A Learning-Based Model Predictive Trajectory Planning Controller for Automated Driving in Unstructured Dynamic Environments

Zhiyuan Li, Pan Zhao, Chunmao Jiang, Weixin Huang, and Huawei Liang

***Abstract*—This paper presents a learning-based model predictive trajectory planning controller for automated driving in unstructured, dynamic environments with obstacle avoidance. We first address the problem of lacking prior knowledge in unstructured environments by introducing a risk map that maps the density and motion of obstacles and the road to an occupancy risk. Model predictive control is then used to integrate trajectory planning and tracking control into one framework to bridge the gap between planning and control. Meanwhile, we use Gaussian Process (GP) regression to learn the residual model uncertainty for improving the model accuracy. An objective function considering both risks within the feasible region and vehicle dynamics is carefully formulated to obtain collision-free and kinematically-feasible local trajectories. Field experiments are performed on real unstructured environments with our automated vehicle. Experimental results demonstrate the effectiveness of the proposed algorithm for successful obstacle avoidance in various complex unstructured scenarios.**

***Index Terms*—Trajectory planning, automated vehicles, gaussian processes, model predictive control, unstructured environments, risk map.**

I. Introduction

AUTOMATED vehicles (AVs) have great potential to improve driving safety, reduce traffic congestion, and provide greater mobility, which has drawn significant attention in academia and industry [1]–[3]. With the significant advances in sensors, computer hardware, software, and other technologies in the last decade, automated driving technology has achieved remarkable progress, which is expected to accelerate the development of intelligent transportation systems.

The architecture of automated driving generally consists of three main layers: perception, planning, and trajectory control [4], and close collaboration between the layers can achieve the goal of automated driving in complex environments. Trajectory planning and tracking control, the most critical of these technologies, can significantly improve the comfort and safety of automated driving. A great deal of research has been carried out on trajectory planning and tracking control for automated driving [4]–[6]. However, most of the publications in this field focus on vehicles operating in highly structured environments such as highways or urban roads. In contrast, automated driving in unstructured, dynamic environments plays a vital role in reducing human resources and improving automation in the military, agriculture, transportation, and other fields [7]. It is not easy to apply these approaches directly from structured environments to unstructured, dynamic environments. The lack of prior knowledge and traffic rules in unstructured environments becomes a complex dynamic environment when dynamic obstacles appear, bringing significant challenges to trajectory planning. Therefore, there is a great potential value in the research of trajectory planning and control in unstructured, dynamic environments.

Trajectory planning and tracking control for AVs are typically treated as two independent problems due to the limited onboard computing resources [4], [8]. Most recently, some researchers have integrated trajectory planning and tracking control into one framework to bridge the gap between planning and control, and they have achieved remarkable achievements [9]–[12]. However, these methods are mainly studied in structured environments, with little research in unstructured, dynamic environments due to the complexity. In general, the challenges of developing a stable and reliable trajectory planning and tracking control framework in unstructured, dynamic environments lie in the following factors: 1) unstructured environments lack prior knowledge, such as traffic rules, lane markings, and road boundaries; 2) unstructured environments have high uncertainty, including inaccurate obstacle locations and irregular movements of dynamic obstacles; 3) unstructured roads are typically rough, and the complicated interaction between tires and roads makes it difficult to model the vehicle dynamics accurately.

Manuscript received June 18, 2021; revised December 26, 2021; accepted March 11, 2022. Date of publication March 16, 2022; date of current version June 24, 2022. This work was supported in part by the National Key Research and Development Program of China under Grants 2020AAA0108103, 2016YFD0701401, 2017YFD0700303, and 2018YFD0700602, in part by the Youth Innovation Promotion Association of the Chinese Academy of Sciences under Grant 2017488, in part by the Key Supported Project in the Thirteenth Five-year Plan of Hefei Institutes of Physical Science, Chinese Academy of Sciences under Grant KP-2019-16, in part by the Natural Science Foundation of Anhui Province under Grant 1508085MF133, and in part by the Technological Innovation Project for New Energy and Intelligent Networked Automobile Industry of Anhui Province. The review of this article was coordinated by Prof. Jun Won Choi. *(Corresponding author: Huawei Liang.)*

Zhiyuan Li, Chunmao Jiang, and Weixin Huang are with the Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China, and also with the University of Science and Technology of China, Hefei 230026, China (e-mail: zyli23@mail.ustc.edu.cn; sa252@mail.ustc.edu.cn; hwx2018@mail.ustc.edu.cn).

