Hindawi
Journal of Advanced Transportation
Volume 2020, Article ID 8495264, 11 pages
https://doi.org/10.1155/2020/8495264

Research Article

Learn to Make Decision with Small Data for Autonomous Driving: Deep Gaussian Process and Feedback Control

Wenqi Fang, Shitian Zhang, Hui Huang, Shaobo Dang, Zhejun Huang, Wenfei Li, Zheng Wang, Tianfu Sun, and Huiyun Li

Shenzhen Institutes of Advanced Technology, Chinese Academy of Sciences, SIAT Branch, Shenzhen Institute of Artificial Intelligence and Robotics for Society, Shenzhen 518172, China

Correspondence should be addressed to Huiyun Li; hy.li@siat.ac.cn

Received 17 February 2020; Revised 21 June 2020; Accepted 20 July 2020; Published 28 August 2020

Academic Editor: Yuchuan Du

Autonomous driving is a popular and promising field in artificial intelligence. Rapid decision of the next action according to the latest few actions and status, such as acceleration, brake, and steering angle, is a major concern for autonomous driving. There are some learning methods, such as reinforcement learning which automatically learns the decision. However, it usually requires large volume of samples. In this paper, to reduce the sample size, we exploit the deep Gaussian process, where a regression model is trained on small sample datasets and captures the most significant features correctly. Besides, to realize the real-time and close-loop control, we combine the feedback control into the process. Experimental results on the Torcs simulation engine illustrate smooth driving on virtual road which can be achieved. Compared with the amount of training data in deep reinforcement learning, our method uses only 0.34% of its size and obtains similar simulation results. It may be useful for real road tests in the future.

1. Introduction

Autonomous driving is one of the most promising field of artificial intelligence [1, 2]. To realize safety driving on the real road, ego-vehicles need to recognize and track the objects with its perceptual equipment [3], as well as act properly according to the current road conditions with decision-making modules. The decision-making module is the most important part for self-driving, yet the most challenging part to achieve. The core mission mainly includes obstacle avoidance, trajectory planning, and action prediction [4, 5]. The decision-making model is built with rule-based [6] or statistical method [7], which are two popular schemes. Rule-based method can implement functionality quickly, but they are confined by the incomplete sets of state and the inability of capturing uncertainty. These shortcomings are overcome by a combination with statistical methods. In addition, with the advent of simulation engines such as Torcs [8] and Carla [9], various methods based on reinforcement learning [10] are proposed in the decision-making research and satisfactory performances are achieved. In the typical models of reinforcement learning, the agent begins, without prior knowledge about the world in advance, only with knowledge of which actions are possible, and it is expected to learn the skill solely by interacting with the environment and receiving rewards after taking actions. With this shortcoming, it requires absurdly huge amounts of time and datasets to learn to do specific tasks, such as board game [11] or self-driving [10]. However, due to insufficient diversity, real-world datasets are often unsatisfactory. And setting up a dataset with precise labels from the real world is labor-intensive and time-consuming, especially in large-scale complex transportation systems [12], not to mention building a dataset with specific features that meets our needs. To address this problem, building diversity of the virtual datasets is a practical way to improve the performance of the trained models in the case of insufficient training data [12, 13]. And in our paper, we propose another feasible way to solve decision-making problem in autonomous driving, which

directly utilizes small datasets without the aid of visual samples.

In recent years, Gaussian process (GP) has become one of prevailing regression techniques [14]. To be precise, a GP is a distribution over functions such that any finite set of function values have a joint Gaussian distribution. The predicted mean function and covariance function are used for regression and uncertainty estimation, respectively. The strength of GP regression lies in avoiding overfitting while still being able to find functions complex enough to describe any observed behavior, even in unstructured or noisy data. GP is commonly used in the situation when observations are expensive or rare to produce and methods such as deep neural network performs poorly. And it has been applied among a wide range from engineering [15], optimization [16], robotics [17], and physics [18] to biology [19]. Nevertheless, the sort of phenomena that can be easily expressed by using GP directly are limited. For example, in a sparse data scenario, the constructed probability distribution is often far away from the true posterior distribution. Recognizing this problem, many interesting research activities have been carried out, which attempt to represent new properties via the hierarchical cascading of Gaussians. Inspired by the widespread success of deep neural network architectures, Damianou and Lawrence proposed a method that a GP was directly composed with another GP; furthermore, the idea was implemented recursively, leading to the so-called deep Gaussian process (deep GP) [20]. A deep GP consists of a cascade of hidden layers of latent variables where each node acts as output for the layer above and as input for the layer below. GPs govern the mappings between the layers with their own kernel. Therefore, deep GP retains valuable properties of GP, such as well-calibrated predictive uncertainty estimation and nonparametric modeling power. In addition, it employs a hierarchical structure of GP mappings which makes it more flexible, has a greater capacity to generalize, and provides better predictive performance. This model is fascinating because it can potentially discover layers of increasingly abstract data representations, while handling and propagating uncertainty in the hierarchy at the same time [21].

Undoubtedly, according to the nature of Bayesian statistics, the deep GP model makes prediction based on statistical average. However, we cannot ensure that the statistical average results are reasonable with such small training data because of model uncertainty. And it turns out that it is not enough to solve decision-making problem in our setting according to our calculation. So, we introduce the feedback control method to compensate this shortcoming.

Based on the analysis above, in this paper, we propose a decision-making framework combining deep GP and feedback control method. The deep GP model makes action prediction possible, and feedback control method assists action uncertainty error correction. When the state is fed into the framework, the action will be obtained immediately. It is kind of an end-to-end learning method [22]. This method is trained with small data in Torcs and tested in Torcs. According to our calculation, in terms of time consumption and data volume, our method is superior to the deep reinforcement learning trained by deterministic policy gradient (DPG) method [23].

2. Related Work

2.1. Deep Reinforcement Learning. As mentioned above, self-driving vehicle is a decision-making system that processes information from various sources, such as cameras, radars, LiDARs, GPS units, and inertial sensors. This information is used by the vehicle's system to make driving decisions. The architecture can be implemented either as a sequential perception-planing-action pipeline, or as an end-to-end system. Recent works are mainly focused on deep reinforcement learning paradigm to achieve self-driving. Existing reinforcement learning algorithms mainly compose of value-based and policy-based methods. Vanilla Q-learning is the first proposed method and then becomes one of the popular value-based methods. Karavolos applies vanilla Q-learning algorithm to simulator Torcs and evaluates the effectiveness of using heuristic during the exploration [24]. Recently, lots of variants of Q-learning algorithm, such as DQN [25], Double DQN, and Dueling DQN [26], have been successfully applied to a variety of games and outperform humans since the resurgence of deep neural networks.

Different from value-based methods, policy-based methods learn the policy directly. In other words, policy-based methods output action given the current state. Silver et al. [27] propose a DPG algorithm to handle continuous action spaces efficiently without losing adequate exploration. By combining idea from DQN and actor-critic, Lillicrap et al. [23] then propose a deep DPG (DDPG) model-free approach and achieve end-to-end policy learning. In 2016, a new technique, which combines policy gradient and off-policy Q-learning (PGQL), is proposed and achieves performance exceeding that of both asynchronous advantage actor-critic and Q-learning on the full suite of Atari games [28]. All these policy-gradient methods can naturally handle the continuous action spaces. Despite validity and practicability of reinforcement learning, the training time costs too much and the volume of training data is its soft spot if we cannot get enough data.

