Learning-based Near-optimal Motion Planning for Intelligent Vehicles with Uncertain Dynamics

Yang Lu, Xinglong Zhang, *Member, IEEE*, Xin Xu*, *Senior Member, IEEE*, Weijia Yao, *Member, IEEE*

***Abstract*—Motion planning has been an important research topic in achieving safe and flexible maneuvers for intelligent vehicles. However, it remains challenging to realize efficient and optimal planning in the presence of uncertain model dynamics. In this paper, a sparse kernel-based reinforcement learning (RL) algorithm with Gaussian Process (GP) Regression (called GP-SKRL) is proposed to achieve online adaption and near-optimal motion planning performance. In this algorithm, we design an efficient sparse GP regression method to learn the uncertain dynamics. Based on the updated model, a sparse kernel-based policy iteration algorithm with an exponential barrier function is designed to learn the near-optimal planning policies with the capability to avoid dynamic obstacles. Thereby, batch-mode GP-SKRL with online adaption capability can estimate the changing system dynamics. The converged RL policies are then deployed on vehicles efficiently under a safety-aware module. As a result, the produced driving actions are safe and less conservative, and the planning performance has been noticeably improved. Extensive simulation results show that GP-SKRL outperforms several advanced motion planning methods in terms of average cumulative cost, trajectory length, and task completion time. In particular, experiments on a Hongqi E-HS3 vehicle demonstrate that superior GP-SKRL provides a practical planning solution.**

***Index Terms*—Motion planning, uncertain dynamics, kernel feature, reinforcement learning, Gaussian process.**

I. Introduction

Intelligent vehicles are promising in reducing traffic accidents due to human factors. As an important and indispensable part of autonomous driving, kinodynamic motion planning algorithms help improve the safety and maneuvering capabilities of intelligent vehicles by planning multistep-ahead trajectory profiles under the dynamic constraints and state limitations [1], [2]. In complex and unstructured road scenarios, realizing optimal kinodynamic motion planning still faces challenges, caused by the varying tire-ground friction coefficient, uncertain model parameters, measurement noise, etc.

Model predictive control (MPC) algorithms generate trajectories by solving optimal control problems under collision constraints [3]–[5]. Since MPC methods rely on an accurate model, uncertain vehicle dynamics could lead to performance degradation in planning and control. Therefore, it is necessary to estimate the model uncertainties online [6]–[8]. On the other hand, online solving nonlinear or non-convex optimization problems may lack the reliability to obtain feasible solutions and may be computationally intensive.

Yang Lu, Xinglong Zhang, and Xin Xu are with the College of Intelligence Science and Technology, National University of Defense Technology, Changsha, Hunan, 40073 P.R. China. (e-mail: luyang18@mail.sdu.edu.cn, zhangxinglong18@nudt.edu.cn, xuxin_mail@263.net).

Weijia Yao is with the School of Robotics, Hunan University, Changsha, Hunan, 410082, P.R. China.

RL and ADP have received significant attention in recent years for solving optimal planning and control problems [9], [10]. Previous RL approaches in solving planning problems with uncertain dynamics employ the integration of Pontryagin's Maximum Principle to address model uncertainty and kinodynamic constraints [11], [12], incorporate model learning strategy and barrier function into the cost function to separately handle model uncertainty and collision constraints [13], [14], measure and select the optimal solution among the planning candidates to address the performance degradation caused by model uncertainty [15], [16]. RL approaches necessitate the use of approximators e.g., actor-critic networks to approximate the optimal policy and value function for continuous control tasks, where the feature representation capability greatly impacts learning efficiency and performance [17], [18]. Due to the requirement for online fast adaptation ability to dynamic environments of motion planning algorithms, it is essential to enhance the feature representation capability for improving the learning efficiency. In our work, we adopt lightweight basis functions to construct features of actor and critic for online efficient learning since deep neural networks are usually with the deep RL framework where the control policy is trained offline and deployed online. Still, expanding the sample size for better feature representations may increase the dimensions of the basis function and lead to higher computational complexity. As far as we know, no prior RL-based planning approaches have addressed automatic feature construction for efficient and near-optimal online policy learning.

Motivated by the above challenge, we propose a sparse kernel-based RL algorithm for online efficient and near-optimal motion planning with uncertain dynamics. Specifically, we utilize a quantified kernel sparsification technique [19], [20] to enhance the feature representation ability. In a unified manner, this kernel sparsification technique is manifested in basis functions for both the actor and critic in RL and for Gaussian processes (GPs), thereby improving the online adaptation capability (please refer to the results in Sec. V-A5). Our approach differs from optimization-based methods [6]–[8], as it allows for offline training and rapid online deployment and can generalize well to similar scenarios. In contrast, optimization-based methods may face issues with computational efficiency and reliability when solving nonlinear optimization problems online.

II. Related Work

We discuss GP-based techniques and present literature review on model-based RL approaches (MBRLs) for systems with uncertain dynamics and state constraints.

GP-based model identification. Rasmussen [21] presented the idea of formulating GP models as a Bayesian framework for regression problems of stochastic processes. Sparse GPs were previously proposed to reduce the computational load by exploiting the structure of matrices. By selecting inducing points corresponding to the local trajectory, sparse GPs were incorporated into MPC algorithms to reduce the conservativeness [6]–[8]. It is assumed that the reference signals typically exhibit minimal variation between adjacent time steps. Motivated by these sparse GP-based methods, we adopt a quantification approach of approximate linear dependence (ALD) [19], [20] based on correlation measurement to obtain less correlated samples. Leveraging this strategy, we integrate it into the RL planning algorithm to address the trade-off between efficiency and feature representation capability in solving nonlinear optimization problems.

MBRL-based optimal control for systems with uncertain dynamics. In [22], PILCO propagated model uncertainties for a feedback policy by using a gradient-based policy search in planning horizons. It is data-efficient for learning control policies from scratch. In [23], [24], MPC algorithms with neural network (NN) based system dynamics were designed to improve the efficiency of MBRL. Related work on improving the sample efficiency can also be found in [25], [26]. These approaches focus on providing training samples for MBRL algorithms to enhance training efficiency. However, we adopt a sparse kernel-based technique to enable the online adaptation of the RL approach, thereby directly improving the planning and control performance. Under the MBRL framework, Bechtle et al. [27] proposed *Curious iLQR*, which combines iLQR and Bayesian modeling of uncertain dynamics. Extensive simulations and real-world experiments on 7-DoF manipulators validate its superiority. In contrast, our approach is a sparse kernel-based RL method trained with batch data samples and the updated model. Therefore, it allows for efficient training and rapid online deployment and can generalize well to similar scenarios. In this manner, it does not need to solve nonlinear optimization problems at each time step.

MBRL-based motion planning for systems with uncertain dynamics. Kamthe et al. [11] utilized probabilistic MPC to generate trajectories, and employed Pontryagin's Maximum Principle to generate safe control policies. In [12], the unknown disturbances are considered in the kinodynamic motion problem and solved by Pontryagin's Maximum Principle. Zhang et al. [13] proposed a receding-horizon RL algorithm with an NN-based uncertain dynamics model for motion planning of intelligent vehicles, while it is generally difficult to collect a sufficient number of samples. In [14], a barrier transformation was incorporated into the RL algorithm for generating optimal control inputs online, but there may exist issues of slow convergence speed and difficulty in selecting appropriate parameters. Performance measurement-based approaches can be found in [15], [16]. In [15], it focuses on deciding when to linearize the system dynamics for mitigating the effect of uncertain dynamics. Bejjani et al. proposed using a sampling-based planner to generate planning results for learning the optimal value function, which is then further optimized through RL. In summary, it remains challenging for these studies to ensure the efficiency and safety of online training, while our model-based batch RL with online adaption capability provides an effective and practical solution for vehicle kinodynamic motion planning.

Contribution. 1) We propose a sparse kernel-based RL motion planning algorithm, called GP-SKRL, which updates its policies in a batch training mode. In a unified manner, GP-SKRL utilizes a sparse kernel-based quantification technique to estimate the uncertain dynamics and train the RL policies. This enhances online adaption capability and the convergence of the algorithm is mathematically proved. 2) The proposed RL planning algorithm can maintain near-optimal performance under uncertain vehicle dynamics and collision constraints. Extensive simulation and experimental results show that GP-SKRL outperforms several advanced optimization-based methods. 3) Due to the consideration of dynamic characteristics during the learning process, GP-SKRL can directly compute control sequences for intelligent vehicles. In particular, real-world experiments on a Hongqi E-HS3 electric vehicle validate its efficiency and effectiveness.