Pan Zhao and Huawei Liang are with the Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei 230031, China, and also with the Anhui Engineering Laboratory for Intelligent Driving Technology and Application, Hefei 230031, China, and also with the Innovation Research Institute of Robotics and Intelligent Manufacturing, Chinese Academy of Sciences, Hefei 230031, China (e-mail: pzhao@hfcas.ac.cn; hwliang@iim.ac.cn).

Digital Object Identifier 10.1109/TVT.2022.3159994

Given the above problems, this paper proposes a learning-based model predictive trajectory planning controller for automated driving in unstructured, dynamic environments with obstacle avoidance. First, we use the drivable area extracted by the perception system to obtain prior information such as road width, boundary, and shape. In this way, we can focus the scope of motion planning mainly in the drivable area, which improves safety and reduces the complexity of motion planning. Subsequently, for the uncertainties arising from unstructured environments, we designed a hierarchical risk map to assess the risk of roads and different types of obstacles. Finally, to improve model accuracy in unstructured environments, we use Gaussian process regression to learn the residual model uncertainty online based on the collected historical data. We validated our system both in simulations and on our automated vehicle in real off-road environments. Experimental results demonstrated the vehicle's ability to navigate smooth, collision-free trajectories in dynamic, unstructured environments.

The remainder of the paper is organized as follows: Section II describes the related work. Section III presents the system framework for automated vehicles in unstructured environments and demonstrates the output of the highly reliable perception system. Section IV details the construction method of the risk map and defines the risk for roads and different types of obstacles. Section V presents the nominal vehicle dynamics model, the learning strategy of the residual model, and the trajectory planning controller design. The trajectory planning controller is tested and validated in both simulation and real experiments under complex scenarios in Section VI and Section VII. We conclude the paper in Section VIII.

II. Related Work

Risk assessment is an essential process in the design and development of automated vehicles, and taking risk into consideration in trajectory planning can significantly improve safety during automated driving [13]–[15]. Research [15] proposes a risk assessment method that can efficiently calculate the collision risks for a set of pre-configured local candidate paths along with the lane-based probabilistic motion prediction for surrounding vehicles. This approach allows for real-time estimation of future collision risk associated with surrounding vehicles. However, this method is mainly applied in multiple lanes on structured roads. In [10]–[12], they used the concept of artificial potential fields to establish road and obstacle potential fields around the ego vehicle. Under the repulsive force of the potential field, the ego vehicle enables driving in the middle of the lane and effectively avoids risk. However, they treated both static and dynamic obstacles as a unified class, which inevitably cannot accurately capture the risk caused by the motion trend of dynamic obstacles. In [16], an online Gaussian risk map in an uncertain environment is proposed, which can be updated online based on real-time tracking data of obstacles. The risk map in this approach is constructed separately according to the characteristics of static and dynamic obstacles, which achieved satisfying results on both structured and unstructured roads. However, they cannot address the risk prediction of high-speed dynamic obstacles.