2.2. Feedback Control Method. In addition, traditional control methods also play an important role for solving self-driving problem. The automatic control is almost the last part in the sequence of the autonomous vehicle, and one of the most critical tasks since it is responsible for its movement. The well-known controller mainly includes PID (proportional integral derivative) controller and MPC (model predictive control) controller [29]. A PID controller is a practical part used in industrial control applications to regulate pressure, speed, temperature, and other core variables [30]. The PID controller uses a control loop feedback mechanism to control process variables, and it is the most accurate and stable controller. It is so named because its output is the summation of three terms (proportional,

integral, and derivative term). Each of these terms depends on the error value between the input and the output.

Differently, the MPC controller relies on dynamic models of the process, the most common being the linear empirical models obtained through system identification. The main advantage of MPC is that it can optimize the current time step, while also taking future time steps into account. This is achieved by optimizing a finite time-horizon, but only implementing the current time slot and then optimizing again, repeatedly.

2.3. Gaussian Process. GP is a Bayesian nonparametric machine learning framework for regression, classification, and unsupervised learning [14]. A GP is a collection of random variables f, any finite combination of which satisfies a multivariate normal distribution. Suppose that a set of noisy observed outputs $\mathbf{y} \triangleq \{y_n\}_{n=1}^N$: $y_n = f(x_n) + \varepsilon$ (ϵ is an independent identically distributed (i.i.d) Gaussian noise with variance ν^2) are available for training inputs set $\mathbf{x} \triangleq \{x_n\}_{n=1}^N$. Then, the latent set $\mathbf{f} \triangleq \{f(x_n)\}_{n=1}^N$ is assumed to be a Gaussian prior $p(\mathbf{f}) \triangleq \mathcal{N}(\mathbf{f} \mid 0, \mathbf{K_{xx}})$, where $\mathbf{K_{xx}}$ is a covariance matrix with components $k(x_n, x_{n\prime})$ for $n, n\prime = 1, \ldots, N$. Since the data likelihood can be written as $p(\mathbf{y} \mid \mathbf{f}) \triangleq \mathcal{N}(\mathbf{y} \mid \mathbf{f}, \nu^2\mathbf{I})$, the GP predictive distribution of the latent outputs $\mathbf{f}^* \triangleq \{f(\mathbf{x}^*)\}$ with any test inputs $\mathbf{x}^*$ can be computed in a closed form by integrating $p(\mathbf{f}^* \mid \mathbf{y}) \triangleq \int p(\mathbf{f}^* \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{y}) \mathrm{d}\mathbf{f}$, where $p(\mathbf{f} \mid \mathbf{y})$ is the posterior distribution. Due to the inversion of the covariance matrix, the training GP model needs $O(N^3)$ operations, which prevents it from scaling well to massive datasets. To improve its scalability, the sparse GP (SGP) models exploit a set $\mathbf{u} \triangleq \{u_m\}_{m=1}^M$ of inducing output variables for some small set $\mathbf{z} \triangleq \{z_m\}_{m=1}^M$ of inducing inputs (i.e., $M \ll N$). Then, the joint probability of $\mathbf{y}$, $\mathbf{f}$, and $\mathbf{u}$ is as follows:

$$p(\mathbf{y}, \mathbf{f}, \mathbf{u}) = p(\mathbf{y} \mid \mathbf{f}) p(\mathbf{f} \mid \mathbf{u}) p(\mathbf{u}), \tag{1}$$

where $p(\mathbf{f} \mid \mathbf{u}) = \mathcal{N}(\mathbf{f} | \mathbf{K_{xz}}\mathbf{K_{xx}^{-1}}\mathbf{u}, \mathbf{K_{xz}}\mathbf{K_{zz}^{-1}}\mathbf{K_{zx}})$ (i.e., $\mathbf{K_{xz}} = \mathbf{K_{xz}^T}$), $p(\mathbf{u}) = \mathbb{N}(\mathbf{u} | 0, \mathbf{K_{zz}})$, and $\mathbf{u}$ is treated as a column vector here. $\mathbf{K_{xz}}$ and $\mathbf{K_{zz}}$ represent covariance matrices with components $k(x_n, z_m)$ for $n = 1, \ldots, N$ and $m = 1, \ldots, M$ and $k(z_m, z_{m\prime})$ for $m, m\prime = 1, \ldots, M$, respectively. The SGP predictive belief can also be computed in a closed form by marginalizing $\mathbf{u}$ out: $p(\mathbf{f}^* \mid \mathbf{y}) \triangleq \int p(\mathbf{f}^* \mid \mathbf{u}) p(\mathbf{u} \mid \mathbf{y}) \mathrm{d}\mathbf{u}$. The unifying view of the SGP model can be referred to in [31, 32].

Inference for the GP model is analytically possible when the likelihood is Gaussian. For the non-Gaussian likelihoods, approximation approach should work. Titsias [33] proposed a seminal variational inference (VI) framework that approximates the joint posterior distribution $p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})$ with a variational posterior $q(\mathbf{f}, \mathbf{u}) \triangleq p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u})$ by minimizing the Kullback–Leibler (KL) distance between them: $\mathrm{KL}[q(\mathbf{f}, \mathbf{u}) | p(\mathbf{f}, \mathbf{u} \mid \mathbf{y})]$. And this procedure is equivalent to maximizing evidence lower bound (ELBO) of the log-marginal likelihood [32, 34]:

$$\mathrm{ELBO} = \mathrm{E}_{p(\mathbf{f} \mid \mathbf{u}) q(\mathbf{u})} [\log(p(\mathbf{y} \mid \mathbf{f}))] - \mathrm{KL}[q(\mathbf{u}) | p(\mathbf{u})]. \tag{2}$$

A common choice in VI is the Gaussian variational posterior $q(\mathbf{u}) = \mathcal{N}(\mathbf{u} \mid \mathbf{m}, \mathbf{S})$, which results in a Gaussian marginal $q(\mathbf{f}) = \mathcal{N}(\mathbf{u} \mid \mu, \sum)$, where $\mu = \mathbf{K_{xz}}\mathbf{K_{zz}^{-1}}\mathbf{m}$ and $\sum = \mathbf{K_{xx}} + \mathbf{K_{xz}}\mathbf{K_{zz}^{-1}}(\mathbf{S} - \mathbf{K_{zz}})\mathbf{K_{zz}^{-1}}\mathbf{K_{zx}}$. A gradient-based algorithm can be employed to maximize the ELBO with respect to the inducing point and hyperparameters in the chosen kernel function. Several common used kernel functions can be found in Table 1 and discussed in [35].

3. Materials and Methods

3.1. Problem Statement. To mathematically formulate the autonomous driving task, we refer to the basic theory of deep reinforcement learning. Let S, A, and R be the state space, action space, and the reward function. In the standard reinforcement learning setting, an agent interacts with the environment at discrete time steps. At each time step t, the agent observes the state $s_t \in S$ and takes an action $a_t \in A$, according to its policy π, which maps a state to a deterministic action or a probability distribution over the actions ($a_t = \pi(s_t)$). Then, it receives an immediate reward $r(s_t, a_t) \in R$ from the environment. The goal of a reinforcement learning task is to learn an optimal policy π^* by maximizing the expected accumulated reward from the beginning. In the DDPG setup, it adopts deep neural network to approximate deterministic policy and action value function. However, training the deep neural network costs too much time and needs a lot of data.

