2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2022)
October 23-27, 2022, Kyoto, Japan

SMS-MPC: Adversarial Learning-based Simultaneous Prediction Control with Single Model for Mobile Robots

Andong Yang, Wei Li* and Yu Hu*

***Abstract*— Model predictive control is a promising method in robot control tasks. How to design an effective model structure and efficient prediction framework for model predictive control is still an open challenge. To reduce the time consumption and avoid compounding-error of the multi-step prediction process in model predictive control, we propose a single-model simultaneous framework, which uses single dynamics model to predict the entire prediction horizon simultaneously by taking all control actions with the current state as inputs. Based on this framework, we further propose an adversarial dynamics model that contains two parts. The generator provides a dynamics model for the prediction process, while the discriminator provides constraints that are hard to describe by manually defined loss. This adversarial dynamics model can accelerate training and improve model accuracy in unstructured environments. Experiments conducted in Gazebo simulator and on a real mobile robot demonstrate the efficiency and accuracy of the single-model simultaneous framework with an adversarial dynamics model.**

I. Introduction

Model Predictive Control (MPC) offers an effective tool for addressing robot control tasks such as car racing [1], [2], quadcopter flight control [3], and robotic arm gripping [4]. In order to achieve continuous trajectory control, MPC needs to predict the behavior of a dynamics system multi-steps ahead in time. Some existing work refers to this process as the multi-step prediction process [5], [6]. This prediction scheme is accomplished with a dynamics model. Therefore, design an efficient multi-step prediction framework and obtain a sufficiently accurate model is important for MPC.

Due to the limitation of computation cost and model capability, early MPC works [7], [8], [6] adopted the iterative framework, which utilizes the one-step-ahead model to complete the multi-step prediction process iteratively, as shown in Fig. 1(a), where u_t and s_t is control action and state in time instance t, respectively. Therefore, the small errors introduced by the dynamics model will be continuously amplified in the prediction process and finally lead to compounding-error [9]. In addition, the time complexity of this process is $O(N)$, where N is the length of prediction horizon. Because the context information is not considered, the capability of the model employed in iterative MPC is limited to represent real dynamics [10].

Andong Yang, Wei Li and Yu Hu are with the Research Center for Intelligent Computing Systems, Institute of Computing Technology, Chinese Academy of Sciences, University of Chinese Academy of Sciences, Beijing, China. {yangandong19b, liwei2019, huyu}@ict.ac.cn

* Corresponding author.

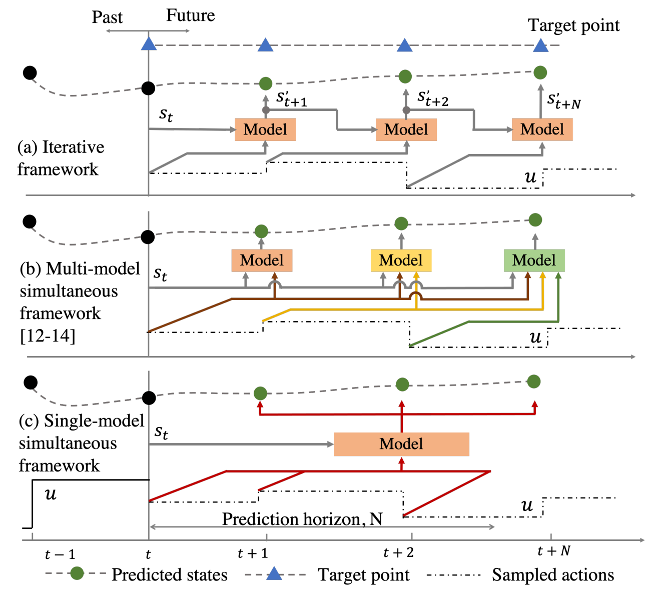

Fig. 1. This figure depicts different frameworks used for the multi-step prediction process in MPC. Iterative framework uses a single model, but requires iterations to predict future states. Multi-model simultaneous framework uses multiple models to complete prediction simultaneously. We propose single-model simultaneous framework to combine the advantages of both. In this figure, we take prediction horizon N=3 as an example.

To alleviate the above-mentioned issues in the iterative framework, researchers proposed methods predicting a sequence of states simultaneously [11], [12], which we call simultaneous framework. As shown in Fig. 1(b), this framework can accelerate the multi-step prediction process and mitigate the compounding-error problem by predicting multiple future states at once. Mishra et al. [13], [14] have presented a compromise approach between iterative framework and simultaneous framework. This work accelerates the prediction process by dividing the prediction horizon into several segments and using a designed model to predict the states in the segments simultaneously. However, this method still suffers from the compounding-error problem because the prediction between segments is iterative. As research on the dynamics model continues to develop, the modeling methods now have the ability to forecast the entire prediction horizon simultaneously in the mobile robot control tasks. Researchers Terzi et al. [11], [12] have proposed methods that use multiple linear models to predict the entire prediction horizon at a time. As illustrated in Fig. 1(b), each model uses actions before the state to be predicted as inputs. This simultaneous framework in the above-mentioned methods not only circumvents the compounding-error problem but also improves the time efficiency of prediction. However, multiple models are complex to design and computationally burdensome. In our work, a more capable dynamics model is proposed. Therefore, it is possible to use only one model to complete the multi-step prediction process and the input actions are extended to align with the prediction horizon. We call this design the Single-Model Simultaneous framework, hereafter referred to as the SMS framework.