A large number of motion planning algorithms for mobile robots have been proposed and applied in unstructured environments, such as sampling based, graph search based, artificial potential field (APF), and optimal methods [17], [18]. The graph search method discretized the configuration space of mobile robots as a graph, and then searches for a minimum-cost path in such a graph. The A* algorithm [19], D* algorithm [20], Dijkstra algorithm [21] and their improvements are the most common approaches. In a hierarchical, graph-based, multi-resolution terrain model, [22] used a distribution-based binary classifier to reduce the planning search space and combined it with a cost heuristic to accelerate convergence. Experimental results demonstrate the applicability to both structured and unstructured environments. These planners, however, are only applicable to a small range of known environments and cannot directly consider the continuity of curvature. Sampling-based methods seek to plan in high-dimensional spaces that cannot be solved by deterministic methods. Rapidly-exploring Random Tree (RRT) [23] and the Probabilistic Roadmap Method (PRM) [24] are the two most common methods. In [25], they utilized a variable step RRT that is capable of automatically adjusting the step size, eliminating the necessity of tuning the step size for different environments. Based on the Bi-directional RRT, [26] considered the kinematic constraints in the planning process and used an efficient pruning strategy to obtain kinematically connected smooth paths. Although these planners have probabilistic completeness, it is difficult to find the optimal solution in an unstructured environment. The principle of APF is to generate the repulsive potential field to the obstacle and the attractive potential field to the goal, which makes the mobile robot avoid collision with the obstacle while moving toward the goal [10]. However, it tends to fall into local minima in the gradient descent process [27]. Although the methods mentioned above can effectively avoid obstacles in low-speed and low-complexity scenarios, the trajectories are typically more aggressive when approaching the obstacles, which results in a giant swing of the vehicle. More complex and predictable behavior can be achieved by applying the optimal control methods, and our approach relies on Model Predictive Control (MPC) [28]. MPC can obtain collision-free and kinematically-feasible trajectories by incorporating the dynamical constraints of automated vehicles and the predicted behavior of obstacles. Meanwhile, researchers also prefer to use MPC for path tracking control with the improvement of computer performance [29], [30]. In contrast, we integrate trajectory planning and tracking control into one module and use MPC for both trajectory planning and control in complex dynamic environments.

MPC relies heavily on a suitable and accurate representation of the system dynamics model. However, in unstructured environments, the complicated interactions between tires and roads make it hard to model vehicle dynamics accurately. To improve model accuracy, learning-based MPC is generally considered to have great potential as it can predict the system more accurately by seeking to estimate the uncertainty of dynamics in the measured data [31]. GP-based MPC (GP-MPC), one of the most popular methods in learning-based MPC, provides a flexible

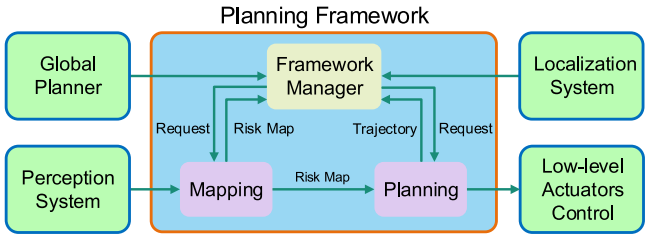


Fig. 1. System framework for AVs.

stochastic nonparametric approach that has been validated in a large number of works [32]–[34]. In [32], [33], they applied GP-MPC to a small high-speed racing car. The GP learns the dynamics of unknown systems from the race car's historical data, which significantly improves the control performance and increases the average speed. [34] utilized GP to achieve optimal control of the mobile robot in off-road terrain, and the controller also achieved high performance and optimal control when the GP reduced the model uncertainty. In this work, we integrated GP-MPC into a trajectory planning and tracking control framework and implemented it on our automated vehicle.

The main contributions of this study can be summarized as follows:

- A practical approach for constructing a hierarchical risk map in unstructured, dynamic environments is proposed to map road risk and obstacle risk, providing helpful prior knowledge for trajectory planning.
- Gaussian process regression is utilized to identify the residual model uncertainty from the vehicle's historical data to correct the vehicle dynamics and improve the model accuracy.
- The method is successfully implemented on our automated vehicle, and the effectiveness of obstacle avoidance is fully verified in an unstructured dynamic environment.

III. System Framework Overview

Due to the lack of prior knowledge of roads such as lane lines and stop lines in unstructured roads and the challenges of rough ground and uncertainty of obstacle locations, the framework of motion planning in urban environments is no longer applicable to unstructured environments. By fully considering the characteristics in an unstructured environment, the system framework, depicted in Fig. 1, has three modules: (i) a mapping module that takes into account the uncertainty features in the environment, (ii) a planning module, which constantly calculates the safe and kinematically-feasible trajectory and the optimal control commands, and (iii) a framework manager that handles cooperation between the various modules. In addition, the system framework also contains the following modules: the global planner, perception system, localization system, and low-level actuator control. The system takes the planning framework as the core, and communication between modules is based on a particular publish/subscribe messaging protocol for Inter-Process Communication.