In our setting, we treat the deep GP model $\mathbf{M}$ as the policy. To get the optimal deep GP model, the N training data $\left\{\mathbf{X} \triangleq \{\overline{s}_t\}_{t=1}^N, \mathbf{Y} \triangleq \{\overline{a}_t\}_{t=1}^N\right\}$ collected from interaction between well-trained neural network and Torcs engine, which consists of state set regarding sensor's states and action set from the controller in Torcs, are used to train the model $\mathbf{M}$. Each state $\overline{s}_t$ and action $\overline{a}_t$, as well as s_t and a_t, are represented by several variables presented in Tables 2 and 3, respectively. And the reward function we defined is as follows:

$$r(s_t, a_t) = v_x * \cos(\psi) * (1 - \sin|\psi|) * \left(1 - \left|d_2\right|\right). \tag{3}$$

The reward function can be constructed more effectively by including other related variables [36]. Although the reward function does not contain the variables of action at explicitly, each state s_{t+1} is observed at time step $t + 1$ after taking the action at time step t. It influences the result of reward value indirectly.

To get the best policy, the evidence lower bound, denoted as $\mathrm{ELBO_M}$, for the deep GP model $\mathbf{M}$, which is more complex than standard GP should also be maximized using training data, $\mathbf{X}$ and $\mathbf{Y}$. According to the Bayesian theory, it means to get the statistical mean of action a_t mapping from state s_t, denoted as $\mathrm{E_M}[a_t \mid s_t, \mathbf{X}, \mathbf{Y}]$. After optimization, the model $\mathbf{M}$ will be settled, denoted as $\widetilde{\mathbf{M}}$, which yields $a_t = \widetilde{\mathbf{M}}(s_t)$. As you can imagine, though the trained deep GP can fit the training data well, it does not guarantee the optimal reward value in the testing period. With that being considered, we introduce the feedback control method F to refine the output of the deep GP model, i.e., $\widetilde{a}_t = F(a_t, s_t, \mathbf{X})$. In this method, we consider data ψ_{ref} and $\mathbf{d}_{2\mathrm{ref}}$, which represent collection of ψ and d_2 in each state $\overline{s}_t$, in the training data $\mathbf{X}$. To achieve better reward, it is designed to optimize action a_t according

Table 1: The mathematical expression of some kernel functions.

Name	Expression
RBF	$\sigma^2 \exp[-((\mathbf{x}-\mathbf{x}')^2/2l^2)]$
StdPeriodic	$\theta \exp[-(1/2)\sum_{i=1}^{d}((\sin(\pi/T_i)(\mathbf{x}_i-\mathbf{x}'_i))/l_i)^2]$
RatQuad	$\sigma^2[1+t((\mathbf{x}-t\mathbf{x}')^2/2)]^{-\alpha}$
White	$\alpha\delta(\mathbf{x}-\mathbf{x}')$
MLP	$(2\sigma^2/\pi)\arcsin((\sigma_w^2\mathbf{x}^{\mathbf{T}}t\mathbf{x}'n+q\sigma_b^2)/(\sqrt{\sigma_w^2\mathbf{x}^{\mathbf{T}}\mathbf{x}'+\sigma_b^2+1}\sqrt{\sigma_w^2\mathbf{x}^{\mathbf{T}}\mathbf{x}'+\sigma_b^2+1}))$
Matern52	$\sigma^2(1+t\sqrt{5}n\lvert xt-n\mathbf{x}'\rvert q+h(5/3)_{(\mathbf{x}-t\mathbf{x}')^2)\exp(-\sqrt{5}t\lvert\mathbf{x}-t\mathbf{x}'\rvert)}$

Table 2: The information of state $\bar{s}_t$.

Name	Range	Detail description
ψ	$[-1, 1]$	Angle between the car direction and track axis direction
d_1	$[0, 1]$	Distance between the car and the track edge in front of the car
d_2	$[-\infty, \infty]$	Shift from the center line
v	$[-\infty, \infty]$	Car speed in x, y, and z direction
ω	$[-\infty, \infty]$	The speed of four wheels
g	$[0, \infty]$	Gear speed

Table 3: The information of state $\bar{a}_t$.

Name	Command	Range	Detail description
ξ	Steering	$[-1, 1]$	-1 and 1 mean full right and full left, respectively
ϕ	Acceleration	$[0, 1]$	Virtual gas pedal (0 means no gas, 1 means full gas)
φ	Brake	$[0, 1]$	Virtual brake pedal (0 means no brake, 1 means full brake)

to the difference between state s_t and the training state $\bar{s}_t$. Our solution presents in the following expression:

$$\text{Deep GP model } \widetilde{\mathbf{M}}\text{: } \max_{\text{ELBO}_{\mathbf{M}}} \mathrm{E}_{\mathbf{M}}\left[a_t \mid s_t, \mathbf{X}, \mathbf{Y}\right], \tag{4}$$

$$\text{Feedback Control model } F\text{: optimize } \underset{|s_t-\bar{s}_t|}{F(a_t, s_t, \mathbf{X})}. \tag{5}$$

All the details will be presented in the next section. To compare our method with deep reinforcement learning, we performed an autonomous driving simulation of the lane keeping task in the Torcs engine.

3.2. Proposed Solution. In this section, the details of our autonomous driving decision-making methods for lane keeping task is given. The whole framework is presented in Figure 1. After training, we can get a fairly good deep GP model $\widetilde{\mathbf{M}}$ to fit the training data. The trained deep GP model is used to predict the action a_t according to the state feedback s_t from Torcs in each step. For validation, all the predicted actions are further refined by feedback control method F for feasibility and safety concerns. Then, the final actions $\tilde{a}_t$ are then sent to Torcs to demonstrate visually the performance on running a successful lap. The overall algorithm flow is shown in Algorithm 1. In the following content, we will discuss core methods in our framework in detail.

3.2.1. Deep GP Model $\widetilde{\mathbf{M}}$. As for this multidimension input and output problem, we use the deep GP method to fit the training data in consideration of its advantage over a standard GP [20]. A multilayer GP model is a hierarchical composition of GP. Considering a deep GP with a depth of L, each GP layer is associated with a set $_{\mathbf{F}l-1}$ of inputs and a set $_{\mathbf{F}l}$ of outputs for $l = 1, \ldots,$ $_{\mathbf{F}0} = \mathbf{X}$ and $_{\mathbf{F}L+1} = \mathbf{Y}$. An example of deep GP is as follows:

$$\mathbf{Y} = {}_{\mathbf{f}1:\,L} + \epsilon = \mathbf{f}_L(\mathbf{f}_{L-1}(\ldots\mathbf{f}_2(\mathbf{f}_1(\mathbf{X}))\ldots)) + \varepsilon, \tag{6}$$

where $\mathbf{f}_l \sim GP(0, \mathbf{K}^l_{\mathbf{F}_{l-1}\mathbf{F}_{l-1}})$ for each layer. Each layer has different kernels. For deep GP, each layer is governed by GP; however, the overall prior $\mathbf{f}_{1:\,L}$ is no longer a GP which makes it intractable to train a deep GP model. For reasonability, we can introduce the Gaussian noise in each layer. In this case, we can get the following recursive definition:

$$\mathbf{F}_l = \mathbf{f}_l(\mathbf{F}_{l-1}) + \varepsilon_l. \tag{7}$$

A graphical model for deep Gaussian process with one hidden node is illustrated in Figure 2.