III. Preliminaries

A. Problem Formulation

Consider the discrete-time vehicle dynamics in [6]:

$$x_{k+1} = f_{\text{nom}}\left(x_k, u_k\right) + B_d \underbrace{\left(g(x_k, u_k) + w_k\right)}_{y_k}, \tag{1}$$

where $x_k \in \mathbb{R}^{n_x}$ is the system state and $u_k \in \mathbb{R}^{n_u}$ is the control input. The model consists of a known nominal part f_{nom}, an additive term g, which lies within the subspace spanned by B_d [6]. We assume that the process noise $w_k \sim \mathcal{N}\left(0, \Sigma^w\right)$ is independent and identically distributed (i.i.d.), with spatially uncorrelated properties, i.e., $\Sigma^w = \text{diag}\{\sigma_1^2, \ldots, \sigma_{n_x}^2\}$. We also assume that both f_{nom} and g are differentiable. Here y_k is the uncertain dynamics, which are to be learned from data.

Consider the reference trajectory as follows:

$$x_{k+1,r} = \mathcal{P}_r(x_{k,r}, u_{k,r}), \tag{2}$$

where $x_{k,r}$ is the reference state, $u_{k,r}$ is the reference control input, and $\mathcal{P}_r(\cdot,\cdot)$ is a smooth mapping function. With Eqs. (1) and (2), one can obtain the error model (deferred in Appendix. A) as

$$\mathbf{x}_{k+1} = f\left(\mathbf{x}_k, \mathbf{u}_k\right),$$

where $\mathbf{x}$ is the error state and $\mathbf{u}$ is the control input.

Given an initial state $x_0 \in \mathbb{R}^{n_x}$, generate the real-time optimal control $\mathbf{u}_k^*$ at the k-th time instant. The infinite-horizon optimal motion-planning problem can be formulated as

$$\min_{\mathbf{u}_k} \; V(\mathbf{x}_k, \mathbf{u}_k), \tag{3}$$

and satisfy the following conditions: 1) Starts at x_0 and tracks the reference trajectory $\mathcal{P}_r$. 2) Avoids collisions with all obstacles $\mathcal{B}_1, \ldots, \mathcal{B}_q \subseteq \mathcal{W}$, where $\mathcal{W} \in \mathbb{R}^2$ denotes the Euclidean workspace. We define the value function as the cumulative discounted sum of infinite-horizon costs:

$$V\left(\mathbf{x}_k, \mathbf{u}_k\right) = \textstyle\sum_{i=k}^{\infty} \gamma^{i-k} L\left(\mathbf{x}_i, \mathbf{u}_i\right),$$

where $0 < \gamma \leq 1$ and we define $L(\mathbf{x}_i, \mathbf{u}_i) = \mathbf{x}_i^\top Q \mathbf{x}_i + \mathbf{u}_i^\top R \mathbf{u}_i$ as the stage cost function, where $Q \in \mathbb{R}^{n_x \times n_x}$ is positive semi-definite and $R \in \mathbb{R}^{n_u \times n_u}$ is positive definite.

The above optimal value function can be obtained by applying the optimal policy $\mathbf{u}^*$ to the system, which is computed by setting $\partial V / \partial \mathbf{u} = 0$, i.e.,

$$\mathbf{u}_k^* = -\tfrac{1}{2} \gamma R^{-1} (\partial \mathbf{x}_{k+1} / \partial \mathbf{u}_k)^\top \lambda^*(\mathbf{x}_{k+1}), \tag{4}$$

where the optimal costate $\lambda_{k+1}^* = \partial V^*(\mathbf{x}_{k+1}) / \partial \mathbf{x}_{k+1}$. With $L(\mathbf{x}_k, \mathbf{u}_k)$ and (3), the optimal costate is computed by

$$\lambda_k^* = 2Q\mathbf{x}_k + \gamma (\partial \mathbf{x}_{k+1} / \partial \mathbf{x}_k)^\top \lambda^*(\mathbf{x}_{k+1}). \tag{5}$$

B. Approximate Linear Dependence (ALD)

In terms of selecting dictionaries online, readers can refer to [8]. Then the dictionaries $\mathcal{D}_{\text{GP}}$ and $\mathcal{D}_{\text{RL}}$ undergo the sparsification process by utilizing ALD. We briefly outline the steps of ALD below (please refer to [19], [20] for details):

Step 1: Given each sample z_t from a collected sample set, the following optimization problem is formulated:

$$\delta_t = \min_c \left\| \sum_{j=1}^{t-1} c_j \phi(z_j) - \phi(z_t) \right\|^2, \tag{6}$$

where $c = [c_1, c_2, \cdots, c_{t-1}]^\top \in \mathbb{R}^{(t-1) \times n_z}$ constitutes the current sparse dictionary $\mathcal{D}_{t-1}$. According to the Mercer kernel theorem [28], there exists a mapping function $\phi(\cdot)$ that maps the state from the collected sample set to the Hilbert space $\mathcal{H}$. Since the Gaussian kernel function $k(\cdot, \cdot)$ satisfies the Mercer kernel condition, one can obtain the following equality:

$$k(z_i, z_j) = \langle \phi(z_i), \phi(z_j) \rangle,$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product in the Hilbert space. The distance δ_t between z_t and $\mathcal{D}_{t-1}$ can be computed by (6).

Step 2: The sparse dictionary $\mathcal{D}_t$ is updated by comparing δ_t with a preset threshold $\delta_{\max}$. If $\delta_t > \delta_{\max}$, $\mathcal{D}_t = \mathcal{D}_{t-1} \cup z_t$; Otherwise, we discard it; i.e., $\mathcal{D}_t = \mathcal{D}_{t-1}$.

The final kernel dictionaries $\mathcal{D}_{\text{SGP}}$ and $\mathcal{D}_{\text{SRL}}$ are used for sparse GP regression and sparse kernel-based RL learning processes, respectively.

C. Sparse GP Regression

Next, we will review a sparse GP regression method called FITC [29], which reduces computational complexity by selecting inducing samples and introduces a low-rank approximation of the covariance matrix, transforming the original GP model into an efficient one. It is briefly introduced in the following.

1) The formulation of full GP Regression: An independent training set is composed of state vectors, i.e., $\mathbf{z} = [z_1, z_2, \cdots, z_n]^\top \in \mathbb{R}^{n \times n_z}$ and the corresponding output vectors $\mathbf{y} = [y_1, y_2, \cdots, y_n]^\top \in \mathbb{R}^{n \times n_y}$. In [21], the mean and variance functions of each output dimension $a \in \{1, \cdots, n_y\}$ at a test point $z = [x^\top, u^\top]^\top$ are computed by

$$\begin{aligned} m_d^a &= K_{z\mathbf{z}}^a (K_{\mathbf{zz}}^a + I\sigma_a^2)^{-1} [\mathbf{y}]_a \\ \Sigma_d^a &= K_{zz}^a - K_{z\mathbf{z}}^a \left(K_{\mathbf{zz}}^a + \sigma_a^2 I\right)^{-1} K_{\mathbf{z}z}^a, \end{aligned} \tag{7}$$

where σ_a is the variance, $K_{\mathbf{zz}}^a = k^a(\mathbf{z}, \mathbf{z}) \in \mathbb{R}^{n_z \times n_z}$ is a Gram matrix containing variances of the training samples.

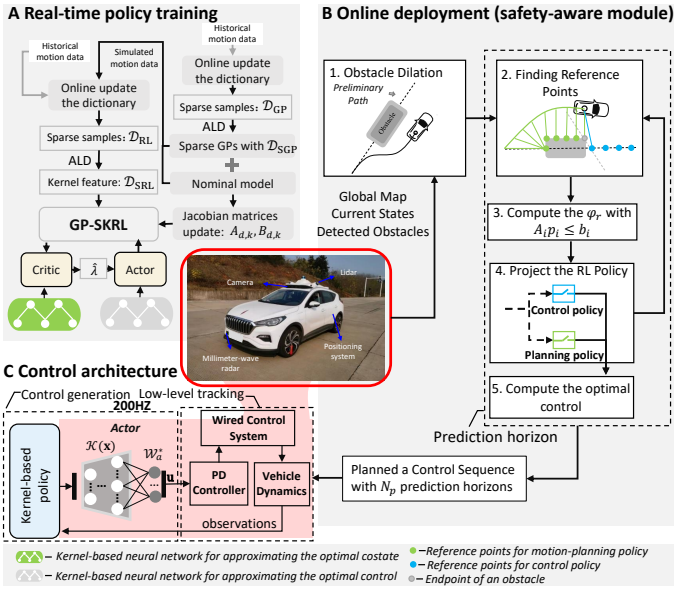

Fig. 1. Illustration of GP-SKRL-based motion planning.