The perception system receives and processes the raw 3D point cloud information from the sensors and uses a specific

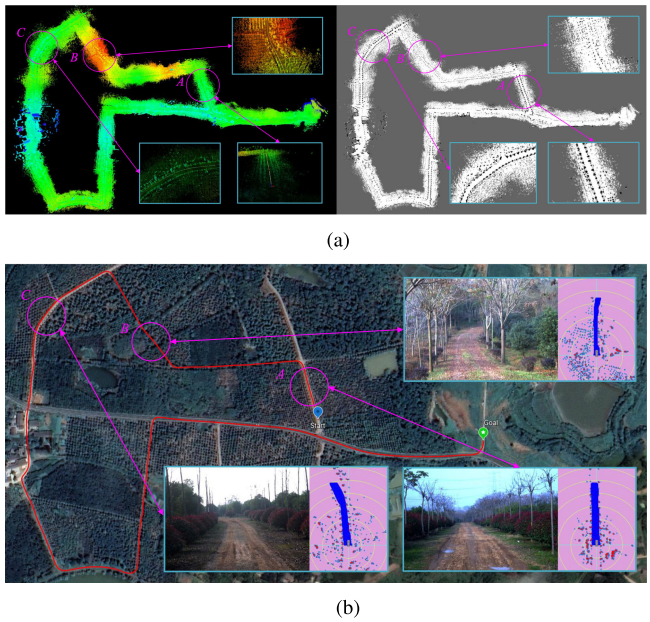


Fig. 2. Illustration of perception and localization systems in unstructured environments. (a) 3D point cloud and 2D occupancy grid map. (b) Aerial view and 2D drivable map.

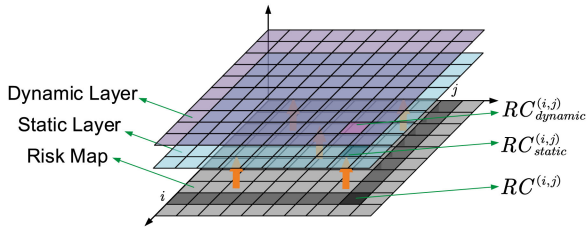


Fig. 3. Illustration of two layers of risk map.

ground segmentation algorithm combined with Simultaneous Localization and Mapping (SLAM) to obtain an occupancy grid map. Finally, the drivable area can be easily extracted based on the current location and the global reference trajectory. The image on the left of Fig. 2(a) is the global 3D point cloud, while the right is the global occupancy grid generated by SLAM. Three common scenarios were selected to demonstrate the validity of the drivable area: Location A with a magenta circle is a wide straight road. At the same time, B is a narrow road and C is a curved road. As shown in Fig. 2(b), the drivable area obtained from the perception system can accurately represent the current driving scenarios.

IV. Risk Map Construction

The main focus of this section is the mapping module, intending to construct a risk map in an unstructured environment by evaluating the drivable area and obstacles. The risk map, which mainly contains the location and risk of obstacles, can provide reliable prior knowledge for the planning module. However, the dynamic changes of environment bring considerable challenges to risk map construction.

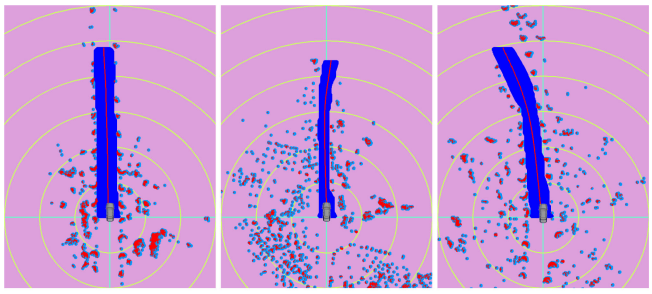


Fig. 4. The extraction results of road center baseline. (a),(b),(c) correspond to the real-time extraction results of the road center baseline at locations A, B, C in Fig. 2.

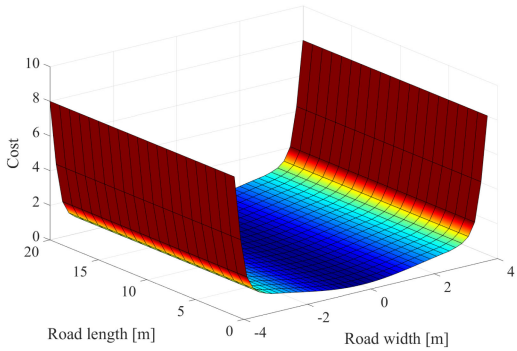


Fig. 5. Illustration of road risk.