Let $\mathscr{F} \triangleq \{\mathbf{F}_l\}_{l=1}^{L}$; for supervised learning case, the distribution of a deep GP model with L hidden layers can be written as follows:

$$p(\mathbf{Y}, \mathscr{F}) = p\left(\mathbf{Y} \mid \mathbf{F}_L\right)\prod_{l=2}^{L} p\left(\mathbf{F}_l \mid \mathbf{F}_{l-1}\right)p(\mathbf{F}_1 \mid \mathbf{X}). \tag{8}$$

As for the conditional probabilities, they can be expanded as follows:

$$p(\mathbf{F}_l|\mathbf{F}_{l-1}) = \int p\left(\mathbf{F}_l \mid \mathbf{f}_l\right)p\left(\mathbf{f}_l \mid \mathbf{F}_{l-1}\right)\mathrm{d}\mathbf{f}_l. \tag{9}$$

The nonlinearities introduced by the GP covariance functions make the Bayesian treatment of this model

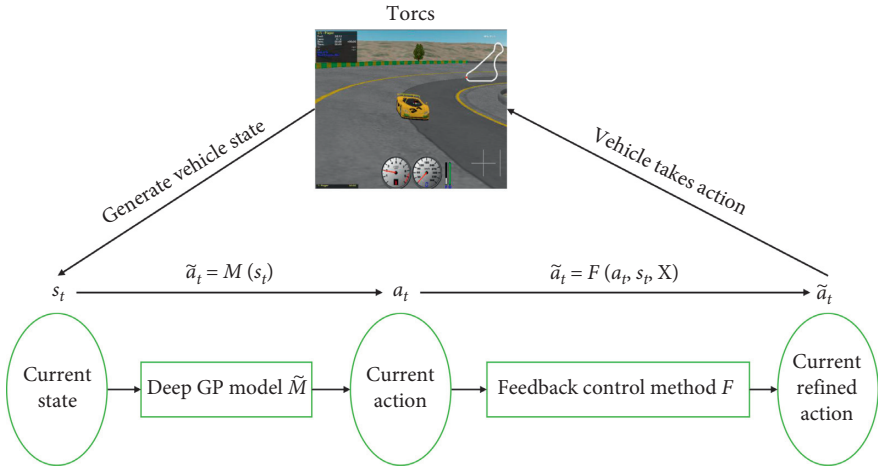

Figure 1: The framework of our solution: in each time step, the Torcs engine generates the vehicle state s_t. The trained deep GP model $\tilde{\mathbf{M}}$ maps the state s_t into action at. After that, the action a_t is refined by feedback control method F, i.e., $\tilde{a}_t = F(a_t, s_t, \mathbf{X})$. Finally, the visual vehicle takes action $\tilde{a}_t$. These procedures cycle until the vehicle finishes the single lap.

(1) collect data from interaction between Well-Trained Network and Torcs engine
(2) set elements for deep GP: layer number, kernel, inducing points, etc.
(3) train a deep GP model and save
(4) aunch Torcs
(5) **for** $i = 1, \mathrm{N}$ **do**
(6) reset Torcs
(7) get the initial state
(8) **for** $j = 1, \mathrm{M}$ **do**
(9) predict action with state using deep GP model $\tilde{\mathbf{M}}$
(10) amend action by feedback control method F
(11) get the next state
(12) **if** unsuccessful loop trip **then**
(13) break
(14) **end if**
(15) **if** successful loop trip **then**
(16) save the experience data
(17) break
(18) **end if**
(19) **end for**
(20) **end for**
(21) shut down Torcs

Algorithm 1: Overall algorithm flow.

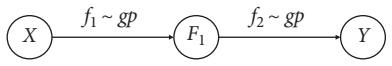

Figure 2: A deep Gaussian process with one hidden node.

challenging. Inspired by the core idea of the SGP model, it is practical to introduce the inducing inputs and corresponding inducing output variables for GP layers, denoted by the respective sets $\mathcal{Z} \triangleq \{\mathbf{Z}_l\}_{l=1}^{L}$ and $\mathcal{U} \triangleq \{\mathbf{U}_l\}_{l=2}^{L+1}$. Now, similar to equation (9), we could write that

$$p\left(\mathbf{F}_l \mid \mathbf{U}_l, \mathbf{F}_{l-1}\right) = \int p\left(\mathbf{F}_l \mid \mathbf{f}_l\right) p\left(\mathbf{f}_l \mid \mathbf{U}_l, \mathbf{F}_{l-1}\right) \mathrm{d}\mathbf{f}_l. \tag{10}$$

In this way, we can obtain the logarithm of the augmented joint distribution:

$$\begin{aligned} \log p(\mathbf{Y}, \mathcal{F} \mid \mathcal{U}) &= \log p\left(\mathbf{F}_1 \mid \mathbf{X}\right) + \sum_{l=2}^{L+1} \log p\left(\mathbf{F}_l \mid \mathbf{U}_l, \mathbf{F}_{l-1}\right), \\ &\geq \log p(\mathbf{F}_1 \mid \mathbf{X}) + \sum_{l=2}^{L+1} \mathcal{L}_l, \end{aligned} \tag{11}$$

where $p(\mathbf{F}_1 \mid \mathbf{X}) = \mathcal{N}(\mathbf{F}_1 \mid 0, \mathbf{K}_{\mathbf{xx}}^1 + \nu_1^2)$ and $\mathcal{L}_l$ is the lower bound for $\log p(\mathbf{F}_l \mid \mathbf{U}_l, \mathbf{F}_{l-1})$:

$$\mathscr{L}_l = \log p\left(\mathbf{F}_l \mid \mathbf{a}_l, \nu_l^2 \mathbf{I}\right) - \frac{1}{2\nu_l^2} tr\left(\widetilde{K}_l\right) \tag{12}$$

where $\mathbf{a}_l = \mathbf{K}^l_{\mathbf{F}_{l-1}\mathbf{Z}_{l-1}} (\mathbf{K}^l_{\mathbf{Z}_{l-1}\mathbf{Z}_{l-1}})^{-1} \mathbf{U}_l$ and $\widetilde{K}_l = \mathbf{K}^l_{\mathbf{F}_{l-1}\mathbf{F}_{l-1}} - \mathbf{K}^l_{\mathbf{F}_{l-1}\mathbf{Z}_{l-1}} (\mathbf{K}^{\mathbf{l}}_{\mathbf{Z}_{l-1}\mathbf{Z}_{l-1}})^{-1} \mathbf{K}^l_{\mathbf{Z}_{l-1}\mathbf{F}_{l-1}}$ [21]. And we can see that the latent variable $\mathbf{f}_l$ are integrated out within each layer. Our aim is to approximate the logarithm of the marginal likelihood:

$$\log p(\mathbf{Y}) = \log \int p(\mathbf{Y}, \mathscr{F} \mid \mathscr{U}) \prod_{l=2}^{L+1} p(\mathbf{U}_l) \mathrm{d}\mathscr{U} \mathrm{d}\mathscr{F}, \tag{13}$$

where $p(\mathbf{U}_l) = \mathcal{N}(\mathbf{U}_l \mid 0, \mathbf{K}^l_{\mathbf{Z}_{l-1}\mathbf{Z}_{l-1}})$. To get the bound $\mathrm{ELBO}_{\mathbf{M}}$ for marginal likelihood, with Jensen's inequality, we can get that

$$\mathrm{ELBO}_{\mathbf{M}} = \int \mathscr{Q}\, p(\mathbf{Y}, \mathscr{F} \mid \mathscr{U}) \frac{\prod_{l=2}^{L} (\mathbf{U}_l)}{\mathscr{Q}\, \mathrm{d}\mathscr{U} \mathrm{d}\mathscr{F}}, \tag{14}$$

where $\mathscr{Q} = q(\mathscr{F}, \mathscr{U})$ is the introduced approximate variational distribution.