In order to implement the SMS framework, as shown in Fig. 1(c), a sufficiently powerful dynamics model is required. The model in MPC is typically defined by two kinds of methods. One kind is the traditional mathematical model, such as dynamic mode decomposition [15]and bicycle model [16]. The other is the learning-based model, such as Gaussian process [17], [4] and neural network [18], [19], [20]. The learning-based model can learn the dynamics of complex environments that cannot be described by the traditional mathematical model, and thus is more suitable for the unstructured environment. However, existing learning-based models [21], [22] uses only the distance between the real and the predicted states to guide model training, which typically does not contain context information such as continuity between two adjacent states or features reflecting different ground, e.g. grass, bricks. Based on adversarial theory, we have built a discriminator that provides hard-to-define constraints, achieving a more accurate dynamic model. The data used to train the dynamic model was collected from casual system, which determines that the model is casual.

In this paper, we propose an SMS framework that employs an adversarial model for the multi-step prediction process of MPC. The SMS framework uses a single model to predict the entire prediction horizon simultaneously, which can reduce the time complexity of the multi-step prediction process from $O(N)$ to $O(1)$ and alleviate the compounding-error problem. Under this framework, we propose an adversarial model that consists of a generator and a discriminator. The generator is used as a dynamics model in the prediction process. The discriminator can provide additional constraints that are hard to represent by manually designed analytical loss for the generator. Therefore, this adversarial model can accelerate the training and improve the model accuracy.

Our contributions in this paper are as follows:

- Propose the single-model simultaneous framework that can decrease the time complexity of the multi-step prediction process in MPC from $O(N)$ to $O(1)$, reducing the number of required dynamics models and alleviating the compounding-error problem.
- Build an adversarial dynamics model to obtain constraints that are difficult to express with hand-crafted loss function, thus accelerating the training process and improving the model accuracy.
- Propose a complete MPC approach that employs a single-model simultaneous framework and an adversarial dynamics model. The effectiveness of our approach was demonstrated by deploying it on both Gazebo simulator and a mobile robot.

II. RELATED WORK

We divide existing frameworks for solving the multi-step prediction process into two categories: iterative and simultaneous. The iterative framework uses a model to predict the state at the next moment based on the current state and action, and then iterates to complete the prediction [23]. This framework alleviates the requirement for a dynamical model [24] but shows non-negligible drawbacks in complex environments, such as computational complexity and compounding-error problem [10].

Recently, the simultaneous framework was proposed to solve problems in the iterative framework. Terzi et al. [11] combine multiple models to predict the entire prediction horizon simultaneously, with each model taking historical states and all the actions before the predicted moment as inputs to predict one future state. In the follow-up, they ensure robust constraints satisfaction by employing the set membership approach to establish relationships between models [12]. The simultaneous framework in the above methods can accelerate the prediction speed, but multiple models are computationally burdensome.

With the development of learning-based dynamics models in recent years [1], [25], [26], more capable models have emerged. For instance, using Gaussian process (GP) [27] to compensate for biases in the fixed first-principles model [2], or using Koopman canonical transform to express the dynamics model as a lifted bilinear model [15]. However, when the environment becomes more complex, such methods may fail to accurately fit the real model. To improve the capability of such model in complex environments, Li et al. [20] use deep neural network as dynamics model while Lefèvre et al. [19] employ multi-layer perceptron to build the dynamics model. Mohajerin et al. [6], [18] leverage the recurrent neural network to extract historical information and thus improve prediction accuracy. But, the losses in the above methods are manually designed analytical expressions, which means constraints or characteristics such as context information and features of different terrains are not considered.

Adversarial theory-based approaches like generative adversarial networks can learn features or constraints that cannot be represented by hand-crafted loss function. This idea has been used to solve prediction problems in recent years. Xu et al. [28] use discriminator to help the generator learn the logic and connections between words that are hard to define, thus encouraging the generator to produce diverse and informative text. Janner et al. [29] address the problem that manually designed loss can not fully reflect domain distinctions when transferring learned policies to new domains with different dynamics by using a discriminant network. However, the input of model in SMS framework includes a series of actions, which are not static historical information, but rather the target to be optimized in MPC. Inspired by the idea of generative adversarial networks, we design an adversarial dynamics model to predict the entire prediction horizon simultaneously and to obtain constraints beyond the manually defined loss.

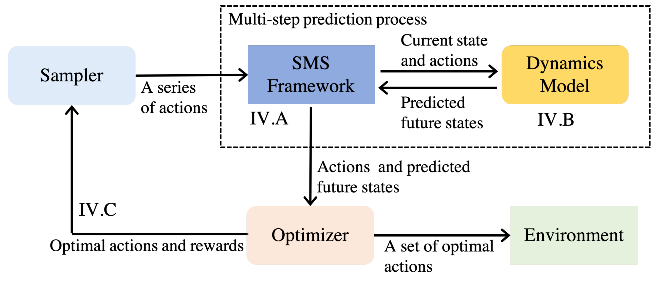

Fig. 2. The entire scheme of SMS-MPC. The multi-step prediction process uses a dynamics model to predict a series of states after the execution of given actions.

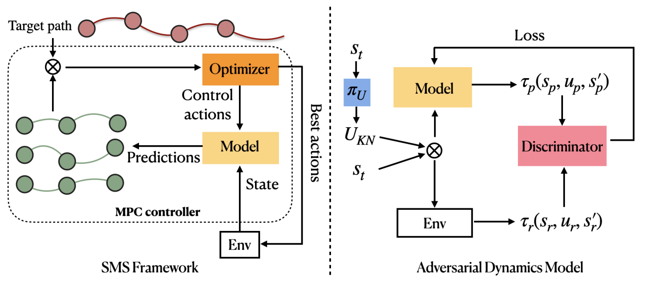

Fig. 3. The overall framework of our method. (Left) Single-model simultaneous framework decreases the time complexity of the multi-step prediction process from $O(N)$ to $O(1)$, not only reducing the number of required dynamics models but also alleviating the compounding-error problem. (Right) Adversarial model obtains additional constraints beyond the manually defined loss to accelerate the training of dynamics model and improve the multi-step prediction performance. τ stands for trajectory.