To accurately express the risk map, as depicted in Fig. 3, the risk map is divided into two layers: static layer and dynamic layer. The static layer mainly describes the driving cost of the drivable area in the unstructured road. In contrast, the dynamic layer mainly describes the uncertainty cost generated by the static obstacles in the drivable area and the risk generated by the future movement trend of the dynamic obstacles. Consider the risk map $G \subset \mathbb{R}^N$, with grids in G denoted g, and the grid in row i and column j has a risk cost $RC^{(i,j)}$, which consists of static risk cost $RC^{(i,j)}_{static}$ and dynamic risk cost $RC^{(i,j)}_{dynamic}$.

A. Road Risk Construction

The edge of an unstructured road is often rough with many obstacles, which poses a potential danger to vehicle safety and stability. However, the central area of the road is typically flat with few obstacles. Therefore, driving close to the road center baseline is an ideal way to ensure safety. In this way, a potential field with low risk in the center and high risk on both sides can be used to represent the road risk. This section first extracts the road center baseline of the drivable area shown in Fig. 2, and then generates the road risk based on the road center baseline.

1) Road Center Baseline Generation: As depicted in Fig. 2, the drivable area has different shapes under different road scenes, which poses a significant challenge to the construction of road risk. To represent the characteristics of the different drivable areas, we extract the road center baseline from the drivable areas by General Regression Neural Network (GRNN). The results are shown as the red curve in the blue area in Fig. 4, which clearly shows that the road center baseline extracted by GRNN in different scenes can accurately represent the geometry of the road.

2) Road Risk: To keep the ego vehicle within the drivable area, a danger threshold d_{th} against the road edge is defined to distinguish the danger from outside the road. The grid is usually considered safe when it is farther than d_{th} from the edge of the drivable area, and the set of all grids is denoted as $\mathcal{B}$. The risk within $\mathcal{B}$ is denoted as U and is represented by a carefully modified Gaussian function, which models each cross-section of the road sequentially. First, the standard Gaussian function is applied to the grids on both sides of the road, with the road center baseline as the mean. Then the results after adjusting the weights are taken as negative, presenting a shape of low middle and high sides. Finally, the risk is adjusted to an appropriate value by the parameter λ_r, resulting in

$$U = \lambda_r - \alpha_r \exp\left(-(g - x_{near})^T \Sigma^{-1}_{road} (g - x_{near})\right) \quad (1)$$

where x_{near} represents the nearest point to g on the road center baseline, λ_r and α_r are weighting parameters.

Within the danger threshold ($g \notin \mathcal{B}$), we employ a similar form of the repulsive potential, and the road risk RC_{road} is given by Eq. (2).

$$RC_{road} = \begin{cases} U & g \in \mathcal{B} \\ \frac{\beta_r}{[\eta_r(g - x_{edge,j})]^2 + 1} + \tau_r & g \notin \mathcal{B} \end{cases} \quad (2)$$

where β_r is a scaling factor to adjust the maximum cost, η_r is the weight factor for adjusting the change rate of the boundary cost, and τ_r represents the cost adjustment parameter. $x_{edge,j}$ is the j^{th} road edge coordinate, $j \in \{1, 2\}$. Fig. 5 illustrates the results of the road risk construction.

B. Static Obstacles Risk Construction

It is assumed that the perception system can acquire the location of obstacles in real-time, and the tracking module can accurately track each obstacle and output in real-time. In this work, the locations of the obstacles are represented in the SL coordinate system. The coordinates in the SL coordinate system are composed of station S (distance along the global path) and lateral coordinate L (distance of a point from the global path, perpendicular to the global path). We consider that the drivable area consists of a set of N obstacles $\{\mathcal{O}^1, \mathcal{O}^2, \ldots, \mathcal{O}^N\}$. For $\mathcal{O}^i$, the real location and the measurements obtained by the sensors are denoted as x^i and z^i, respectively. Assuming that x^i is a Gaussian process (GP) with known variance and the mean of the GP is the exact value of the obstacle location, i.e. $x^i \sim \mathcal{N}(x^i_0, \Sigma^i_0)$. Considering the uncertainties of the sensor, we presume that the sensor's noise ε follows a Gaussian distribution, i.e. $\varepsilon \sim \mathcal{N}(0, \Sigma_R)$.