Generally, the $\mathrm{ELBO}_{\mathbf{M}}$ can be more simplified by mean field approximation $\mathscr{Q} = \prod_{l=1}^{L} q(\mathbf{U}_{l+1}) q(\mathscr{F})$ (i.e., $q(\mathbf{U}_{l+1}) = \mathcal{N}(\mathbf{m}_{l+1}, \mathbf{S}_{l+1})$, $q(\mathscr{F}) = \mathcal{N}(\mu_l, \Sigma_l)$ in each layer), and the final form of $\mathrm{ELBO}_{\mathbf{M}}$ can be tractable because of these conjugate distributions, when the covariance functions selected in each layer are feasibly convoluted with the Gaussian density [20, 21]. A gradient-based algorithm, such as L-BFGS-B algorithm [37], can be employed to maximize the variational lower bound ELBOM with respect to the model parameters (i.e., kernel hyperparameter θ_l and noise variance ν_l in each layer) and variational parameters are introduced:

$$\text{model parameters} : \{\theta_l, \nu_l\}_{l=1}^{L+1}, \tag{15}$$

$$\text{variational parameters} : \{\mathbf{Z}_l, \mathbf{m}_{l+1}, \mathbf{S}_{l+1}, \boldsymbol{\mu}_l, \boldsymbol{\Sigma}_l\}_{l=1}^{L}. \tag{16}$$

The trained deep GP model $\widetilde{M}$ can fit the training data well. In recent years, other several approximation methods are put forward to train deep GP such as importance-weighted variational inference [38], stochastic gradient Hamiltonian Monte Carlo [39], and approximate expectation propagation [40].

3.2.2. Feedback Control Model F. After training the deep GP model, all the parameters in the model $\widetilde{M}$ are settled. We try this model in Torcs and find that it can only finish a little more than half loop trip on the CG road. After analysing the failed experience and the input data, we find the essential cause is that the data from DDPG well-trained network only contain the state cases s_t with small ψ and d_2, which are close to the center line of the lane. With the input state s_t with highly deviated ψ or d_2 value, the trained deep GP $\widetilde{M}$ may generate action a_t with improper ξ, ϕ, or φ value. And it indirectly affects the value of the reward function. We assume that if the values of ψ and d_2 stay in a reasonable range, the successful loop trip can be achieved regardless of the values of other state variables. With that being thought, we design an extra feedback control method F, for reward optimization, to amend action a_t predicted by the deep GP model.

In this method, we refer to the idea of the PID controller method. For simplicity, unlike the PID controller, we only add proportional changing errors, but it is composed of two different positive items, to the predicted steer value ξ in the action a_t. In addition, instead of using integral error terms, we take the past state into consideration by adding the error between the current state and a reference state in the past. Firstly, several critical values γ_n, $\eta_n (n = 1, 2, 3, 4)$ need to be set in our method. The reason for doing this is that the feedback control is only needed for those improper feedback state variables, ψ and d_2. Secondly, the error λ_n is calculated by the difference between jth or $-(j - N)$th number of variable, when j is smaller or larger than N, ψ_{ref} and $\mathbf{d}_{2\mathrm{ref}}$ in $\bar{s}_t$, and current feedback state variable, ψ and d_2. Finally, the parameters b_n, c_n of the linear error term need to be regulated to achieve loop trip. The detailed algorithm flow is presented in Algorithm 2.

As we can see, there are many adjustable parameters in our feedback control method. Actually, it turns out that all the parameters can be easily determined:

(i) According to our experience, γ_n and $\eta_n$can be set immediately after analysing the domain of ψ_{ref} and $\mathbf{d}_{2\mathrm{ref}}$. Because the logical judgment in F should only be needed when the vehicle deviates too much from the center line or the steering angle is too large.

(ii) And the value of $b_{3,4}$ and $c_{3,4}$ can be set the same as $b_{1,2}$ and $c_{1,2}$. Consequently, only four parameters in the feedback control method are left to be considered seriously.

(iii) Besides, there are two logical judgment statements after iteration number check in our method. This actually corresponds to the case that the visual vehicle is in the left or right side of the center line. We should consider these two situations separately. Moreover, the reason why we use the absolute value of the error λ_n is that the sign of the steer angle ξ should be always in the correct direction, with its value larger (left side of center line) or smaller (right side of center line) than predicted values by deep GP model $\widetilde{M}$.

(iv) In other words, the absolute value of output action from model $\widetilde{\mathbf{M}}$ is not large enough to drag the vehicle back into the safe road in some extreme dangerous situations. From this perspective, the sign of rest four parameters to be determined will be obvious.

Unlike usual optimization routine, the optimization of the reward value in each step is not carried out by a gradient-based algorithm. Actually, for lane keeping task, if the vehicle can complete the lap successfully, the obtained reward value may be not the best, but it must be one of the local optimal values. After enough trial and error, we can get the relatively optimal parameters introduced in F. In this way, after the parameters in method F are determined, the vehicle

Input: action at each time step a_t: $\{\xi, \phi, \varphi\}$
two variables of state s_t: $\{\psi, d_2\}$
training state set X: $\{\psi_{\text{ref}}, \mathbf{d}_{2\text{ref}}\}$
Output: refined action $\widetilde{a}_t$: $\{\widetilde{\xi}, t\phi n, q\varphi\}$

(1) $a_t = \{\xi, \phi, \varphi\}$
(2) if $j < N$ then
(3) if $\psi > \gamma_1$ or $d_2 < \eta_1$ then
(4) $\lambda_1 = b_1 |\psi - \psi_{\text{ref}}[j]| + c_1 |d_2 - \mathbf{d}_{2\text{ref}}[j]|$
(5) $\widetilde{\xi} = |\xi| + \lambda_1$
(6) end if
(7) if $\psi < \gamma_2$ or $d_2 > \eta_2$ then
(8) $\lambda_2 = b_2 |\psi - \psi_{\text{ref}}[j]| + c_2 |d_2 - \mathrm{d}_{2\text{ref}}[j]|$
(9) $\widetilde{\xi} = \lambda_2 - |\xi|$
(10) end if
(11) else
(12) if $\psi > \gamma_3$ or $d_2 < \eta_3$ then
(13) $\lambda_3 = b_3 |\psi - \psi_{\text{ref}}[-(j-N)]| + c_3 |d_2 - \mathbf{d}_{2\text{ref}}[-(j-N)]|$
(14) $\widetilde{\xi} = |\xi| + \lambda_3$
(15) end if
(16) if $\psi < \gamma_4$ or $d_2 > \eta_4$ then
(17) $\lambda_4 = b_4 |\psi - \psi_{\text{ref}}[-(j-N)]| + c_4 |d_4 - \mathrm{d}_{2\text{ref}}[-(j-N)]|$
(18) $\widetilde{\xi} = \lambda_4 - |\xi|$
(19) end if
(20) **end if**
(21) $\widetilde{a}_t = \{\widetilde{\xi}, t\phi n, q\varphi\}$

Algorithm 2: Feedback control method F.

can immediately respond to the Torcs engine through the refined action $\widetilde{a}_t$.

4. Results and Discussion

In this section, we conduct extensive simulations to valid our method and compare it with reinforcement learning approach that are typically used in a similar setting. We start with experiment setup about data preparation, then show how well our model fits the training data, and finally provide comparison by examining the performance of lane keeping in a simulation environment.

