. As a result, the risk constraint is violated and the MPC problem becomes infeasible. Meanwhile, the vehicle controlled by our method successfully bypasses the obstacle. The safety margin increases with the radius θ of the Wasserstein ambiguity set.

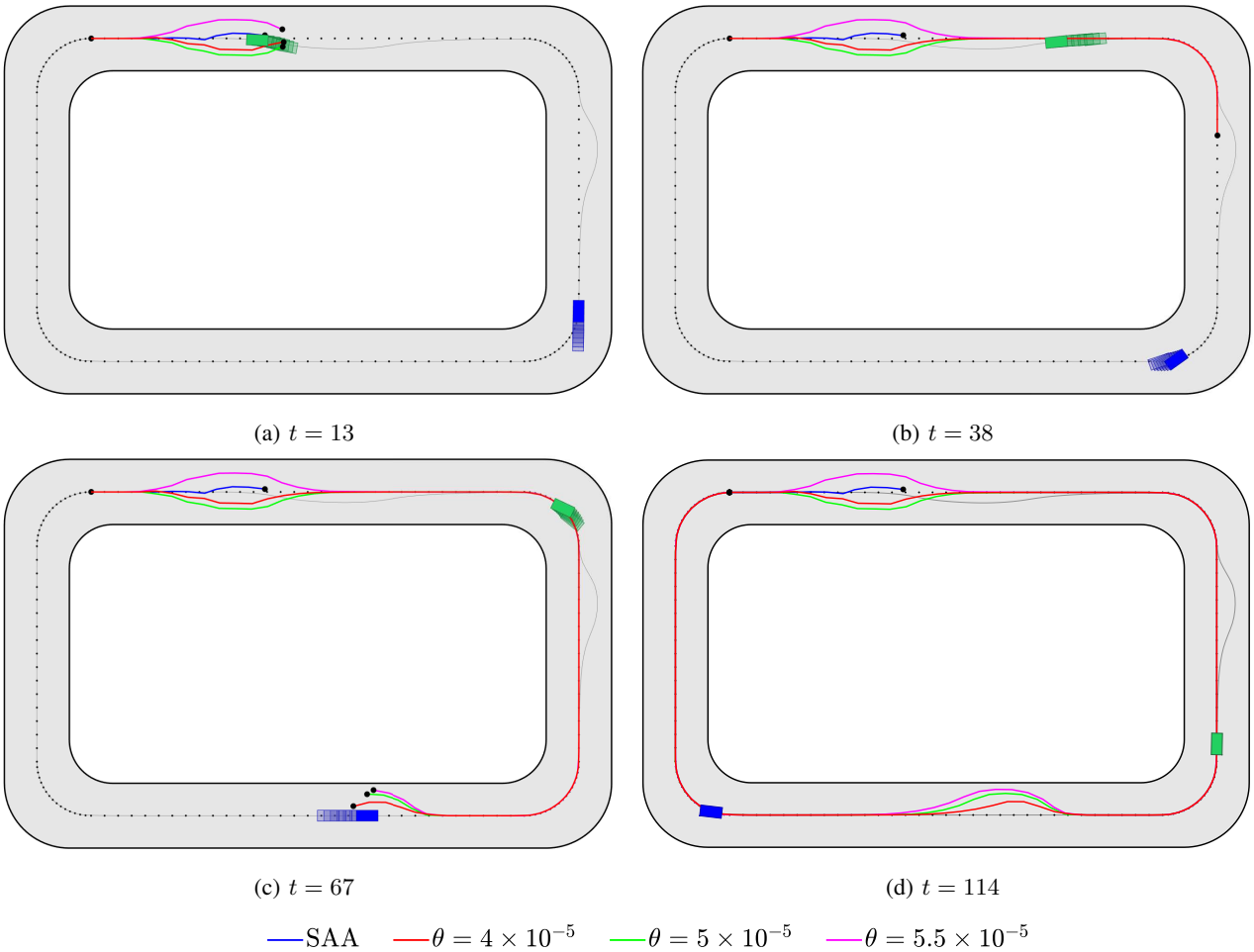


Fig. 3: The trajectories of the vehicle controlled by SAA-MPC and DR-MPC with $\theta = 4 \times 10^{-5}, 5 \times 10^{-5}$, and 5.5×10^{-5}. The current vehicle position is marked with a black dot. The green and blue rectangles represent the two obstacles, while the transparent ones are the K steps-ahead prediction of the obstacles, obtained via GP regression. The reference centerline for the vehicle is displayed with points, while the thin grey curve is the actual trajectory of the obstacles.