For obstacle $\mathcal{O}^i$, the streamline output $\{z^i_{t-K+1}, z^i_{t-K+2}, \ldots, z^i_t\}$ of the tracking module from time $t - K + 1$ to time t, which we define as $z^i_{t-K+1:t}$. Based on the outputs of the previous K steps, the posterior distribution of $\mathcal{O}^i$ at time t is defined as $P(x^i | z^i_{t-K+1:t})$, which follows a Gaussian

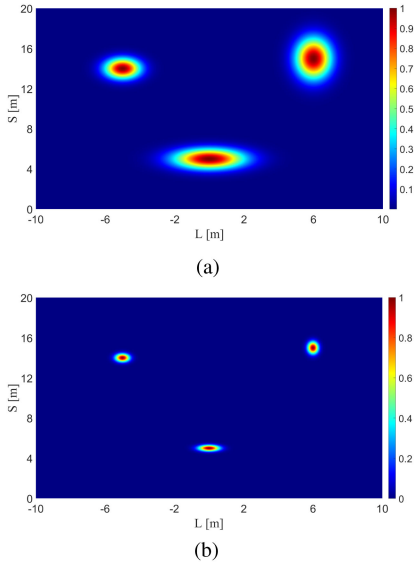


Fig. 6. (a) Risk generated by static obstacles with initial belief. (b) Risk generated by static obstacles after 50-step update.

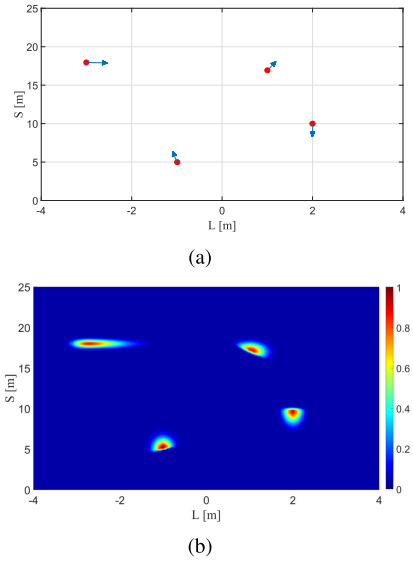


Fig. 7. (a) The red point indicates the location of low-speed dynamic obstacle, while the blue line indicates the velocity direction. (b) Risk generated by low-speed dynamic obstacles.

distribution, i.e. $P(x^i|z^i_{t-K+1:t}) \propto \mathcal{N}(x^i_t, \Sigma^i_t)$. x^i_t and Σ^i_t can be calculated by Bayesian inference, and the specific derivation procedure is available in [16]. In addition, the transfer between states is assumed to follow a Markov process, the online update process for x^i_t and Σ^i_t is given by

$$x^i_t = \left(\left(\Sigma^i_{t-1}\right)^{-1} + \left(\Sigma^i_R\right)^{-1}\right)^{-1}\left(\left(\Sigma^i_{t-1}\right)^{-1} x^i_{t-1} + \Sigma^{-1}_R z^i_t\right) \tag{3a}$$

$$\Sigma^i_t = \left(\left(\Sigma^i_{t-1}\right)^{-1} + \left(\Sigma^i_R\right)^{-1}\right)^{-1} \tag{3b}$$

The mean and variance of each static obstacle at the current time can be calculated by Eq. (3), and Eq. (4) is used to generate the risk of static obstacles. Note that for clarity, we use $x_{sob,i}$ and $\Sigma_{sob,i}$ to replace x^i_t and Σ^i_t, respectively.

$$RC_{sob} = \sum_{i=1}^{N} \beta_i exp\left(-\left(g - x_{sob,i}\right)^T \Sigma^{-1}_{sob,i}\left(g - x_{sob,i}\right)\right) \tag{4}$$

where $x_{sob,i} = [l_i, s_i]^T$, $\Sigma_{sob,i} = diag\{[\sigma^2_{l,i}, \sigma^2_{s,i}]\}$.

Fig. 6 illustrates the risk map update process for static obstacles.