4.1. Experiment Setup. DDPG is a variant of deterministic policy gradient algorithm [23], which adopts deep neural network to approximate deterministic policy and action value function. It is an off-policy algorithm, utilizing the experience replay technique introduced in DQN [25] to break the correlation of the samples and keep samples i.i.d. In addition, the learning method of Q function is similar to that in DQN as well. In our case, we train a deep neural network by DDPG to achieve successful loop trip. It takes about 16 hours and 4000 episodes to achieve a high performance deep neural network. And tens of thousands of data will be updated in the centralized experience replay buffer during training period.

We collect the training data by DDPG well-trained network on the CG road in software Torcs. 338 records ($N = 338$) are collected during the loop trip simulation. It contains state set and action set of the visual vehicle. The detail of state $\overline{s}_t$ and action $\overline{a}_t$ are already shown in Tables 2 and 3, respectively. To train deep GP network, the state set $\mathbf{X}$ is the input and the action set $\mathbf{Y}$ is the output. And the raw data are fed into deep GP network without any additional data processing before training.

4.2. Experimental Results. In our case, we use the *GPy* [41], which is an open framework developed by Sheffield machine learning group, to conduct simulation. We use two layers GP to fit the training data. The kernels we used per layer are as follows:

$$[\text{StdPeriodic} * \text{RatQuad} + \text{RBF} + \text{White}, \text{MLP} * \text{Matern52} + \text{RBF} + \text{White}]. \tag{17}$$

All the function expressions corresponding to these function names can be found in Table 1 or the *GPy* document web page. And their corresponding automatic relevance determination (ARD) [35] version can be easily extended. The number of inducing points we used in each layer is 200. After optimization, the output action values are shown in Figures 3–5. In these figures, the true data and predicted mean value legends mean the true training data action values and the predicted values after finishing training deep GP, respectively. The green zone, with its margin depicted by the green dashed line, in the figures, represents the 95% credible interval of the predicted value. The x-axis

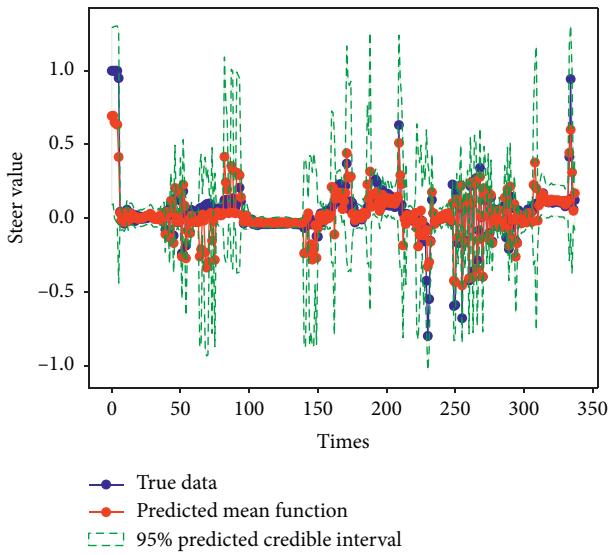

Figure 3: Steer value vs. times.

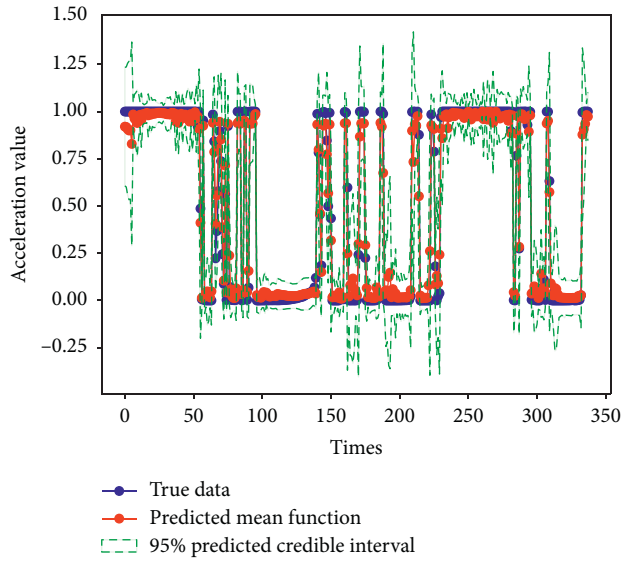

Figure 4: Acceleration value vs. times.

variable *Times* represents the time steps of self-driving. We can see that the model $\widetilde{M}$ can capture the most main features of the train data except for few strong vibration zones.

Recall that we stated at the beginning that the deep GP mode $\widetilde{M}$ is not enough to solve lane keeping task in our setting. And we compare the cumulative reward value between the deep GP method with and without combining the feedback control method. In Figure 6, it demonstrates that the deep GP model can only finish about half loop trip on CG road, but after combining the feedback control method, the accumulated reward value increases to about 2.7 times more as using the deep GP model only. It proves the effectiveness of the feedback control method. In Section 2.2, we already explained the main reasons why the lane keeping task cannot be completed using the deep GP model only. In

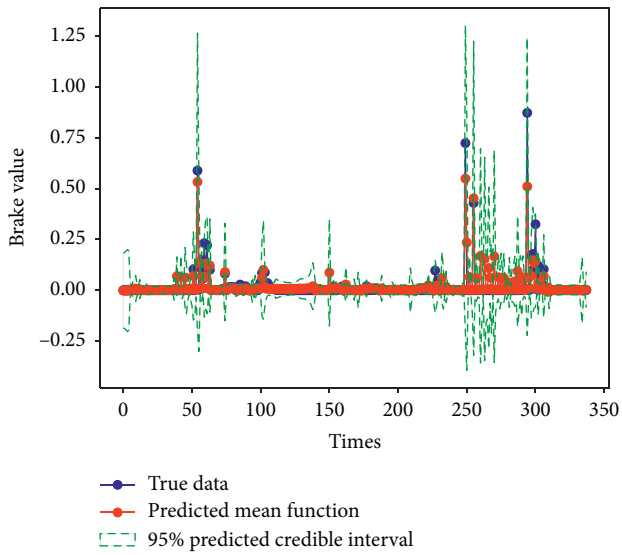

Figure 5: Brake value vs. times.

addition, although the deep GP model can capture the uncertainty very well, it does not have the ability to correct the wrong predicted actions. In such a rapidly interactive environment, these unreasonable actions are so fatal that the vehicle is much more likely to rush out of the track.

4.3. Experimental Comparison. In Section 4.2, we show how our model fit the training data and the necessity of the feedback control method in our case. And now, we compare their performance with the DDPG method. Compared to the DDPG method, the other two methods take more steps to achieve loop trip and the total rewards are a little less than DDPG. So, their ascending curves of accumulated rewards in each step have more flat slopes than the DDPG method. Table 4 lists several properties, such as *Total Rewards*, *Training Time*, and *Training Data*, of the three methods. In spite of advantages in iteration times and total rewards, the DDPG well-trained network, which is used to get training data, costs about 16 hours to train, and it only takes about 1.5 hours to train the deep GP model. It is also much less than the well-trained network with the AMDDPG method according to the result in the paper [36]. In DDPG and AMDDPG methods, they need to interact with the environment in each episode to update the new training data, and this procedure will be repeated for multiple times for exploration and exploitation. Thus, these data-hungry approaches need tens of thousands of data, but the training data we used only contains about 340 items, which is far more less than what is required. All the simulations are conducted on the CG track in Torcs (the overview map in upright corner in Figure 1). Other complex tracks, shown in Figure 7, can be found in Torcs engine or generated by an online tool named TrackGen [42].