Correspondingly, $K_{z\mathbf{z}}^a = (K_{\mathbf{z}z}^a)^\top = k^a(z, \mathbf{z})$ denotes the variance between a test sample and training samples, and $K_{zz}^a = k^a(z, z)$ represents the covariance, $k^a(\cdot, \cdot)$ is the kernel function and defined as follows:

$$k^a(z_i, z_j) = \sigma_{f,a}^2 \exp(-1/2 (z_i - z_j)^\top L_a^{-1} (z_i - z_j)), \tag{8}$$

where $\sigma_{f,a}^2$ is the signal variance and $L_a = \ell^2 I$. Here $\sigma_{f,a}$ and ℓ are hyperparameters of the covariance function.

2) Sparse GP Regression: Given a dictionary set $\{\mathbf{z}_{\text{ind}}, \mathbf{y}_{\text{ind}}\}$ with n_{ind} samples from $\{\mathbf{z}, \mathbf{y}\}$, the prior hyperparameters can be optimized by maximizing the marginal log-likelihood of the observed samples. In [29], the mean and variance functions of a full GP are approximated by using inducing targets $\mathbf{y}_{\text{ind}}$, inputs $\mathbf{z}_{\text{ind}}$, i.e.,

$$\begin{aligned} \tilde{m}_a^d(z) &= Q_{z\mathbf{z}}^a (Q_{\mathbf{zz}}^a + \Lambda)^{-1} [\mathbf{y}]_a, \\ \tilde{\Sigma}_a^d(z) &= K_{zz}^a - Q_{z\mathbf{z}}^a (Q_{\mathbf{zz}}^a + \Lambda)^{-1} Q_{\mathbf{z}z}^a, \end{aligned} \tag{9}$$

where $\Lambda = \text{diag}\{K_{\mathbf{zz}}^a - Q_{\mathbf{zz}}^a + I\sigma_a^2\}$ and the notation $Q_{\zeta\tilde{\zeta}}^a := K_{\zeta \mathbf{z}_{\text{ind}}}^a (K_{\mathbf{z}_{\text{ind}} \mathbf{z}_{\text{ind}}}^a)^{-1} K_{\mathbf{z}_{\text{ind}} \tilde{\zeta}}^a$. Several matrices in (9) do not depend on z and can be precomputed, such that they only need to be updated when updating $\mathbf{z}_{\text{ind}}$ or $\mathcal{D}$ itself.

A multivariate GP is established by combining n_y outputs, i.e.,

$$d(z) \sim \mathcal{N}(m_d, \Sigma_d), \tag{10}$$

where $m_d = [m_d^1, \cdots, m_d^{n_y}]^\top$, and $\Sigma_d = \text{diag}\{\Sigma_d^1, \cdots, \Sigma_d^{n_y}\}$.

IV. Sparse Kernel-based Reinforcement Learning with Online Adaptation Capability

To obtain a near-optimal yet efficient performance, we propose a sparse kernel-based RL method. We employ a quantized sample sparse technique. Under a unifying manner, the sparsity is not only applied in RL but also in GP. This technique enables efficient and rapid adaptation from both model learning and policy learning perspectives. In this section, we

will present the specific details of the sparse kernel-based RL algorithm. Firstly, we introduce the design of sparse GPs to facilitate the learning of model uncertainties. Building upon this foundation, we apply the sparse technique to design a kernel-based RL framework. Finally, we present a safety-aware module for deploying the RL policies to intelligent vehicles.

A. Sparse GP Regression for Learning Model Uncertainties

Nominal vehicle models often fail to perfectly represent the exact dynamics. Thus it is necessary to identify the differences to achieve better planning performance. To construct the "input-output" form of a GP, Eq. (1) is rewritten as

$$d(z_k) = B_d^{\dagger}(x_{k+1} - f_{\text{nom}}(x_k, u_k)), \tag{11}$$

where $B_d^{\dagger}$ is the Moore–Penrose pseudoinverse. Historic vehicle motion data is collected to capture the exact dynamics.

With the optimized parameters, GP models are utilized to compensate for uncertainties. Therefore, we define the learned model of (1) as

$$x_{k+1} = f_{\text{nom}}(x_k, u_k) + d(z_k) + \epsilon, \tag{12}$$

where $\epsilon \in \mathbb{R}^{n_x}$ is the estimation error.

Remark 1: Given a training data set with n samples, the computational complexity of the full GP is $\mathcal{O}(n^3)$, while it is $\mathcal{O}(nn_{\text{ind}}^2)$ for FITC. If we obtain a sparse dictionary $\mathcal{D}_{\text{SGP}}$ with $n_{\text{sp}} \ll n$ samples, the complexity is $\mathcal{O}(n_{\text{sp}}n_{\text{sp,ind}}^2)$, where $n_{\text{sp,ind}}$ is the inducing samples obtained from $\mathcal{D}_{\text{SGP}}$. The Jacobian matrices of (12) are

$$A_{d,k} = A_{\text{nom},k} + \frac{\partial d(z_k)}{\partial x_k}, B_{d,k} = B_{\text{nom},k} + \frac{\partial d(z_k)}{\partial u_k}, \tag{13}$$

where $A_{d,k} \in \mathbb{R}^{n_x \times n_x}$, $B_{d,k} \in \mathbb{R}^{n_x \times n_u}$, $A_{\text{nom},k} = I + T_s \partial f_{\text{nom}}^0 / \partial x$ and $B_{\text{nom},k} = T_s \partial f_{\text{nom}}^0 / \partial u$, T_s is the sampling interval and f_{nom}^0 is the nominal dynamics. Finally, the following model is used for training:

$$\mathbf{x}_{k+1} = A_{d,k}\mathbf{x}_k + B_{d,k}\mathbf{u}_k, \tag{14}$$

where the error state $\mathbf{x}$ is defined by $x - x_r$; Namely, $\mathbf{x} = [e_{v_x}, e_{v_y}, e_\varphi, e_\omega, e_X, e_Y]^\top$, and $\mathbf{u} = [a_x, \delta_f]^\top$.

B. Sparse Kernel-based RL with State Constraints

1) Cost function reformulation with barrier function: For the vehicle dynamics in Sec. A, we define a projection function $\psi : \mathbb{R}^6 \to \mathbb{R}^2$ by $(v_x, v_y, \varphi, \omega, X, Y) \mapsto (e_X, e_Y)$, mapping the current vehicle states to the position errors. The safety constraint is incorporated into the barrier function, i.e.,

$$\mathcal{B}(x_k) = \exp\left(-\|\psi(x_k)\|\right). \tag{15}$$

Thus the cost function L is re-defined as follows:

$$L(\mathbf{x}_k, \mathbf{u}_k) = \underbrace{\mathbf{x}_k^\top Q \mathbf{x}_k}_{\textit{state term}} + \underbrace{\mathbf{u}_k^\top R \mathbf{u}_k}_{\textit{control term}} + \underbrace{\mu \mathcal{B}(x_k)}_{\textit{barrier term}}, \tag{16}$$

where the penalty coefficient μ is a non-negative constant, $Q = \text{diag}\{Q_1, 0, Q_3, 0, Q_5, Q_6\} \in \mathbb{R}^{6\times 6}$ is positive semi-definite, and $R = \text{diag}\{R_1, R_2\} \in \mathbb{R}^{2\times 2}$ is positive definite.

Remark 2: Fig. 2 illustrates the designed cost function L, which is a trade-off between training accuracy and safety. The

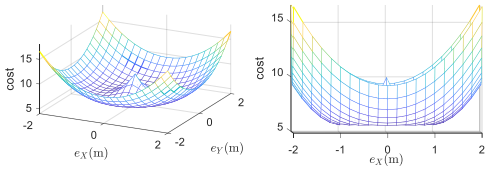

Fig. 2. An illustration of the values of L with e_X and e_Y.

state term implies that a smaller value of $\|\psi(x_k)\|$ is preferred. As obstacles on the global path would lead to safety issues, the *barrier term* comes into effect, driving the vehicle away from the *desired path*. However, as $\|\psi(x_k)\|$ increases, the cost of the *state term* becomes larger, which lowers tracking accuracy. Finally, one can obtain a policy considering both tracking accuracy and safety embedded in L.