III. PROBLEM STATEMENT

In this section, we will briefly introduce the overall structure of the multi-step prediction process and then describe the role of dynamics model in this process. As shown in Fig. 2, the multi-step prediction process in MPC uses a dynamics model to predict future states based on a given set of actions, so as to provide basic information for selecting the most appropriate actions.

A. Multi-Step Prediction in MPC

In our multi-step prediction process, the inputs include the current state and a series of given action sequence. At time t, the state $\mathbf{s_t} = [x_t, y_t, \mathbf{d_t}, \mathbf{v_t}]^T$ contains coordinates $[x_t, y_t]$ with $x_t, y_t \in \mathbb{R}^1$, orientation $\mathbf{d_t} \in \mathbb{R}^3$ and velocity $\mathbf{v_t} \in \mathbb{R}^3$. The action sequence can be created by a sampler. We denote the total number of samples as K, and the ith action sequence $U_i = \{\mathbf{u^i_t}, \mathbf{u^i_{t+1}}, ..., \mathbf{u^i_{t+N-1}}\}, i$ in $1, ..., K$. The action $\mathbf{u_t}$ at time t contains acceleration and steering angle. Using N to denote the prediction horizon and i to denote a certain sampling, the outputs of the multi-step prediction process is $S_i' = \{\mathbf{s^{i\prime}_{t+1}}, \mathbf{s^{i\prime}_{t+2}}, ..., \mathbf{s^{i\prime}_{t+N}}\}, i$ in $1, ..., K$. To solve the path following task, the prediction result and the target path $S_{tar} = \{\mathbf{s^{tar}_t}, \mathbf{s^{tar}_{t+2}}, ..., \mathbf{s^{tar}_{t+N}}\}$ will be fed to the optimizer. The optimizer will select the optimal actions $U^* = \{\mathbf{u^*_t}, \mathbf{u^*_{t+1}}, ..., \mathbf{u^*_{t+N-1}}\}$, which will interact with the environment to obtain the state of the next moment.

B. Nonlinear Dynamics Model

In the multi-step prediction process, the dynamics model provides the basis for accurate prediction of future states based on actions. The real dynamics model of environment f^* is defined as

$$\mathbf{s_{t+1}} = f^*(\mathbf{s_t}, \mathbf{u_t}) \tag{1}$$

The dynamics model in the single-model simultaneous framework needs to learn the mapping from the current state s_t and an action sequence U_i to the real future states $S_i = \{\mathbf{s_{t+1}}, ..., \mathbf{s_{t+N}}\}$. The approximate model f parameterized with θ is defined as

$$S_i' = f(\mathbf{s_t}, U_i; \theta) \tag{2}$$

During the training phase, given $(\mathbf{s_t}, U_i, S_i)$ obtained from the collected data, the objective of $f(\mathbf{s_t}, U_i; \theta)$ is to find an optimal θ to minimize a Mean Squared Error Loss (MSELoss) between real future states S_i and predicted future states S_i'. During the testing phase, f outputs the predicted state S_i' based on $\mathbf{s_t}$ and U_i. Compared to the model in the iterative framework, the model in the single-model simultaneous framework needs to have stronger fitting ability. Also, the model will face a more complex loss space since the loss will be computed based on a sequence of states. Thus, we propose an adversarial dynamics model.

IV. METHODOLOGY

In this section, we first present the detailed implementation of the SMS framework. Afterward, we introduce how to design an adversarial dynamics model for the proposed SMS framework. Finally, we describe a complete MPC scheme realized by integrating these structures.

A. Single-Model Simultaneous Framework

As discussed in Sec III-A, in order to find the best action sequence U_i^* for the current state s_t at time t, the multi-step prediction process needs to predict future state sequence S_i' of length N based on the current state $\mathbf{s_t}$ and given actions U_i. Specifically, the sampler first generates a batch of action sequences U_i according to the sampling policy π_U. As shown in Fig. 1, the generated action sequence are fed into the dynamics model f simultaneously, which predicts states during $t+1$ to $t+N$.

In the iterative framework, assume the deviation of dynamics model is δ and the dynamics model is $\mathbf{s'_{t+1}} = f(\mathbf{s_t}, \mathbf{u_t}) + \delta$. The total prediction error will be accumulated in iterations and become severe as the predict length N becomes larger. For single-Model simultaneous framework, assume the deviation of dynamics model is δ_{seq} and the predicted result is $S'_{t+1} = f(\mathbf{s_t}, U_i; \theta) + \delta_{seq}$. Since the prediction only conducted once, the error will not accumulated. However, the single-model simultaneous framework has strict requirements on the capability of the dynamics model, and the loss is defined based on sequence, which is more complex and hard to converge. To address this obstacle, we propose a model based on adversarial learning, which will be presented in Subsection IV-B.