Fig. 3b shows the situation at $t = 38$, where the vehicle controlled by our method continues to follow the reference trajectory for all θ's. Meanwhile, the GP is not well enough to be able to predict the motion of the obstacles around the corners, although it shows good performance when there is no sudden change in the obstacle's movement. As shown in Fig. 3c, at $t = 67$ the second obstacle interferes with the path of the vehicle. Similar to the previous obstacle, the vehicle controlled by DR-MPC avoids the obstacle for all θ's. In the case of the smallest radius $\theta = 4 \times 10^{-5}$, the vehicle chooses to take aggressive action while satisfying the risk constraint. As the Wasserstein ambiguity set increases, i.e., θ increases, the robot makes a more conservative (i.e., safer) decision, inducing a bigger safety margin. Fig. 3d displays the trajectories for all cases after the vehicle completes one lap. Note that only the non-robust SAA version failed to complete the lap due to collision, while our method succeeded to do so for all θ's.

In summary, we conclude that the proposed distributionally robust method successfully preserves safety even with moderate errors in the learning results. In the case of very small ambiguity sets (e.g., $\theta = 4 \times 10^{-5}$), the resulting control action may be too aggressive to guarantee safety when the learning errors are significant. Whereas, for $\theta = 5.5 \times 10^{-5}$, the vehicle deviates too much from the reference trajectory, inducing a large cost. Based on our experiments, $\theta = 5 \times 10^{-5}$ may be selected for a good tradeoff between safety and cost.

TABLE I: Accumulated cost, lap time, and average computation time for the nonlinear car-like vehicle motion control with $N = 50$, $\delta = 0.01$, and $\alpha = 0.95$.

	SAA	DR-MPC (θ)		
		4×10^{-5}	5×10^{-5}	5.5×10^{-5}
Accumulated Cost	$+\infty$	491.79	594.68	703.59
Lap Time (sec)	-	105.26	109.45	110.31
Avg. Run Time (sec)	-	0.6572	0.6767	0.6942

Table I shows the accumulated cost and the amount of time for completing one lap on the track, and the average

computation time required for solving a single DR-MPC problem (22). As expected, both of the total cost and the lap time increase with θ since the vehicle controlled by DR-MPC with larger θ is more conservative and deviates further from the reference trajectory. Computation time is small in all cases although a nonconvex optimization problem is solved in each iteration. This result shows the potential of using our distributionally robust method in real-time applications.

V. Conclusion

We have proposed a distributionally robust decision-making tool for safe motion control of robotic vehicles in an environment with dynamic obstacles. Our DR-MPC method limits the risk of unsafety even with moderate errors in the obstacle's motion predicted by GP regression. For computational tractability, we have also developed a reformulation approach exploiting modern distributionally robust optimization techniques. The experimental results demonstrate the safety-preserving capability of our method under moderate learning errors and the potential for real-time application. In the future, the proposed method can be extended to enhance the capability of fast adaptive reactions, especially when considering sudden motion changes, and to address partial observability.

References

[1] 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.

[2] S. Di Cairano, D. Bernardini, A. Bemporad, and I. V. Kolmanovsky, "Stochastic MPC with learning for driver-predictive vehicle control and its application to HEV energy management," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 3, pp. 1018–1031, 2013.

[3] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," *The International Journal of Robotics Research*, vol. 35, no. 13, pp. 1547–1563, 2016.

[4] G. Williams, P. Drews, B. Goldfain, J. M. Rehg, and E. A. Theodorou, "Information-theoretic model predictive control: Theory and applications to autonomous driving," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1603–1622, 2018.

[5] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using Gaussian process regression," *IEEE Transactions on Control Systems Technology*, 2019.

[6] Q. M. Hester, T. and P. Stone, "RTMBA: A real-time model-based reinforcement learning architecture for robot control," in *IEEE International Conference on Robotics and Automation*, 2012.

[7] A. Venkatraman, R. Capobianco, L. Pinto, M. Hebert, D. Nardi, and J. A. Bagnell, "Improved learning of dynamics models for control," in *International Symposium on Experimental Robotics*, 2016.

[8] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 82, no. 2, pp. 153–173, 2017.

[9] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *IEEE International Conference on Robotics and Automation*, 2015.

[10] M. Herman, V. Fischer, T. Gindele, and W. Burgard, "Inverse reinforcement learning of behavioral models for online-adapting navigation strategies," in *IEEE International Conference on Robotics and Automation*, 2015.