With these benefits, we believe that the proposed framework is a promising way to make decisions in simulation environments and actual road conditions. However, there are many technical problems to tackle to achieve the

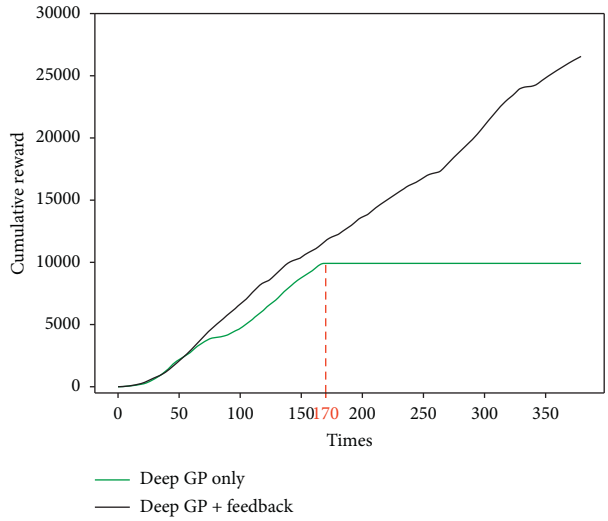

Figure 6: Cumulative reward in each step by deep GP with or without combining feedback control.

Table 4: Comparison of three methods.

Methods	Times	Total rewards	Training time (h)	Training data	Pass CG
DDPG	338	28148.09	16	10^5	Yes
Deep GP	170	9917.06	1.5	338	No
Deep GP + Feedback	380	26555.32	1.5	338	Yes

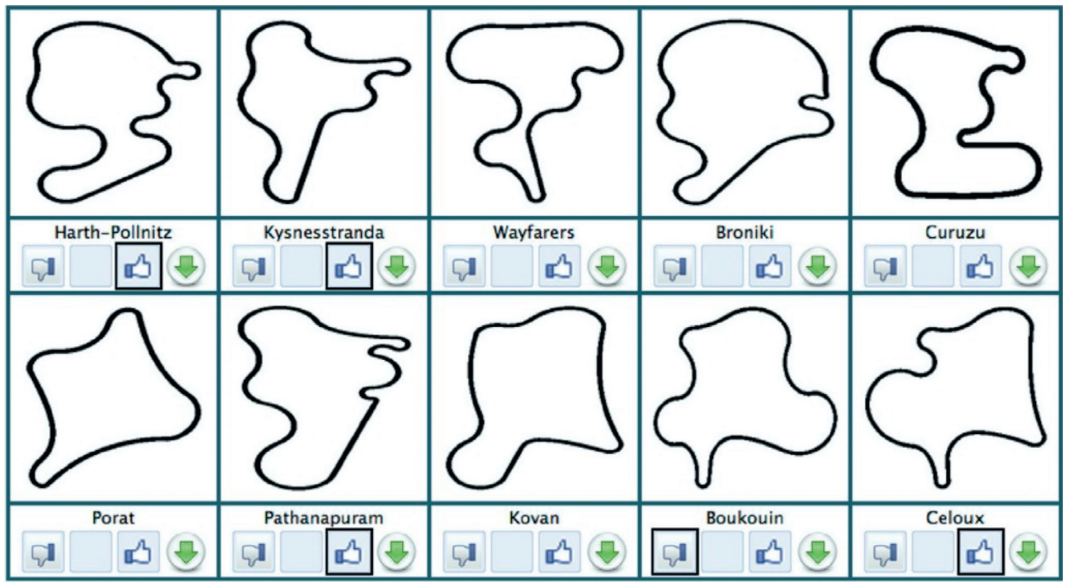

Figure 7: Ten tracks generated by TrackGen.

real road test. Admittedly, the shortcomings of the proposed method should also be acknowledged. In this paper, we only test the methods for lane keeping task on a relative simple road. On a complex road or executing a complex task, it is obvious that more data should be fed into the deep GP model and the feedback control method also needs to be dedicatedly designed and validated. For example, doing simulation on Curuzu track in Figure 7, we can imagine that more training data will be recorded by a well-performed reinforcement learning model. And for this road with many irregular turns, the feedback control method must be tested with extensive trial and error. To check the effectiveness of the proposed framework in real road tests similar to simulation setting, we can record the training data with the aid

of perceptual equipment by manual driving. After training the deep GP model off-line, we can test its validity in both autonomous driving and manual driving modes. In addition, the parameters in the feedback control method should also be regulated to complete the self-driving task. For more complex tasks, such as car-following or overtaking, since the feedback control method in our framework do not take the motional characteristics of the vehicle into consideration, we plan to combine our framework with other motion control methods, such as pure pursuit [43] or Stanley [44], in the future work.

5. Conclusions

In conclusion, we presented an end-to-end learning method which combines the deep GP and feedback control method to solve decision-making problem of lane keeping task in self-driving simulation. The proposed method achieved almost the same performance with only 0.34% of the training data, compared with deep reinforcement learning, and the time consumption of the training is only 10%. We believe this method is a promising one when dealing with complex self-driving tasks with small training data.

Data Availability

The raw data are available online in my github repository (https://github.com/Fangwq/Traning-data-for-decision-making-research).

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors would like to thank Junta Wu's, who is the author of AMDDPG, help for training the DDPG network, supplying the data, and providing many helpful clarifications and suggestions. This work was supported by Shenzhen Engineering Laboratory on Autonomous Vehicles, NSFC (61672512 and 61702493) and Shenzhen Basic Research Program (JCYJ20170818164527303 and JCYJ20180507182619669), Science and Technology Development Fun, Macao S.A.R. (FDCT) (No.0015/2019/AKP), and CAS Key Laboratory of Human-Machine Intelligence-Synergy Systems, Shenzhen Institutes of Advanced Technology. The work was also funded by Shenzhen Institute of Artificial Intelligence and Robotics for Society.

References

[1] L. Li, J. Song, F.-Y. Wang, W. Niehsen, and N.-N. Zheng, "IVS 05: new developments and research trends for intelligent vehicles," *IEEE Intelligent Systems*, vol. 20, no. 4, pp. 10–14, 2005.

[2] Y. Ma, Z. Wang, H. Yang, and L. Yang, "Artificial intelligence applications in the development of autonomous vehicles: a survey," *IEEE/CAA Journal of Automatica Sinica*, vol. 7, no. 2, pp. 315–329, 2020.

[3] Z. Li, L. Li, and Y. Zhang, "IVS 09: future research in vehicle vision systems," *IEEE Intelligent Systems*, vol. 24, no. 6, pp. 62–65, 2009.

[4] B. Paden, M. Cáp, S. Z. Yong, D. Yershov, and E. Frazzoli, "A survey of motion planning and control techniques for self-driving urban vehicles," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 1, pp. 33–55, 2016.

[5] L. Chen, X. Hu, W. Tian, H. Wang, D. Cao, and F.-Y. Wang, "Parallel planning: a new motion planning framework for autonomous driving," *IEEE/CAA Journal of Automatica Sinica*, vol. 6, no. 1, pp. 236–246, 2018.

[6] J. Ziegler, P. Bender, M. Schreiber et al., "Making bertha drive-an autonomous journey on a historic route," *IEEE Intelligent Transportation Systems Magazine*, vol. 6, no. 2, pp. 8–20, 2014.

[7] A. G. Cunningham, E. Galceran, R. M. Eustice, and E. Olson, "Mpdm: multipolicy decision-making in dynamic, uncertain en- vironments for autonomous driving," in *Proceeding of 2015 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1670–1677, IEEE, Seattle, WA, USA, May 2015.