Algorithm 1 Adversarial Model Training

```
1:  Initialize generator $G$, discriminator $D$, prediction horizon $N$, ratio of training times for discriminators and generators $k$, hyperparameters $\alpha, \beta, c$.
2:  Collect data $\mathscr{D} : \{(U_i, S_i'), ...\}$ by executing a random control policy in the training environment.
3:  while not converged do
4:      for $k$ steps do
5:          Sample a batch of $m$ examples from $\mathscr{D}$
            $S_i = \{s_t^i, s_{t+1}^i, ..., s_N^i\}, U_i = \{u_t^i, u_{t+1}^i, ..., u_N^i\}$
6:          Stack $s_t^i$ for $N$ times to form vector $S_{base}$
7:          Predict state sequence $S_i' = G_\theta(s_t^i, U_i) + S_{base}$
8:          $\Delta_\omega = \frac{1}{m}\sum_{i=1}^m D_\omega(s_t, U_i, S_i') - \frac{1}{m}\sum_{i=1}^m D_\omega(s_t, U_i, S_i)$
9:          $\omega_D = \omega_D + clip(\Delta_\omega, -c, c)$
10:     end for
11:     Sample a batch of $m$ examples
        $S_i = \{s_t^i, ..., s_N^i\}, U_i = \{u_t^i, ..., u_N^i\}$ from $\mathscr{D}$
12:     Stack $s_t^i$ for $N$ times to form vector $S_{base}$
13:     Predict state sequence $S_i' = G_\theta(s_t^i, U_i) + S_{base}$
14:     $\Delta_\theta = \alpha\frac{1}{m}\sum_{i=1}^m D_\omega(s_t, U_i, S_i') + \beta\frac{1}{m}\sum_{i=1}^m ||S_i' - S_i||^2$
15:     $\theta_G = \theta_G + \Delta_\theta$
16: end while
```

B. Adversarial dynamics model

In unstructured environments, there are many properties hard to represent by manually defined loss, e.g. context information between states, state sequence patterns of different surfaces such as grass and mud [30]. To make full use of the above properties and make the model compatible with the SMS framework, we propose an adversarial model structure. In this work, the real system including the interaction between mobile robot and environment is hard to describe, so we use an adversarial network to represent the mapping from the current state and action sequence to the future state space. The structure contains two multi-layer neural networks, one is a generator $G(s, U; \theta)$, the input includes the current state s_t and a sequence of action U_i. The other is a discriminator $D(s, U, S'; \omega)$ that estimates the probability that a sample came from the real future state rather than $G(s, U; \theta)$. The discriminator can provide loss to the generator, as illustrated in Fig. 3. In addition, the discriminator provides context information that cannot be directly represented by manually designed loss, especially dramatic state changes that rarely occur in normal environments. The discriminator is only used in the training process and does not increase the complexity at test time.

To exactly match the dynamics model without cumbersome fine-tuning parameters, we propose the following new loss regularizer for the generative net $G(s, U; \theta)$

$$L_g = \alpha\frac{1}{m}\sum_{i=1}^{m} D_\omega(\mathbf{s_t}, U_i, S_i') + \beta\frac{1}{m}\sum_{i=1}^{m} ||S_i' - S_i||^2$$

where S_i' is the predicted state sequence, α, β are the weights of the regularizer, $D_\omega(\cdot)$ is the discriminator parameterized by ω and m is the batch size. This regularizer can provide correct gradient direction to the generator in the early stage of training, thereby accelerating the training phase. The loss used to train the discriminator $D_\omega(\cdot)$ is

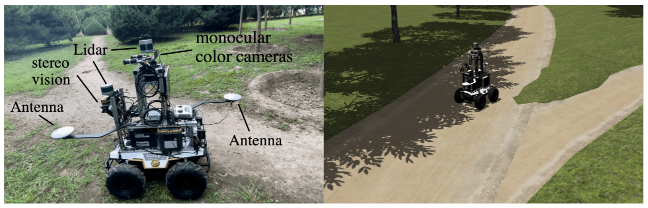

Fig. 4. The mobile robot in real world (left) and Gazebo simulation environment (right).

$$L_d = \frac{1}{m}\sum_{i=1}^{m} D_\omega(\mathbf{s_t}, U_i, S_i') - \frac{1}{m}\sum_{i=1}^{m} D_\omega(\mathbf{s_t}, U_i, S_i)$$

where S_i' is the future state sequence predicted by $G(\mathbf{s_t}, U_i; \theta)$, S_i is the real state sequence and m is the batch size. Based on this adversarial framework, we outline the training phase of our proposed algorithm to approximate the dynamics model in Algorithm. 1. For more training tricks and hyperparameters utilized in this paper to enhance training stability and streamline the training process, please refer to WGAN [31].

C. MPC Design

In the previous subsection, we showed the SMS framework, and an adversarial network model. In this subsection, we provide a brief discussion on the MPC design.

Basically, our simultaneous predictive process and adversarial model can be used with multiple sampling policies such as MPPI [32], CEM [33], and random policy. In this paper, we use CEM as the sampling policy. This policy can adjust the sampling strategy based on the optimal actions to obtain better sampling results.

As previously mentioned, the objective of the MPC process is to find a set of control actions U^* that optimizes the plant behavior over a given prediction horizon N based on the current state and a given loss. The loss is defined based on tasks and used to show the effect of the controller completing the task. Therefore, we define the corresponding cost function for the path-following task as

$$L_U = (S_{tar} - S')^T Q (S_{tar} - S') + U^T R U$$

where S' is predicted future states based on action sequence U, S_{tar} is the target state sequence, Q and R are information matrices, where Q is positive semidefinite, R is positive-definite. The optimal action selection process is illustrated in Algorithm. 2 and Fig. 2 shows our SMS-MPC structure. The control horizon is set to two to reduce the number of optimizations required.

V. EXPERIMENTAL SETUP

This section outlines the platform we used in this work, the details of parameters in SMS-MPC and how we collected data to train the adversarial dynamics model.