2) Sparse kernel-based structure for approximating the optimal control and value function: ALD can also be utilized to sparsify samples and hence acquire representative ones. Sparse kernel-based functions are adopted in actor-critic networks to construct a basis function vector for approximating λ^* and $\mathbf{u}^*$ (specific derivation can be found in [10]). To this end, their estimations are expressed using the sparse kernel function, i.e.,

$$\hat{\mathbf{u}}_k = \mathcal{W}_a^\top \mathcal{K}(\mathbf{x}_k), \tag{17a}$$

$$\hat{\lambda}_k = \mathcal{W}_c^\top \mathcal{K}(\mathbf{x}_k), \tag{17b}$$

where $\mathcal{W}_a \in \mathbb{R}^{n_\mathcal{K}\times 2}$ and $\mathcal{W}_c \in \mathbb{R}^{n_\mathcal{K}\times 6}$ are the weights of the actor and critic networks, respectively, and $n_\mathcal{K}$ denotes the dimension of $\mathcal{D}_{\text{SRL}}$. The basis function of state $\mathbf{x}_k$ is

$$\mathcal{K}(\mathbf{x}_k) = [k(\mathbf{x}_k, c_1), \ldots, k(\mathbf{x}_k, c_{n_\mathcal{K}})]^\top, \tag{17c}$$

where $c_1, \cdots, c_{n_\mathcal{K}} \in \mathbb{R}^6$ are the elements in $\mathcal{D}_{\text{SRL}}$.

3) Iterative batch-mode training framework using sparse kernel feature: Here, we utilize a batch-mode RL learning strategy to train the actor-critic networks with a sample set $\mathcal{D}_{\text{RL}} = \{\mathbf{x}_k\}_{k=1}^M$. In terms of the actor network, the objective is to learn the optimal policy $\mathbf{u}^*$. At each iteration, the mean square cost to be minimized can be expressed as follows:

$$E_a = \|\hat{\mathbf{U}} - \mathbf{U}\|^2 + \rho_a \|\mathcal{W}_a\|^2, \tag{18a}$$

where $\hat{\mathbf{U}} = [\hat{\mathbf{u}}_1, \cdots, \hat{\mathbf{u}}_M]$ and $\mathbf{U} = [\mathbf{u}_1, \cdots, \mathbf{u}_M]$, M is the number of samples, ρ_a is a small positive number, and $\rho_a\|\mathcal{W}_a\|^2$ is an L_2 regularization term to limit the excessive increase of the weights during the update process. Here, $\mathbf{u}_k$ is the target control policy, which can be obtained by letting $\partial V / \partial \mathbf{u} = 0$ and defined by

$$\mathbf{u}_k = -1/2\gamma R^{-1} B_{d,k}^\top \hat{\lambda}_{k+1}. \tag{18b}$$

In the critic network, the target is to learn the optimal value function $\mathbf{\Lambda}$ of the planning task, which yields the following cost function to be minimized:

$$E_c = \|\hat{\mathbf{\Lambda}} - \mathbf{\Lambda}\|^2 + \rho_c \|\mathcal{W}_c\|^2, \tag{19a}$$

where $\hat{\mathbf{\Lambda}} = [\hat{\lambda}_1, \cdots, \hat{\lambda}_M]$ and $\mathbf{\Lambda} = [\lambda_1, \cdots, \lambda_M]$, $\rho_c\|\mathcal{W}_c\|^2$ is an L_2 regularization term, ρ_c is a small positive real number. Here λ_k the target value function defined by

$$\lambda_k = 2Q\mathbf{x}_k + \mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k} + \gamma A_{d,k}^\top \hat{\lambda}_{k+1} \tag{19b}$$

Algorithm 1: GP-SKRL algorithm

Input: Hyper-parameters $\gamma, \rho_a, \rho_c, \sigma_a, \sigma_c$, randomly initialized $\mathcal{W}_a, \mathcal{W}_c$, pre-collected datasets $\mathcal{D}_{\rm GP}, \mathcal{D}_{\rm RL}$.
Output: The converged policy π.

Update the dictionaries $\mathcal{D}_{\rm GP}$ and $\mathcal{D}_{\rm RL}$ online.
Obtain the sparse dictionaries $\mathcal{D}_{\rm SGP}$ and $\mathcal{D}_{\rm SRL}$ using ALD.
Train GP models using $\mathcal{D}_{\rm GP}$ and $\mathcal{D}_{\rm SGP}$ (inducing points).
for $\varrho \in \{0, 1\}$(*control and planning*) **do**
  $\varrho = 0 : \mu = 0$; `// Control policy`
  $\varrho = 1 : \mu \leftarrow$ a positive real number. `// Planning policy`
  while *policy has not converged* **do**
    for $k = 1 \cdots M$ (*sample number of* $\mathcal{D}_{\rm RL}$) **do**
      Update $A_{d,k}, B_{d,k}$ using GP models yields Eq. (14).
      Obtain $\hat{\lambda}_k$ and $\hat{\mathbf{u}}_k$ with (17b) and (17a).
      Construct kernel feature $\mathcal{K}(\mathbf{x}_k)$ with (17c).
    end
    for $k = 1 \cdots M$ (*sample number of* $\mathcal{D}_{\rm RL}$) **do**
      Obtain $\mathbf{x}_{k+1}$ and λ_{k+1} with (14) and (17b)..
      Compute $\mathbf{u}_k$ and λ_k with (18b) and (19b).
    end
    Update actor-critic weights with Eqs. (20) and (21).
  end
  $\pi(\varrho) = \mathcal{W}_a^*$.
end
while *the vehicle has not reached the destination* **do**
  Compute the control input $\hat{\mathbf{u}}$ with (17a) and perform steps 1-20.
  if *the planning and control policies converge* **then**
    Update the current policy π.
  end
end

Letting $\partial E_a/\partial \mathcal{W}_a$ and $\partial E_c/\partial \mathcal{W}_c$ be 0, the update rules of actor-critic networks can be obtained. Hence, at the i-th iteration, the weight update of the critic network is

$$\mathcal{W}_c = (\mathbf{K}\mathbf{K}^\top + \rho_c I)^{-1}\mathbf{K}\mathbf{\Lambda}^\top, \tag{20}$$

where $\mathbf{K} = [\mathcal{K}(\mathbf{x}_1), \cdots, \mathcal{K}(\mathbf{x}_M)]$. In the same way, the weight update of the actor network is

$$\mathcal{W}_a = (\mathbf{K}\mathbf{K}^\top + \rho_a I)^{-1}\mathbf{K}\mathbf{U}^\top. \tag{21}$$

GP-SKRL is trained for a combinatorial near-optimal planning policy based on the collected samples. After the weights of the actor-critic networks converge at the i-th iteration, i.e., $\|\mathcal{W}_{a,i} - \mathcal{W}_{a,i-1}\|^2 \leq \sigma_a$ and $\|\mathcal{W}_{c,i} - \mathcal{W}_{c,i-1}\|^2 \leq \sigma_c$, where σ_a and σ_c are small positive real numbers, the learned policies can be deployed to real-time motion planning tasks.

Remark 3: The computational complexity of GP-SKRL in training approximates to $\mathcal{O}(Mn_{\mathcal{K}}^2)$. When the trained policy is applied to real-time tasks, the computational complexity is approximately $\mathcal{O}(2N_u n_{\mathcal{K}})$, where N_u is the dimension of u.

Remark 4: Our approach is built upon kernel-based RL methods [17], [18]. On the one hand, we extend the previous kernel-based RL works to the online adaptive learning scenarios under the uncertain dynamics constraint. On the other hand, we incorporate cost-shaping into sparse kernel-based RL to realize near-optimal motion planning of intelligent vehicles.

In the following, we present the convergence analysis of the GP-SKRL algorithm.

Theorem 1: (Convergence of GP-SKRL). As $i \to \infty$, sequences $\hat{\mathbf{\Lambda}}^{[i]}$ and $\hat{\mathbf{U}}^{[i]}$ will converge to $\mathbf{\Lambda}^*$ and $\mathbf{U}^*$, respectively, i.e., as $i \to \infty$, $\hat{\mathbf{\Lambda}}^{[i]} \to \mathbf{\Lambda}^*$ and $\hat{\mathbf{U}}^{[i]} \to \mathbf{U}^*$.

Proof: See Appendix D. ■

C. Safety-Aware Module for Deploying RL Policies

The safety-aware scheme (right half of Fig. 1) consists of five modules. The first module, obstacle dilation, dilates obstacles to ensure that the physical constraints of the vehicle are not violated. The second module, reference point determination, determines the current reference point by finding the nearest point on the *desired path*, which is computed by the following strategy. When the vehicle is within the potential obstacle zone, the *desired path* is adjusted to align with the contour of the obstacle. The control policy is employed to repeatedly calculate N_p-step real-time trajectories with the global path. If the trajectories collide with obstacles, the current *desired path* is maintained; Otherwise, the *desired path* is switched to the global path. Note that the global path can be obtained by connecting the starting and end points or through path planning techniques. For clarity, we provide an illustrative diagram depicting the variations of reference points during the execution of motion planning tasks in Fig. 3.