[11] M. Wulfmeier, D. Rao, D. Z. Wang, P. Ondruska, and I. Posner, "Large-scale cost function learning for path planning using deep inverse reinforcement learning," *The International Journal of Robotics Research*, vol. 36, no. 10, pp. 1073–1087, 2017.

[12] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *IEEE Intelligent Vehicles Symposium*, 2017.

[13] F. Codevilla, M. Miiller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *IEEE International Conference on Robotics and Automation*, 2018.

[14] S. Chernova and M. Veloso, "Confidence-based policy learning from demonstration using Gaussian mixture models," in *International Joint conference on Autonomous Agents and Multiagent Systems*, 2007.

[15] D. Lenz, F. Diehl, M. T. Le, and A. Knoll, "Deep neural networks for markovian interactive scene prediction in highway scenarios," in *IEEE Intelligent Vehicles Symposium*, 2017.

[16] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. MIT Press, 2006.

[17] P. Mohajerin Esfahani and D. Kuhn, "Data-driven distributionally robust optimization using the Wasserstein metric: Performance guarantees and tractable reformulations," *Mathematical Programming*, vol. 171, no. 1-2, pp. 115–166, 2018.

[18] F. Berkenkamp and A. P. Schoellig, "Safe and robust learning control with Gaussian processes," in *European Control Conference*, 2015, pp. 2496–2501.

[19] J. Umlauft and S. Hirche, "Learning stochastically stable Gaussian process state-space models," *IFAC Journal of Systems and Control*, p. 100079, 2020.

[20] A. Girard, C. Rasmussen, and R. Murray-Smith, "Gaussian process priors with uncertainty inputs: multiple-step-ahead prediction," *Technical Report TR-2002–119, Dept. of Computer Science*, 2002.

[21] A. Girard, C. E. Rasmussen, J. Q. Candela, and R. Murray-Smith, "Gaussian process priors with uncertain inputs application to multiple-step ahead time series forecasting," in *Advances in Neural Information Processing Systems*, 2003.

[22] A. Hakobyan, G. C. Kim, and I. Yang, "Risk-aware motion planning and control using CVaR-constrained optimization," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3924–3931, 2019.

[23] S. Samuelson and I. Yang, "Safety-aware optimal control of stochastic systems using conditional value-at-risk," in *American Control Conference*, 2018.

[24] R. T. Rockafellar and S. Uryasev, "Conditional value-at-risk for general loss distribution," *Journal of Banking & Finance*, vol. 26, pp. 1443–1471, 2002.

[25] P. Artzner, F. Delbaen, J.-M. Eber, and D. Heath, "Coherent measures of risk," *Mathematical Finance*, vol. 9, no. 3, pp. 203–228, 1999.

[26] A. Majumdar and M. Pavone, "How should a robot assess risk? towards an axiomatic theory of risk in robotics," in *International Symposium on Robotics Research*, 2017.

[27] A. Hakobyan and I. Yang, "Wasserstein distributionally robust motion control for collision avoidance using conditional value-at-risk," *arXiv preprint arXiv:2001.04727*, 2020.

[28] R. Gao and A. J. Kleywegt, "Distributionally robust stochastic optimization with Wasserstein distance," *arXiv:1604.02199*, 2016.

[29] C. Zhao and Y. Guan, "Data-driven risk-averse stochastic optimization with Wasserstein metric," *Operations Research Letters*, vol. 46, no. 2, 2018.

[30] J. Blanchet, K. Murthy, and F. Zhang, "Optimal transport based distributionally robust optimization: Structural properties and iterative schemes," *arXiv:1810.02403*, 2018.

[31] I. Yang, "A convex optimization approach to distributionally robust Markov decision processes with Wasserstein distance," *IEEE Control Systems Letters*, vol. 1, no. 1, pp. 164–169, 2017.

[32] ——, "Wasserstein distributionally robust stochastic control: A data-driven approach," *arXiv preprint arXiv:1812.09808.*, 2018.

[33] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The International Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.

[34] J. Nocedal and S. Wright, *Numerical Optimization*. Springer Science & Business Media, 2006.

[35] P. Polack, F. Altché, B. d'Andréa Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *IEEE Intelligent Vehicles Symposium*, 2017.

[36] R. Fourer, D. M. Gay, and B. W. Kernighan, "A modeling language for mathematical programming," *Management Science*, vol. 36, no. 5, pp. 519–554, 1990.

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