Algorithm 2 SMS-MPC

```
1:  Initialize G net and D net with trained parameters, prediction horizon $N$, maximum iteration $K$, hyperparameters $Q,R$.
2:  Initialize target trajectory $S_{tar} = \{s_1, s_2, ...\}$.
3:  while not reach the final target do
4:      Update current state $s_t$
5:      Determine next target state sequence $S_{tar}$ of length $N$ based on current state $s_t$
6:      for i=1,...,K do
7:          Sample action action sequence $U_i = \{u_1^i, ..., u_N^i\}$ by CEM strategy
8:          Stack $s_t$ for $N$ times to form vector $S_{base}$
9:          Predict state sequence $S_i' = G_\theta(s_t, U_i) + S_{base}$
10:         Calculate cost $L_i = (S_{tar} - S_i')^T Q (S_{tar} - S_i')$
            $+ U_i^T R U_i$
11:     end for
12:     Compute $U^* = argmin_{U_i, i \in [1,K]}(L_i)$
13:     Set $u_t$ to the first two actions of $U^*$, and execute $u_t$.
14: end while
```

A. Platform

The platform used in this work, is a custom-built differential steering ground robot (weight 68 kg; LWH 0.93m×0.7m×1.5m), as shown in Fig. 4. The ground robot has four states: translational positions, translational velocities, attitudes, and rotational velocities. We use some of these data for control, containing horizontal position, orientation, linear velocity, and angular velocity.

The vehicle chassis can return the linear velocity and angular velocity of the vehicle and subscribe control commands by CAN bus. The chassis is also equipped with an industrial computer, which installed an Intel i7-9700E CPU, 16GB RAM, and an outdoor power bank with 150000 mAh. As for sensors, two Velodyne lidar of VLP-32C and VLP-16, two FLIR Blackfly S camera, a Bumblebee2 stereo vision cameras, a NovAtel PwrPak7D-E2 GNSS-inertial integration system were instrumented. In this work, we only use GPS, inertial measurement unit (IMU), and states return from chassis for accurate position and orientation estimation. We utilize real-time kinematic (RTK) technology to provide positioning information with centimeter-level accuracy. When combined with IMU, the rate of position data can reach 50Hz, and the accuracy of it is about 10 cm.

The software system was developed based on the Robot Operating System (ROS) in Ubuntu. Industrial computer and chassis communicate via CAN bus. We built a simulation environment to evaluate the performance of the system before real track tests, as shown in Fig. 4. In addition, the vehicle chassis contains an RC transmitter which can be used to remotely control the vehicle by an operator. The RC transmitter has the highest priority and can interrupt CAN bus messages to disable all motions in case of emergency. In the real-world experiments, all computation was executed on-board the vehicle in real-time.

B. Dataset

Our work conducts experiments on two datasets. The dataset collected in the Gazebo simulator environment contains trajectory samples with a total length of 39.8km. The real environment dataset contains trajectory samples with a total length of 12.1km. Each sample contains positioning information, wheel speed readings, and the corresponding steering and throttle commands. In real world, the vehicle states are measured using on-board IMU and GPS, while in the simulator environment the data is the ground truth obtained directly from the simulator. During the data collection process, the vehicle is operated by actions sampled from different distributions. The sampled actions are designed to cover all possible action spaces as much as possible. Specifically, the linear velocity command is sampled from a normal distribution, with mean = 0.7 m/s, var = 0.1 and resolution=0.01 m/s to cover more data and increase the robustness. The angular velocity command is generated by uniform random sampling with a range of -0.2 rad/s and 0.2 rad/s and at an interval of 0.01 rad/s.

C. Adversarial Dynamics Model Setup

The adversarial dynamics model contains two parts, the generator is set with a four-layer fully connected network. The neurons of each layer are 128, 256, 256 and 128. the discriminator has the same network structure as the generator.

The current state and a sequence of actions with the length of prediction horizon are fed into the generator, where the current state contains coordinates, orientation, angular velocity, linear velocity. To accelerate the training process, all data are normalized. The output δS of the dynamics model is a sequence of changes of vehicle coordinates ,orientation and velocity corresponding to the input state. The discriminator takes current state with actions and predicted states as input. The output of the discriminator is the probability that the input future state sequence belongs to the real trajectory. The discriminator will provide additional loss to training the generator as described in Section IV-B.

VI. EXPERIMENTS

To demonstrate the effectiveness of the proposed algorithm, we tested our algorithm on four aspects. First, the time consumption of SMS framework is compared with previous works. Second, the accuracy of adversarial model is compared with previous works. Third, the functionality of each module is verified by an ablation study. Finally, the effectiveness of SMS-MPC is further demonstrated in the simulation environment and on a real ground robot.

A. Model Accuracy and Speed Comparison

In this paper, the advantages of SMS-MPC are demonstrated by the time consumption to complete the prediction process and the accuracy of prediction. In addition, we further verify the advantages of the adversarial model by conducting an ablation study. The experiments were performed in Gym [34], Gazebo simulator, and a real mobile robot.

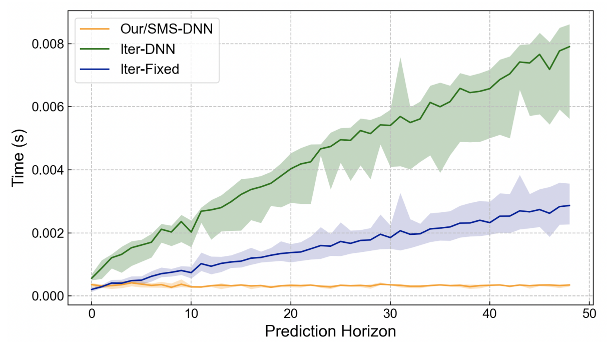

Fig. 5. The time taken by the methods in the SMS framework to complete the prediction does not increase as the prediction length increases, indicating that the time complexity of the multi-step prediction process decreases from $O(N)$ to $O(1)$ in the SMS framework compared to the iterative framework.