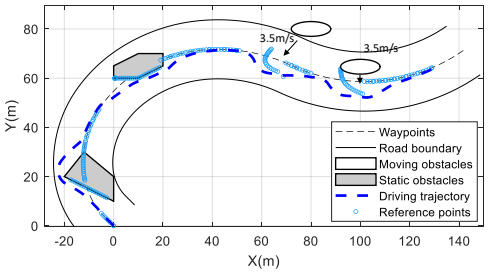

Fig. 3. An illustration of selecting reference points in a planning task.

When dealing with polygonal obstacles, the third module, reference heading angle calculation, employs a consistent strategy described in [5]. The established constraint at the i-th time instant is $A_i^\top p_i \leq b_i$, where $A_i = [a_{i,1}, a_{i,2}]^\top \in \mathbb{R}^2$ and $-a_{i,1}/a_{i,2}$ is the slope of a straight line, $b_i \in \mathbb{R}$ is the intercept, and $p_i = (X_i, Y_i)^\top \in \mathbb{R}^2$ is the position of the vehicle. The reference heading angle is obtained using the inverse tangent function: $\varphi_{r,i} = \text{atan}(-a_{i,1}/a_{i,2})$. In the fourth module, RL policy projection, a switching scheme is implemented. The planning policy, denoted as $\pi(1)$, is applied when the *desired path* is the obstacle's contour. Conversely, when the *desired path* is the global path, the policy is adjusted to $\pi(0)$. The final module, optimal control computation, allows for the computation of optimal control using Eq. (17a).

V. Performance Evaluation and Comparisons

The performance of GP-SKRL was tested through comparisons with kinodynamic RRT* (Kd-RRT*) [2], model predictive control with control barrier function (MPC-CBF) [3], local model predictive contouring control (LMPCC) [4] and convex feasible set algorithm (CFS) [5] under specific metrics. Their characteristics and differences are shown in TABLE I. Then we compared the tracking-control performance of GP-SKRL with and without model learning, abbreviated as GP-SKRL (w/ ML) and GP-SKRL (w/o ML), to validate the capability of learning uncertainties. Furthermore, real-world experiments on a Hongqi E-HS3 electric car were performed.

TABLE I
COMPARISON BETWEEN METHODS INVOLVED IN SIMULATION OR REAL-WORLD EXPERIMENTS.

Methods	Kd-RRT*	MPC-CBF	LMPCC	CFS	GP-SKRL
Vehicle dynamics	✓	✓	✓	✓	✓
Online optimization	×	✓	✓	✓	✓
Offline training	✓	×	×	×	✓
Dynamic obstacles	×	✓	✓	×	✓
Model online learning	×	×	×	×	✓

TABLE II
PERFORMANCE EVALUATIONS OF FIG. 4

Quantitative Metrics	GP-SKRL	MPC-CBF	LMPCC	CFS	Kd-RRT* [1]
J	**44.91**	105.90	60.86	47.80	463.59
Length	**243.3** m	249.3 m	246.3 m	246.0 m	261.6 m
CT	**24.3** s	25.8 s	24.8 s	24.9 s	25.62 s
Aver. S.T.	**0.005** ms	100 ms	80 ms	60 ms	174 ms

[1] : Kd-RRT* approach converges at the 236-th iteration.

A. Performance Comparisons with Other Approaches

The settings and hyperparameters of the comparative methods discussed in this section are described in Appendix C.

1) Evaluations of the computational results: To better evaluate the GP-SKRL approach, we compare it with other kinodynamic motion planning algorithms under several specific metrics, which are Aver. S.T. (average solution time of each time step), J_{Lat} (cost of the lateral error), J_{Lon} (cost of the longitudinal error), J_{Heading} (cost of the heading error), J_{Con} (control cost), and J_{Safe} (safety cost). In this regard, we use a weighted average over all costs to evaluate the performance, which is expressed as follows:

$$J = \frac{1}{N}\sum_{k=1}^{N}(\mathcal{Q}_1 J_{k,\text{Lon}} + \mathcal{Q}_2 J_{k,\text{Lat}} + \mathcal{Q}_3 J_{k,\text{Heading}} + J_{k,\text{Con}}), \quad (22)$$

where $J_{\text{Lon}} = \|e_x\|^2, J_{\text{Lat}} = \|e_y\|^2, J_{\text{Heading}} = \|e_\varphi\|^2$, the control penalty matrix is $\mathcal{R} = \text{diag}\{\mathcal{R}_1, \mathcal{R}_2\} \in \mathbb{R}^{2\times 2}$, where $\mathcal{R}_1, \mathcal{R}_2 > 0$, the control cost is $J_{\text{Con}} = \|\mathbf{u}\|^2_{\mathcal{R}}$, and N denotes the number of the overall time steps. Another quantitative metric of overall length can be computed by

$$\text{Length} = \sum_{k=1}^{N-1}\|(X_{k+1}, Y_{k+1}) - (X_k, Y_k)\|.$$

Additionally, when a planning algorithm is deployed on the vehicle, the completion time from the starting point to the goal point is denoted as CT.

2) Analyses of simulation results in the first scenario: We set the starting point as $(X, Y) = (5, 58)$ and the end point at $(X, Y) = (238, 50)$. A reference trajectory is obtained by connecting the two points with a constant reference heading angle of -0.0343 rad. In this scenario, we set the maximum desired speed V_{max} as 10 m/s for the testing approaches, and vehicles are required to avoid static polygonal obstacles.

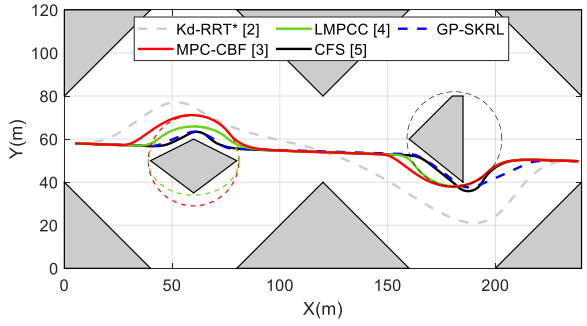

Fig. 4. Obstacle avoidance in an unstructured road: the grey regions are static obstacles that are physically inaccessible.

In Fig. 4, the red dashed circle and green dashed ellipse are used to enclose the first static obstacle to establish the MPC-CBF and LMPCC constraints, respectively. The black dash circle is used to enclose the second static obstacle to establish the MPC-CBF and LMPCC constraints. According to the testing results in our implementation, decreasing $\mathcal{Q}_1$ and $\mathcal{Q}_3$ while increasing $\mathcal{Q}_2$ enables better performance. Finally, state penalty values in (22) are set as $\mathcal{Q}_1 = \mathcal{Q}_2 = 2$ and $\mathcal{Q}_3 = 5$. The control penalty matrix $\mathcal{R}$ is $\text{diag}\{3, 3\}$. The comparison results are listed in TABLE II, which show that GP-SKRL makes the vehicle move along the obstacle boundary and achieves a better performance than LMPCC, MPC-CBF, and CFS in terms of computation time, cost, and trajectory length. Under the constraint of V_{max}, GP-SKRL completes the task with the shortest time among the approaches. It should be noted that GP-SKRL is more computationally efficient among all compared approaches due to the scheme of batch RL.

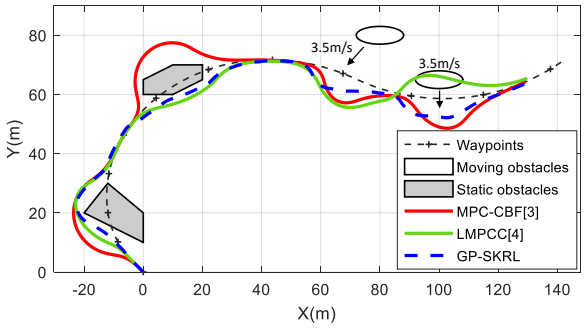

Fig. 5. Planning results in Scenario II with static and moving obstacles. The starting point coordinate is set as $(0, 0)$. The moving obstacles (dark ellipses) are at their initial positions and the black arrows are their moving directions.

TABLE III
PERFORMANCE EVALUATIONS OF FIG. 5

Quantitative Metrics	GP-SKRL	MPC-CBF	LMPCC	CFS
J	**33.64**	94.28	54.62	-[1]
Length	**215.12** m	241.53 m	217.41 m	-
CT	**21.50** s	24.50 s	22.0 s	-
Aver. S.T.	**0.005** ms	90 ms	100 ms	-

[1] CFS with the kinodynamic constraint fails to avoid moving obstacles.

3) Analyses of simulation results in the second scenario: To implement the comparison methods, we enclose all obstacles separately with circles and ellipses, and the black line with cross marks is the reference trajectory (see Fig. 5). In this scenario, the maximum desired speed is set as 10 m/s for the testing approaches. As shown in Fig. 5, MPC-CBF, LMPCC, and GP-SKRL successfully enable the vehicle to avoid obstacles and arrive at the destination. In the following, we analyze the specific metrics of them in completing this task.