[8] D. Loiacono, L. Cardamone, and P. L. Lanzi, "Simulated car racing championship: competition software manual," 2013, https://arxiv.org/abs/1304.1672.

[9] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, and Carla, "An open urban driving simulator," 2017, https://arxiv.org/abs/1711.03938.

[10] A. Kendall, J. Hawke, D. Janz et al., "Learning to drive in a day," in *Proceeding of 2019 International Conference on Robotics and Automation (ICRA)*, pp. 8248–8254, IEEE, Montreal, QC, Canada, May 2019.

[11] S. R. Granter, A. H. Beck, and D. J. Papke, "Alphago, deep learning, and the future of the human microscopist," *Archives of Pathology & Laboratory Medicine*, vol. 141, no. 5, pp. 619–621, 2017.

[12] Y. Tian, X. Li, K. Wang, and F.-Y. Wang, "Training and testing object detectors with virtual images," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 2, pp. 539–546, 2018.

[13] H. Han, M. Zhou, and Y. Zhang, "Can virtual samples solve small sample size problem of kissme in pedestrian re-identification of smart transportation?" *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–11, 2019.

[14] C. K. Williams and C. E. Rasmussen, *Gaussian processes for machine learning*, MIT press Cambridge, Cambridge, MA, USA, 2006.

[15] S. Thewes, M. Lange-Hegermann, C. Reuber, and R. Beck, *Advanced Gaussian Process Modeling Techniques, Design of Experiments (DoE) in Engine Development*, Expert Verlag, Renningen, Germany, 2015.

[16] M. A. Osborne, R. Garnett, and S. J. Roberts, "Gaussian processes for global optimization," in *Proceeding of 3rd international conference on learning and intelligent optimization*, pp. 176–190, LION 3, Rome, Italy, January 2011.

[17] M. P. Deisenroth, D. Fox, and C. E. Rasmussen, "Gaussian processes for data-efficient learning in robotics and control," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 2, pp. 408–423, 2013.

[18] R. Garnett, S. Ho, and J. Schneider, "Finding galaxies in the shadows of quasars with Gaussian processes," in *Proceeding of International Conference on Machine Learning*, pp. 1025–1033, Carnegie, Mellon University, Pittsburgh, Pennsylvania, University,July 2105.

[19] J. Futoma, *Gaussian Process-Based Models for Clinical Time Series in Healthcare*, Duke University, Durham, NC, USA, 2018.

[20] A. Damianou and N. Lawrence, "Deep Gaussian Processes," in *Proceeding of Artificial Intelligence and Statistics*, pp. 207–215, Universityof Sheffield, Sheffield, England, UK, May 2013.

[21] A. Damianou, *Deep Gaussian Processes and Variational Propa- Gation of Uncertainty*, University of Sheffield, Sheffield, England, UK, 2015.

[22] M. Bojarski, D. Del Testa, D. Dworakowski et al., *End to end learning for self-driving cars*, 2016. https://arxiv.org/abs/1604.07316.

[23] T. P. Lillicrap, J. J. Hunt, A. Pritzel et al., *Continuous control with deep rein- forcement learning*, 2015, https://arxiv.org/abs/1509.02971.

[24] D. Karavolos, *Q-learning with Heuristic Exploration in Simulated Car Racing*, University of Amsterdam, Amsterdam, Netherlands, 2013.

[25] V. Mnih, K. Kavukcuoglu, D. Silver et al., "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[26] M. Sewak, "Deep q network (dqn), double dqn, and dueling dqn," in *Proceeding of Deep Reinforcement Learning*, pp. 95–108, Springer,Berlin, Germany, July 2019.

[27] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proceedings of the 31st International Conference on International Conference on Machine Learning*, DeepMind Technologies, London, UK, June 2014.

[28] B. O'Donoghue, R. Munos, K. Kavukcuoglu, and V. Mnih, "Combining policy gradient and Q-learning," 2016, https://arxiv.org/abs/1611.01626.

[29] F. M. Salem, M. I. Mosaad, and M. A. Awadallah, "A comparative study of mpc and optimised pid control," *International Journal of Industrial Electronics and Drives*, vol. 2, no. 4, pp. 242–250, 2015.

[30] H. O. Bansal, R. Sharma, and P. Shreeraman, "Pid controller tuning techniques: a review," *Journal of Control Engineering and Technology*, vol. 2, no. 4, pp. 168–176, 2012.

[31] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *Journal of Machine Learning Research*, vol. 6, pp. 1939–1959, 2005.

[32] T. D. Bui, J. Yan, and R. E. Turner, "A unifying framework for Gaussian process pseudo-point approximations using power expectation propagation," *The Journal of Machine Learning Re- Search*, vol. 18, no. 1, pp. 3649–3720, 2017.

[33] M. Titsias, "Variational learning of inducing variables in sparse gaussian processes," in *Proceedings of Intelligence and Statistics*, pp. 567–574, University of Manchester, Manchester, England, UK, June 2009.

[34] D. M. Blei, A. Kucukelbir, and J. D. McAuliffe, "Variational inference: a review for statisticians," *Journal of the American Statistical Association*, vol. 112, no. 518, pp. 859–877, 2017.

[35] D. Duvenaud, *Automatic Model Construction with Gaussian Processes*, University of Cambridge, Cambridge, England, UK, 2014.

[36] J. Wu and H. Li, "Aggregated multi-deep deterministic policy gradient for self-driving policy," in *Proceedings of International Conference on Internet of Vehicles*, pp. 179–192, Springer, Paris, France, November 2018.

[37] C. Zhu, R. H. Byrd, P. Lu, and J. Nocedal, "Algorithm 778: L-bfgs-b: fortran subroutines for large-scale bound-constrained optimization," *ACM Transactions on Mathematical Software*, vol. 23, no. 4, pp. 550–560, 1997.

[38] H. Salimbeni, V. Dutordoir, J. Hensman, and M. P. Deisenroth, "Deep gaussian processes with importance-weighted variational inference," 2019, https://arxiv.org/abs/1905.05435.

[39] M. Havasi, J. M. Hernández-Lobato, and J. J. Murillo-Fuentes, "Inference in deep Gaussian processes using stochastic gradient Hamiltonian Monte Carlo," in *Proceedings of Advances in Neural Information Processing Systems*, pp. 7506–7516, Association for Computing Machinery, New York, Ny, USA, December 2018.

[40] T. Bui, D. Hernández-Lobato, J. Hernandez-Lobato, Y. Li, and R. Turner, "Deep Gaussian processes for regression using approx- imate expectation propagation," in *Proceedings of International Conference on Machine Learning*, pp. 1472–1481, Associationfor Computing Machinery, New York, Ny, USA, June 2016.

[41] G. Py GPy, *A Gaussian Process Framework in python*, University of Sheffield, Sheffield, England, UK, 2012.

[42] L. Cardamone, P. L. Lanzi, D. Loiacono, and Trackgen, "TrackGen: an interactive track generator for TORCS and Speed-Dreams," *Applied Soft Computing*, vol. 28, pp. 550–558, 2015.

[43] R. S. Wallace, A. Stentz, C. E. Thorpe, H. P. Moravec, W. Whit- taker, and T. Kanade, "First results in robot road-following," in *Proceedings of the 9th international joint conference on Artificial intelligence*, pp. 1089–1095, Association for Computing Machinery, New York, Ny, USA, August 1985.

[44] S. Thrun, M. Montemerlo, H. Dahlkamp et al., "Stanley: the robot that won the DARPA grand challenge," *Journal of Field Robotics*, vol. 23, no. 9, pp. 661–692, 2006.