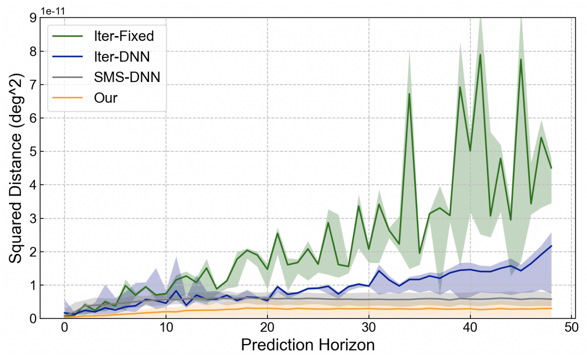

Fig. 6. The accuracy comparison between Iter-DNN and SMS-DNN shows that the SMS framework can alleviate the cumulative error problem. The comparison between SMS-DNN and our method shows that the discriminator provides additional constraints and information to improve the training effectiveness. In addition, our adversarial model has advantages in accuracy over the classical fixed model.

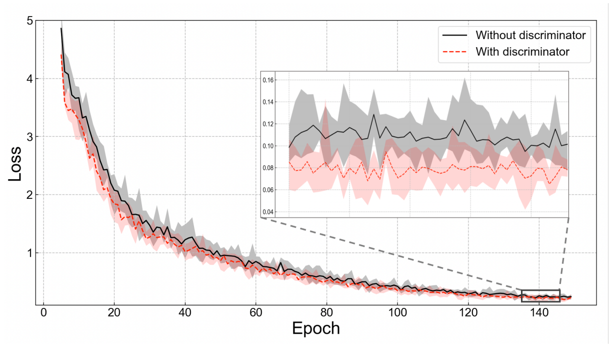

Fig. 7. The training process shows that the loss decreases faster and better after adding the discriminator.

The Gym environment contains two scenarios CartPole and Pendulum. To be persuasive, we analyze the experimental results on the real mobile robot. The detailed data can be found in Table. I, where the column Time in the table indicates the time taken to complete the multi-step prediction process, and the column Accuracy indicates the rate of predictions that lie within a certain threshold of ground truth. The detailed data shows the advantages of our method in terms of time and accuracy.

The dynamics models used for experiments include fixed [24], DNN [20], and adversarial models. The fixed method represents the traditional modeling approach. In Gym environment, the fixed model is a ground truth model obtained according to the simulator. In the Gazebo simulator and real environments, the fixed models are empirically adjusted classical linear models. The DNN represents the previous learning-based approach and uses the same four-layer fully connected neural network as in the generator and discriminator. The RNN method contains loops that can only be used in an iterative framework, and its time complexity cannot be satisfied on our mobile robot, so it's not included in the comparison. All models are trained and tested on the same Intel Xeon E5-2650 v4 processor and Nvidia 2080Ti with a batch size of 128, full precision (fp32).

As shown in Fig. 5, when using the DNN model as dynamics model in the iterative framework (Iter-DNN), it consumes more time compared to the fixed model (Iter-Fixed), but when it is used in the SMS framework (SMS-DNN), the time consumption no longer grows linearly with prediction length. This means the time complexity is reduced from $O(N)$ to $O(1)$. Since our adversarial model uses the same network structure as the DNN model in SMS-SNN, both take the same time consumption. A comparison of the prediction accuracy calculated based on the euclidean distance between each point of the predicted path and the real point is shown in Fig. 6. Specifically, we sample different initial states and a sequence of actions in test data, then feed data to different models and calculate the deviations of predicted results from the ground truth. From the results, we can see that the learning-based DNN model is more advantageous than the fixed model in iterative framework, but when the DNN model is used directly in the SMS framework the accuracy is reduced due to the more complex model that needs to be learned. The accuracy is improved after adding the discriminator to form an adversarial model, indicating that the discriminator can improve accuracy.

B. Ablation Study

To demonstrate the advantage of discriminator in the adversarial model, we conducted an ablation study. As shown in Fig. 7, the adversarial model with discriminator has a faster and lower loss reduction during the training phase compared to the one without discriminators. This demonstrates that the discriminator has the potential to provide additional constraints and information that are hard to include by manually designed loss. In addition, As shown in Fig. 6, the predictions are more accurate after adding discriminator to DNN, indicating that the discriminators can help the model learn more constraints that are hard to describe by manually defined loss.

C. Path Tracking Task

The objective of the path tracking task is to design a control system such that the center of mass of the vehicle tracks desired trajectories S_{tar}. In this work, the desired

TABLE I

DETAILED COMPARISON DATA.

Env	Methods	Accuracy	Time[ms]
CartPole	Iter-Fixed	1.0±0.0	6.7±0.1
	Iter-DNN	0.92±0.03	7.4±0.08
	SMS-DNN	0.85±0.08	3.2±0.12
	Our	**0.95±0.01**	**3.2±0.11**
Pendulum	Iter-Fixed	1.0±0.0	7.3±0.03
	Iter-DNN	0.91±0.02	7.9±0.02
	SMS-DNN	0.85±0.03	5.1±0.03
	Our	**0.92±0.02**	**5.1±0.02**
Gazebo	Iter-Fixed	0.89±0.04	79±0.7
	Iter-DNN	0.86±0.03	84±0.8
	SMS-DNN	0.83±0.07	59±0.8
	Our	**91±0.05**	**59±0.9**
Real	Iter-Fixed	0.87±0.09	71±1.9
	Iter-DNN	0.81±0.07	78±2.0
	SMS-DNN	0.77±0.10	55±1.9
	Our	**0.88±0.06**	**54±2.1**

trajectories are collected by recording the state of the vehicle that manually driven. The linear velocity along the desired trajectory is set to 0.7 m/s, leaving only the angular velocity command for the MPC algorithm to optimize.