As shown in TABLE III, GP-SKRL is more advantageous than MPC-CBF and LMPCC regarding computation time, average cost and route length. Regarding cost, MPC-CBF and LMPCC use circles or ellipses to enclose all obstacles, which results in too many extra safe areas being included. Likewise, GP-SKRL can obtain a shorter route among the comparison approaches. And it takes a shorter computational time.

4) Validation of the model learning scheme: To improve planning and control performance, GP-SKRL trains GP models to learn model uncertainties. The parameters of the nominal model are set as $m = 20,000$ kg and $I_z = 20,000$ kg·m, while $m = 2257$ kg and $I_z = 3524.9$ kg·m are set for the exact model. Hyperparameters are set as $\sigma_{f,a} = 5$ and $\ell = 2$. Due to the structure of the nominal model, parameter uncertainties are assumed to only affect v_y and ω of the system; that is, $g(x,u) = g\left(v_x, v_y, \omega, a_x, \delta_f\right) : \mathbb{R}^5 \to \mathbb{R}^2$. The i.i.d. process noise $w_k \in \mathbb{R}^6$ is added to the vehicle dynamics, where $\Sigma^w = 0.001 \cdot \text{diag}\{1/3, \cdots, 1/3\}$. The proposed sparse GP is employed to identify model uncertainties on all six states.

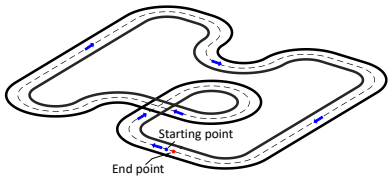

Fig. 6. Racing road with a desired speed of 10 m/s and maximum length of 508 m. The black dot curve is the reference path and the black solid lines are the road boundaries. We use blue arrows to represent the driving directions.

After collecting the training samples, the posterior hyperparameters were inferred from the marginal log-likelihood optimization. Fig. 6 depicts the racing road for testing the tracking-control performance of GP-SKRL (w/ ML) and GP-SKRL (w/o ML). We counted the lateral stage errors and the average lateral errors of their two driving trajectories to present the performance improvement clearly. In Fig. 7, using the ML strategy reduces the lateral tracking control error, demonstrating that the ML strategy improves the planning and control performance of GP-SKRL under model uncertainties.

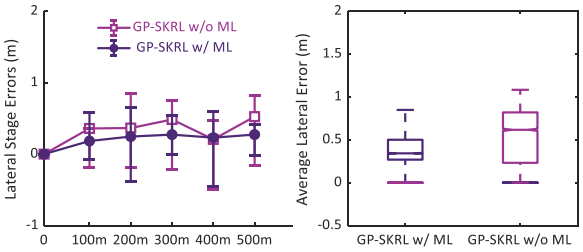

Fig. 7. Tracking error comparison in the racetrack road of Fig. 6 between GP-SKRL (w/o ML) and GP-SKRL (w/ ML).

To validate the effectiveness of ALD-GP (sparse GP with ALD), we conducted the following test. For a fixed number of data points (NDPs), training and testing were performed using ALD-GP and FITC-GP (sparse GP with FITC) methods separately. The resulting training time and the one-step average prediction error (APE) were recorded and are presented in TABLE IV. In summary, the ALD-GP method shows slightly higher average errors in v_y and ω, but the training time is significantly lower compared with FITC-GP when the value of NDPs is high. Therefore, ALD-GP significantly improves computational efficiency, sacrificing only little accuracy.

TABLE IV
COMPARISON BETWEEN TWO SPARSE GP METHODS.

NDPs	**training time** (s)		**APE in** v_y (10^{-3} m/s)		**APE in** ω (10^{-3}rad/s)	
	ALD-GP	FITC-GP	ALD-GP	FITC-GP	ALD-GP	FITC-GP
9000	0.10	43.14	1.36	0.29	1.05	0.30
6000	0.05	14.65	1.37	0.31	1.05	0.31
3000	0.04	2.71	1.38	0.36	1.05	0.35
1000	0.03	0.31	2.86	1.76	2.13	1.56

5) Validation of the Online Adaption Capability: To highlight the online adaption capability of the algorithm, abbreviated as GP-SKRL w/ OA, we deployed the RL policy trained with the nominal model (abbreviated as GP-SKRL w/o OA) in a tracking control task. The exact vehicle parameters were set as different values in three stages: 0-170 m ($m = 2257$ kg, $I_z = 3524.9$ kg· m), 170-340 m ($m = 1957$ kg, $I_z = 3224.9$ kg·m), and 340-508 m ($m = 1657$ kg, $I_z = 2924.9$ kg·m). The first half stages, i.e., 0-85 m, 170-255 m, and 340-425 m, adopted the same policy as GP-SKRL w/o OA to collect training data. Online policy updates were performed at positions 85 m, 255 m, and 425 m, with an average time of 0.99 seconds. From the simulation results in Fig. 8, the tracking control error is reduced in the second half stages, 85-170 m, 255-340 m, and 425-508 m. The simulation results validate the online adaption capability of our algorithm.

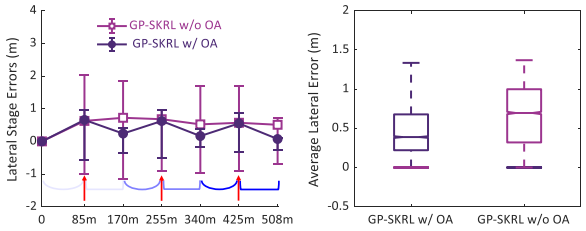

Fig. 8. Tracking error comparison in the racetrack road of Fig. 6 between GP-SKRL (w/o OA) and GP-SKRL (w/ OA). The red arrows indicate the positions where RL policies were updated.

B. Real-World Experiments

The experimental platform, shown in the central image of Fig. 1, is a real Hongqi E-HS3 electric car equipped with a positioning system and perception sensors, including millimeter-wave radar, LiDAR, and cameras. The planning program was run on a laptop in the Windows operating system with an Intel i7-11800H CPU @2.30GHz, and the computed control sequence was transferred to the low-level control module using the Robot Operating System (ROS) [30].

1) Case 1: Experimental comparisons with optimization-based approaches: We further compared GP-SKRL with several optimization-based approaches in an unstructured environment. With an initial velocity of 0 m/s and a desired velocity of 2 m/s, the ego vehicle has to avoid the moving obstacle

to remain safe. For comparison, the prediction horizons of LMPCC and MPC-CBF were both set to 20. To improve the computational efficiency, a successive linearization strategy was adopted in our implementation, and the OSQP [31] solver was used. Consistent with the error definitions of the lateral and longitudinal directions in [4], the cost function was set to $\mathcal{Q}_1 e_x^2 + \mathcal{Q}_2 e_y^2 + \mathcal{Q}_3 e_\varphi^2 + \mathcal{R}_1^2 \delta_f + \mathcal{R}_2^2 a_x + P_{\text{colli}}$, where the state penalty matrix $[\mathcal{Q}_1, \mathcal{Q}_2, \mathcal{Q}_3]$ was set to $[10, 5, 500]$, the control matrix $\mathcal{R}$ was set to $\text{diag}\{50, 360\}$. If a collision with obstacles occurs, then $P_{\text{colli}} = 100$; otherwise, $P_{\text{colli}} = 0$.

Two static obstacles are placed on the reference trajectory while a moving obstacle is crossing the road.

Fig. 9. Views from inside and outside the car during the experiment. For simplicity, we ignore the presence of lanes and it is assumed safe for the vehicle to drive into the opposite lane in this experiment.

The vehicle travels from the starting point $(0, 0)$ to the end point $(150, 0)$. It must avoid two static obstacles and a moving pedestrian to maintain safe driving. As seen in Fig. 10, the final driving trajectories using the LMPCC, GP-SKRL and MPC-CBF planners are depicted with the green, blue, and red gradient-colour lines, respectively. LMPCC can successfully avoid the first two static obstacles, but it does not take any actions due to the failure to find a feasible solution. MPC-CBF successfully avoids the two grey static obstacles. As the pedestrian crosses the road, the ego vehicle must engage in emergency obstacle-avoidance behaviour, but the vehicle fails to avoid collision in time, causing a solution failure of the optimization problem, which is mainly attributed to the heavy computational load. Notably, GP-SKRL successfully completes the tasks in real-time.