The performance index of the task was formulated as the root-mean-square (RMS) error of N pairs of (x,y)-coordinates, o-orientation, between the desired trajectory S_{tar} and the observed trajectory S_{obs}, that sampled at 10 Hz. Considering we only execute the first few actions, the actions obtained several steps later are less important. So we add a discount λ to reduce the weight of the loss as the prediction steps grow. We also add an action punishment to avoid inactive actions, then we can define the cost function to be

$$c(S,U) = \alpha \frac{1}{N} \sum_{i=1}^{N} \lambda^i ||R \odot (\mathbf{s_i^{obs}} - \mathbf{s_i^{tar}})||^2 + \beta U'$$

in which U' is the sum of the absolute values of all elements in action sequence U. R is a coefficient vector, which is used to reduce the weight of orientation error. This coefficient can allow the car to steer at a large angle which can avoid converging to a local optimum.

D. Simulation Experiment

SMS-MPC is first deployed in a simulator environment to evaluate different parameter settings and test the ability to resist noise or outliers of sensor data. The environment is a road with sandy surface built in the Gazebo simulation, and the goal is to complete the path tracking task. We set fixed trajectories for the car to follow and use the same model as the mobile robots in real-world, as shown in Fig. 4. In the training phase, the adversarial model will be trained based on the previously introduced data. In the testing phase, the robustness of the learned model is tested by adding noise to the sensor data. The deviations of the mobile robot using different methods to follow a given path are summarized in Fig. 8. The attached video has more experimental details.

E. Experiments in the Real World

Finally, the entire SMS-MPC was implemented on the custom-built ground robot. The car drives in an outdoor area and the desired 110 meters long path includes slopes, dusty ground, and grass surfaces. During the test, the linear velocity speed is set to a fixed value 0.7 m/s, and the angular velocity speed is controlled by SMS-MPC.

The controller runs at 10Hz, GNSS/IMU integrated system runs at 50Hz. Other relevant parameters are listed in Table. II. We counted the sum of cost prediction error over the prediction horizon at each sampling time. The results are shown in Fig. 9. SMS-MPC can run in different environments and meet the time limit for running on mobile robots, see the video for more details. In addition, due to the rapidity of the SMS framework and the powerful expressiveness of the adversarial model, SMS-MPC shows great potential in unstructured environments.

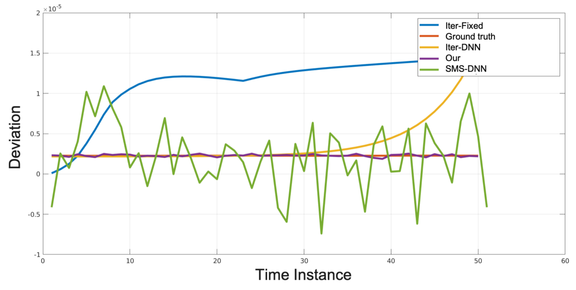

Fig. 8. This figure shows the deviation of the mobile robot from the desired path in Gazebo simulation after adding noise in sensor data that feed to the dynamics model. Methods based on the SMS framework perform better than methods based on the iterative framework, which means SMS framework can help the model to learn the context-information which can alleviate the effects of noise. In addition, the discriminator in the adversarial model can help to learn the potential information contained in states and perform better than other methods, while the fixed model is the most affected.

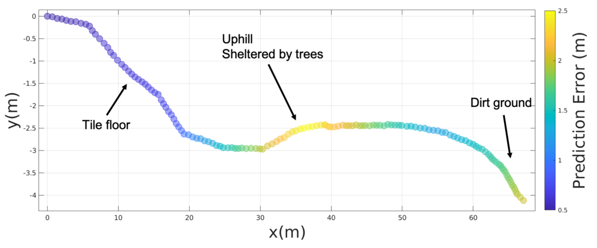

Fig. 9. Prediction error along the path (in meters) is marked with different colors. The prediction error becomes larger when the robot passes through the area covered by trees which can affect the GPS signal. Also, the actual dynamics model becomes more complex when the robot travels on the dirt ground. As a result, the prediction error becomes larger.

TABLE II

PARAMETERS OF SMS-MPC PARAMETERS

Parameter	Setting
Prediction horizon N	30
Linear velocity	0.7 m/s
Angular velocity bound	[-0.2, 0.2] rad/s
λ	0.95
R	[0.1,0.1,0.01]
Control horizon	2

VII. CONCLUSION

In this paper, we propose SMS-MPC, which contains two main innovations. The SMS framework can reduce the time complexity of the multi-step prediction process in MPC from O(N) to O(1), reduce the number of required dynamics models and alleviate the compounding-error problem. Based on this framework, we employ adversarial theory to build a dynamics model, which can obtain additional constraints beyond the hand-crafted loss to improve the training results. In unstructured environments, we demonstrated in simulation and a real mobile robot that this combination can improve the accuracy and robustness of MPC and finally improve performance for mobile robot in the path tracking task.

ACKNOWLEDGMENT

This work was supported by National Natural Science Foundation of China under Grant No. 62176250 and No. 62003323, in part by the Innovation Project of State Key Laboratory of Computer Architecture, ICT, CAS under Grant No. CARCH5202 and No. CARCH5406.

REFERENCES

[1] J. Coulson, J. Lygeros, and F. Dörfler, "Data-enabled predictive control: In the shallows of the deepc," in *IEEE European Control Conference*, 2019, pp. 307–312.

[2] 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.

[3] W. Zhang, M. Tognon, L. Ott, R. Siegwart, and J. Nieto, "Active model learning using informative trajectories for improved closed-loop control on real robots," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 4467–4473.

[4] 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.

[5] T. M. Moerland, J. Broekens, and C. M. Jonker, "Model-based reinforcement learning: A survey," *arXiv preprint arXiv:2006.16712*, 2020.

[6] N. Mohajerin and S. L. Waslander, "Multi-step prediction of dynamic systems with recurrent neural networks," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3370–3383, 2019.

[7] C. D. McKinnon and A. P. Schoellig, "Learn fast, forget slow: Safe predictive learning control for systems with unknown and changing dynamics performing repetitive tasks," *IEEE Robotics and Automation Letters*, vol. 4, no. 2, pp. 2180–2187, 2019.

[8] J. Kabzan, M. I. Valls, V. J. Reijgwart, H. F. Hendrikx, C. Ehmke, M. Prajapat, A. Bühler, N. Gosala, M. Gupta, R. Sivanesan, *et al.*, "Amz driverless: The full autonomous racing system," *Journal of Field Robotics*, vol. 37, no. 7, pp. 1267–1294, 2020.

[9] K. Asadi, D. Misra, and M. Littman, "Lipschitz continuity in model-based reinforcement learning," in *International Conference on Machine Learning*, 2018, pp. 264–273.

[10] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 3, pp. 269–296, 2020.

[11] E. Terzi, M. Farina, L. Fagiano, and R. Scattolini, "Robust predictive control with data-based multi-step prediction models," in *European Control Conference*, 2018, pp. 1710–1715.

[12] E. Terzi, L. Fagiano, M. Farina, and R. Scattolini, "Learning-based predictive control for linear systems: A unitary approach," *Automatica*, vol. 108, p. 108473, 2019.

[13] N. Mishra, P. Abbeel, and I. Mordatch, "Prediction and control with temporal segment models," in *International Conference on Machine Learning*, 2017, pp. 2459–2468.

[14] E. Terzi, M. Farina, L. Fagiano, and R. Scattolini, "Robust multi-rate predictive control using multi-step prediction models learned from data," *Automatica*, vol. 136, p. 109852, 2022.

[15] C. Folkestad and J. W. Burdick, "Koopman nmpc: Koopman-based learning and nonlinear model predictive control of control-affine systems," in *IEEE International Conference on Robotics and Automation*, 2021, pp. 7350–7356.

[16] K. Yeom, "Kinematic and dynamic controller design for autonomous driving of car-like mobile robot," *International Journal of Mechanical Engineering and Robotics Research*, vol. 9, no. 7, pp. 1058–1064, 2020.

[17] L. Hewing, E. Arcari, L. P. Fröhlich, and M. N. Zeilinger, "On simulation and trajectory prediction with gaussian process dynamics," in *Learning for Dynamics and Control*, 2020, pp. 424–434.

[18] N. Mohajerin, M. Mozifian, and S. Waslander, "Deep learning a quadrotor dynamic model for multi-step prediction," in *IEEE International Conference on Robotics and Automation*, 2018, pp. 2454–2459.

[19] S. Lefevre, A. Carvalho, and F. Borrelli, "Autonomous car following: A learning-based approach," in *IEEE Intelligent Vehicles Symposium*, 2015, pp. 920–926.

[20] Q. Li, J. Qian, Z. Zhu, X. Bao, M. K. Helwa, and A. P. Schoellig, "Deep neural networks for improved, impromptu trajectory tracking of quadrotors," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 5183–5189.

[21] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*, 2020, pp. 1101–1112.

[22] K. Chua, R. Calandra, R. McAllister, and S. Levine, "Deep reinforcement learning in a handful of trials using probabilistic dynamics models," *Advances in neural information processing systems*, vol. 31, 2018.

[23] M. A. Kamel, A. T. Hafez, and X. Yu, "A review on motion control of unmanned ground and aerial vehicles based on model predictive control techniques," *Journal of Engineering Science and Military Technologies*, vol. 2, no. 1, pp. 10–23, 2018.

[24] J. B. Rawlings, D. Q. Mayne, and M. Diehl, *Model predictive control: theory, computation, and design*. Nob Hill Publishing Madison, 2017, vol. 2.

[25] Y. Zhang, G. Tao, and M. Chen, "Adaptive neural network based control of noncanonical nonlinear systems," *IEEE transactions on neural networks and learning systems*, vol. 27, no. 9, pp. 1864–1877, 2015.

[26] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman, "Cautious adaptation for reinforcement learning in safety-critical settings," in *International Conference on Machine Learning*, 2020, pp. 11 055–11 065.

[27] C. D. McKinnon and A. P. Schoellig, "Context-aware cost shaping to reduce the impact of model error in receding horizon control," in *IEEE International Conference on Robotics and Automation*, 2020, pp. 2386–2392.

[28] J. Xu, X. Ren, J. Lin, and X. Sun, "Diversity-promoting gan: A cross-entropy based generative adversarial network for diversified text generation," in *Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 3940–3949.

[29] M. Janner, I. Mordatch, and S. Levine, "gamma-models: Generative temporal difference learning for infinite-horizon prediction," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1724–1735, 2020.

[30] D. Silver, J. A. Bagnell, and A. Stentz, "Applied imitation learning for autonomous navigation in complex natural terrain," in *Field and Service Robotics*, 2010, pp. 249–259.

[31] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein generative adversarial networks," in *International conference on machine learning*. PMLR, 2017, pp. 214–223.

[32] G. Williams, N. Wagener, B. Goldfain, P. Drews, J. M. Rehg, B. Boots, and E. A. Theodorou, "Information theoretic mpc for model-based reinforcement learning," in *IEEE International Conference on Robotics and Automation*, 2017, pp. 1714–1721.

[33] A. Nagabandi, K. Konolige, S. Levine, and V. Kumar, "Deep dynamics models for learning dexterous manipulation," in *Conference on Robot Learning*, 2020, pp. 1101–1112.

[34] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.