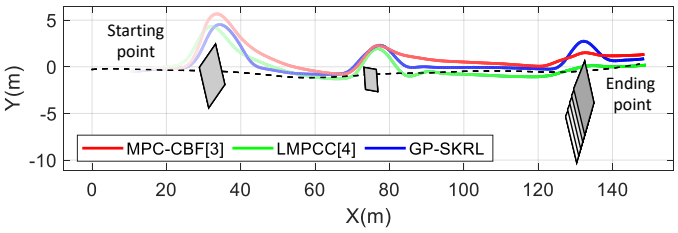

Fig. 10. Experimental results of three approaches for obstacle avoidance tasks. The polygons with black contours are obstacles blocking the road. The moving pedestrian is represented by multiple grey polygons. All approaches are required to have a desired speed of 2 m/s.

2) Case 2: Moving obstacle avoidance: In this case, a moving polygonal obstacle moves along the reference trajectory. With an initial velocity of 0 m/s and a desired velocity of 5 m/s, the ego vehicle has to avoid the moving obstacle to remain safe. As seen in Fig. 11, the ego vehicle (blue boxes) starts from the coordinate origin and tracks the reference trajectory. Before crashing into a blocking moving obstacle (grey polygons), the ego vehicle proactively avoids it with a lateral displacement and finally tracks the reference trajectory.

TABLE V
PERFORMANCE EVALUATIONS OF FIG. 10

Quantitative Metrics	GP-SKRL	MPC-CBF*	LMPCC*
J	**62.18**	185.77	161.58
Aver. S.T.	**0.005** ms	90 ms	100 ms

* MPC-CBF and LMPCC with the kinodynamic constraint fail to solve the planning problem in time.

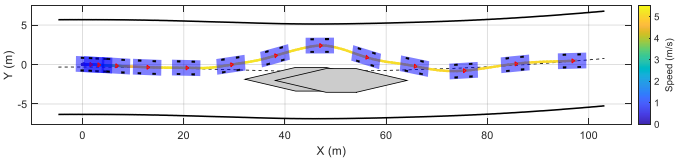

Fig. 11. Moving obstacle avoidance where the black dashed line is the reference trajectory, and the thick black line is the road boundary.

C. Analysis and Discussion

As mentioned in Sec. IV-B3, the computational complexity of GP-SKRL is $\mathcal{O}(Mn_{\mathcal{K}}^2)$. The complexity increases exponentially as the values of M and $n_{\mathcal{K}}$ increase. For methods without using the quantized sparse technique, the complexity is $\mathcal{O}(M^3)$, which is much higher. In our implementation, GP-SKRL converges after about 60 training iterations with 30000 samples. And each iteration only takes around 0.015s.

From the comparison results, several factors contribute to the superiority of GP-SKRL: 1) The proposed sparse GP can generate representative samples using a quantized sparse technique, thus outperforming the FITC method in terms of computational efficiency; 2) We propose an efficient sparse kernel-based RL method for learning optimal motion planning policies, enabling online adaption with near-optimal performance; 3) By incorporating a safety-aware module, GP-SKRL is able to assess the feasibility of generated local trajectories, and therefore, it enables a vehicle to return to the reference trajectory as soon as possible. GP-SKRL ultimately achieves performance surpassing that of several advanced methods.

VI. CONCLUSION

This paper proposed a learning-based near-optimal motion planning approach with online adaption capability for intelligent vehicles with uncertain dynamics. The sparse kernel-based technique is manifested not only in the basis functions of RL but also in GPs. Therefore, the RL policies can be efficiently learned to guide the vehicle to maintain a safe distance from obstacles. We demonstrated the effectiveness of GP-SKRL in completing motion planning tasks via simulation and real-world experimental results. In particular, we demonstrated that the performance in a racing scenario can be significantly improved by using the sparse GP technique.

APPENDIX

A. Vehicle Dynamics

The vehicle dynamics are described using a "bicycle" model in [32], and the nominal dynamics are expressed as

$$\begin{bmatrix} \dot{v}_x \\ \dot{v}_y \\ \dot{\varphi} \\ \dot{\omega} \\ \dot{X} \\ \dot{Y} \end{bmatrix} = \underbrace{\begin{bmatrix} v_y\omega + a_x \\ 2C_{af}(\frac{\delta_f}{m} - \frac{v_y + l_f\omega}{mv_x}) + 2C_{ar}\frac{l_r\omega - v_y}{mv_x} - v_x\omega \\ \omega \\ \frac{2}{I_z}\left[l_f C_{af}(\delta_f - \frac{v_y + l_f\omega}{v_x}) - l_r C_{ar}\frac{l_r\omega - v_y}{v_x}\right] \\ v_x\cos\varphi - v_y\sin\varphi \\ v_x\sin\varphi + v_y\cos\varphi \end{bmatrix}}_{f^0_{\text{nom}}(x,u)},$$

where $x = (v_x, v_y, \varphi, \omega, X, Y) \in \mathbb{R}^6$ is the state vector, v_x, v_y denote the longitudinal and lateral velocities, respectively, φ is the yaw angle, ω denotes the yaw rate, X, Y are the global horizontal and vertical coordinates of the vehicle, respectively, l_f, l_r are the distances from the center of gravity (CoG) to the front and rear wheels, respectively, C_{af}, C_{ar} represent the cornering stiffnesses of the front and rear wheels, respectively, and I_z denotes the yaw moment of inertia. In this model, acceleration a_x and steering angle δ_f are two variables of the control vector u, i.e., $u = [a_x, \delta_f]^\top$. The reference state is denoted by $x_r = (v_x^r, v_y^r, \varphi_r, \omega_r, X_r, Y_r)$. Fig. 12 displays the relationship between the driving vehicle and the projection point $x_r = (v_x^r, v_y^r, \varphi_r, \omega_r, X_r, Y_r)$. To construct an error model for facilitating subsequent algorithm design, we subtract the current state from the desired state, i.e, $\mathbf{x} = x - x_r$ and the system control from desired control, i.e., $\mathbf{u} = u - u_r = [a_x, \delta_f]^\top$, where $u_r = [0, 0]$.

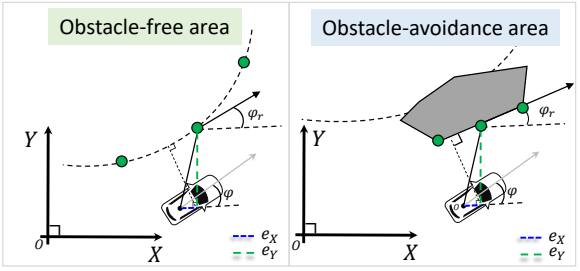

Fig. 12. The relationship between the vehicle and the reference points x_r. The green circles are the reference points, while the distance errors in the X and Y directions are blue and green dashed lines, respectively.

B. Parameter Settings of the Training Process

The state boundaries in sampling and training are set as follows: $e_{v_x} \in [-6, 6]$ m/s, $e_{v_y} \in [-6, 6]$ m/s, $e_\varphi \in [-\frac{\pi}{3}, \frac{\pi}{3}]$ rad, $e_\omega \in [-6, 6]$ rad/s, $e_X \in [-3, 3]$ m, $e_Y \in [-3, 3]$ m. The control boundaries are set as follows: $a_x \in [-1, 1]$ m/s^2 and $\delta_f \in [-\pi/6, \pi/6]$ rad. The real vehicle parameters for the simulation are set as follows: $M = 2257$ kg, $l_f = 1.33$ m, $l_r = 1.81$ m, $C_{af} = 60790$ N/rad, $C_{ar} = 50400$ N/rad, $g = 9.8$ m/s^2, $I_z = 3524.9$ kg·m. Gaussian kernel function is adopted, i.e., $k(s_i, s_j) = \exp(-\|s_i - s_j\|^2/\tau^2)$. The kernel width τ used to construct the dictionary is 0.9, and a vector $\tau = [0.7, 0.8, 0.9, 1]$ is adopted to construct the multikernel feature. The penalty coefficient μ is set to 6 for an obstacle-avoidance policy and $\mu = 0$ for a control policy.

C. Simulation Settings of Comparison Approaches

Comparison approaches, including MPC-CBF, LMPCC, and CFS, use the IPOPT [33] solver. They all consider the nonlinear vehicle dynamics (A) to compare the performance. Kd-RRT* approach solves the planning problem through iterative optimization. For CFS, we used the obstacle-avoidance idea in [5] to establish the obstacle constraints and implemented a kinodynamic motion planning approach; see Sec. IV-E. For MPC-CBF, the extended class $\mathcal{K}_\infty$ function γ is set to 0.4 in the first scenario and 0.6 in the second one. For LMPCC, the potential function is included in the cost, i.e.,

$$J_{\text{repulsive}} = Q_R \sum_{k=1}^{N_{\text{obs}}} \left(\frac{1}{(\Delta x_k)^2 + (\Delta y_k)^2 + \epsilon}\right),$$

where the coefficient Q_R is set to 5000 in the first scenario and 30000 in the second scenario, N_{obs} is the number of moving obstacles, $\Delta x_k, \Delta y_k$ represents the distance from the vehicle to the center of the k-th moving obstacle, and the parameter ϵ is set to 0.001. In Scenario II, the first elliptical obstacle moves if $X > 30$, while the second obstacle moves if $X > 80$. The simulation was performed on a laptop running MATLAB 2020a with an i7-11800H CPU @2.30GHz without parallel acceleration.

D. Proof of Theorem 1

(Proof of Theorem 1): According to Eq. (19b), the costate variable at the i-th iteration can be computed by

$$\lambda_k^{[i]} = 2Q\mathbf{x}_k + \gamma A_{d,k}^\top \frac{\partial V_i(\mathbf{x}_{k+1})}{\partial \mathbf{x}_{k+1}} + \mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k},$$

where $\mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k}$ is bounded. As $i \to \infty$, the costate variable λ_∞ can be written as

$$\lambda_\infty(\mathbf{x}_k) = 2Q\mathbf{x}_k + \gamma A_{d,k}^\top \frac{\partial V_\infty(\mathbf{x}_{k+1})}{\partial \mathbf{x}_{k+1}} + \mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k}.$$

Using Lemma in [34], as $i \to \infty$, $V_\infty(\mathbf{x}_k) \to V^*(\mathbf{x}_k)$. The costate $\lambda_\infty(\mathbf{x}_k)$ can be rewritten as

$$\begin{aligned} \lambda_\infty(\mathbf{x}_k) = &\ 2Q\mathbf{x}_k + \gamma A_{d,k}^\top \frac{\partial V^*(\mathbf{x}_{k+1})}{\partial \mathbf{x}_{k+1}} + \mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k} \\ = &\ 2Q\mathbf{x}_k + \gamma A_{d,k}^\top \lambda^*(\mathbf{x}_{k+1}) + \mu \frac{\partial \mathcal{B}(x_k)}{\partial \mathbf{x}_k} \quad = \lambda^*(\mathbf{x}_k). \end{aligned}$$

That is, $\lambda_k^{[i]}$ will converge to λ_k^*, as $i \to \infty$. After we substitute the optimal costate λ^* into (4), it follows that $\mathbf{u}^{[i]} \to \mathbf{u}^*$ as $i \to \infty$. As the number of iterations goes to infinity, $\hat{\mathbf{\Lambda}}^{[i]} \to \mathbf{\Lambda}^*$ and $\hat{\mathbf{U}}^{[i]} \to \mathbf{U}^*$. ■

REFERENCES

[1] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The International Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.

[2] D. J. Webb and J. Van Den Berg, "Kinodynamic RRT*: Asymptotically optimal motion planning for robots with linear dynamics," in *IEEE International Conference on Robotics and Automation*, 2013, pp. 5054–5061.

[3] J. Zeng, B. Zhang, and K. Sreenath, "Safety-critical model predictive control with discrete-time control barrier function," in *American Control Conference (ACC)*, 2021, pp. 3882–3889.

[4] B. Brito, B. Floor, L. Ferranti, and J. Alonso-Mora, "Model predictive contouring control for collision avoidance in unstructured dynamic environments," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4459–4466, 2019.

[5] C. Liu, C.-Y. Lin, and M. Tomizuka, "The convex feasible set algorithm for real time optimization in motion planning," *SIAM Journal on Control and Optimization*, vol. 56, no. 4, pp. 2712–2733, 2018.

[6] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Transactions on Control Systems Technology*, vol. 28, no. 6, pp. 2736–2743, 2019.

[7] L. Hewing, A. Liniger, and M. N. Zeilinger, "Cautious NMPC with gaussian process dynamics for autonomous miniature race cars," in *European Control Conference (ECC)*, 2018, pp. 1341–1348.

[8] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 3363–3370, 2019.

[9] C. Lian and X. Xu, "Motion planning of wheeled mobile robots based on heuristic dynamic programming," in *World Congress on Intelligent Control and Automation*. IEEE, 2014, pp. 576–580.

[10] D. Liu and Q. Wei, "Policy iteration adaptive dynamic programming algorithm for discrete-time nonlinear systems," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 3, pp. 621–634, 2013.

[11] S. Kamthe and M. Deisenroth, "Data-efficient reinforcement learning with probabilistic model predictive control," in *International conference on artificial intelligence and statistics*. PMLR, 2018, pp. 1701–1710.

[12] C. He, Y. Wan, Y. Gu, and F. L. Lewis, "Integral reinforcement learning-based approximate minimum time-energy path planning in an unknown environment," *International Journal of Robust and Nonlinear Control*, vol. 31, no. 6, pp. 1905–1922, 2021.

[13] X. Zhang, Y. Jiang, Y. Lu, and X. Xu, "Receding-horizon reinforcement learning approach for kinodynamic motion planning of autonomous vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 7, no. 3, pp. 556–568, 2022.

[14] S. N. Mahmud, K. Hareland, S. A. Nivison, Z. I. Bell, and R. Kamalapurkar, "A safety aware model-based reinforcement learning framework for systems with uncertainties," in *2021 American Control Conference (ACC)*. IEEE, 2021, pp. 1979–1984.

[15] M. Hoerger, H. Kurniawati, and A. Elfes, "Non-linearity measure for POMDP-based motion planning," *arXiv preprint arXiv:2005.14406*, 2020.

[16] W. Bejjani, R. Papallas, M. Leonetti, and M. R. Dogar, "Planning with a receding horizon for manipulation in clutter using a learned value function," in *2018 IEEE-RAS 18th International Conference on Humanoid Robots (Humanoids)*. IEEE, 2018, pp. 1–9.

[17] X. Xu, C. Lian, L. Zuo, and H. He, "Kernel-based approximate dynamic programming for real-time online learning control: An experimental study," *IEEE Transactions on Control Systems Technology*, vol. 22, no. 1, pp. 146–156, 2013.

[18] J. Liu, Z. Huang, X. Xu, X. Zhang, S. Sun, and D. Li, "Multi-kernel online reinforcement learning for path tracking control of intelligent vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, vol. 51, no. 11, pp. 6962–6975, 2021.

[19] X. Xu, D. Hu, and X. Lu, "Kernel-based least squares policy iteration for reinforcement learning," *IEEE Transactions on Neural Networks*, vol. 18, no. 4, pp. 973–992, 2007.

[20] Y. Engel, S. Mannor, and R. Meir, "The kernel recursive least-squares algorithm," *IEEE Transactions on Signal Processing*, vol. 52, no. 8, pp. 2275–2285, 2004.

[21] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[22] M. Deisenroth and C. E. Rasmussen, "PILCO: A model-based and data-efficient approach to policy search," in *Proceedings of the 28th International Conference on machine learning (ICML-11)*, 2011, pp. 465–472.

[23] C. Xie, S. Patil, T. Moldovan, S. Levine, and P. Abbeel, "Model-based reinforcement learning with parametrized physical models and optimism-driven exploration," in *2016 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2016, pp. 504–511.

[24] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine, "Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 7559–7566.

[25] Q. Zhou, H. Li, and J. Wang, "Deep model-based reinforcement learning via estimated uncertainty and conservative policy optimization," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 04, 2020, pp. 6941–6948.

[26] C. E. Luis, A. G. Bottero, J. Vinogradska, F. Berkenkamp, and J. Peters, "Model-based uncertainty in value functions," in *International Conference on Artificial Intelligence and Statistics*. PMLR, 2023, pp. 8029–8052.

[27] S. Bechtle, Y. Lin, A. Rai, L. Righetti, and F. Meier, "Curious ilqr: Resolving uncertainty in model-based RL," in *Conference on Robot Learning*. PMLR, 2020, pp. 162–171.

[28] T. Phienthrakul and B. Kijsirikul, "Evolutionary strategies for multi-scale radial basis function kernels in support vector machines," in *Proceedings of the 7th annual conference on Genetic and evolutionary computation*, 2005, pp. 905–911.

[29] E. Snelson and Z. Ghahramani, "Sparse gaussian processes using pseudo-inputs," *Advances in neural information processing systems*, vol. 18, 2005.

[30] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, A. Y. Ng *et al.*, "ROS: an open-source robot operating system," *ICRA workshop on open source software*, vol. 3(2), p. 5, 2009.

[31] B. Stellato, G. Banjac, P. Goulart, A. Bemporad, and S. Boyd, "OSQP: An operator splitting solver for quadratic programs," *Mathematical Programming Computation*, vol. 12, no. 4, pp. 637–672, 2020.

[32] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[33] 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.

[34] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Discrete-time nonlinear HJB solution using approximate dynamic programming: Convergence proof," *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 38, no. 4, pp. 943–949, 2008.