C. Dynamic Obstacles Risk Construction

Since dynamic obstacles have high uncertainty and different types of dynamic obstacles have different motion characteristics, they cannot be grouped to predict their future movements. In this paper, dynamic obstacles are divided into high-speed dynamic obstacles (such as cars) and low-speed dynamic obstacles (such as pedestrians). A dynamic obstacle is considered as a high-speed dynamic obstacle when the instantaneous velocity of successive k frames is higher than the dynamic velocity threshold v^{dy}_{th} or the average velocity is higher than v^{dy}_{th}. In addition, when the vision detection system successfully identified some types of dynamic obstacles, such as motorcycles, cars, and agricultural tractors, we also treated them as high-speed dynamic obstacles even if their velocities were lower than v^{dy}_{th} because of their potential ability to accelerate suddenly. Consequently, other dynamic obstacles are considered as low-speed dynamic obstacles. In this work, we use two different methods to calculate their respective risk.

1) Low-speed Obstacles: Since the motion of low-speed dynamic obstacles is irregular, the corresponding risk is established based on its real-time location and velocity in the SL coordinate system. To simplify the calculation, we use the mean and variance update strategy of static obstacles to estimate the real-time location of low-speed dynamic obstacles, and the velocity is given by the tracking module, resulting in

$$RC_{ldob} = \sum_{i=1}^{m} \frac{exp\left(-\left(g - Ax_{ldob,i}\right)^T \Sigma^{-1}_{ldob,i}\left(g - Ax_{ldob,i}\right)\right)}{1 + exp\left(-\lambda_i x^T_{ldob,i} \mathrm{B}\left(g - Ax_{ldob,i}\right)\right)} \tag{5}$$

where $x_{ldob,i} = [l_i, s_i, v_{l,i}, v_{s,i}]^T$, $A = \begin{bmatrix}1&0&0&0\\0&1&0&0\end{bmatrix}$, $B = \begin{bmatrix}0&0&1&0\\0&0&0&1\end{bmatrix}^T$, $\Sigma_{ldob,i} = diag\{[\sigma^2_{l,i}, \sigma^2_{s,i}]\}$.

Fig. 7 illustrates the mapping from locations and velocities of low-speed obstacles to the risk map.

2) High-speed Obstacles: For high-speed dynamic obstacles, this paper will use the LSTM (long short-term memory) neural network to learn continuous features of historical trajectories to obtain future prediction trajectories and then generate the corresponding risk according to the prediction trajectories.

LSTM is a specific form of RNN (recurrent neural network) that makes the weights of the self-loops variable by adding the input gate, forget gate, and output gate. In this way, the scale

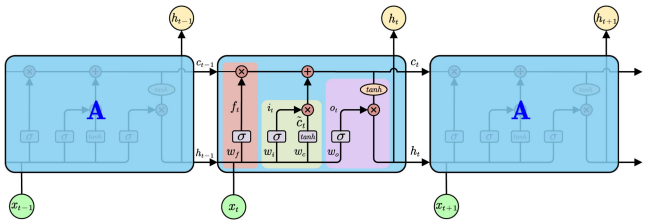


Fig. 8. The internal structure of an LSTM cell.

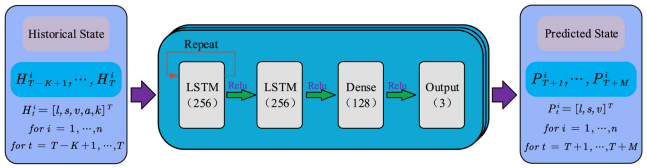


Fig. 9. Network structure of trajectory prediction.

of integration can be changed dynamically at different moments with fixed model parameters, thus avoiding gradient disappearance or gradient expansion. The specific network structure is shown in Fig. 8, and the following recursive equation represents the calculation process

$$f_t = \sigma\left(w_f s_t + b_f\right) \tag{6a}$$

$$i_t = \sigma\left(w_i s_t + b_i\right) \tag{6b}$$

$$\tilde{c}_t = \tan h\left(w_c s_t + b_c\right) \tag{6c}$$

$$c_t = \tilde{c}_t \odot i_t + c_{t-1} \odot f_t \tag{6d}$$

$$o_t = \sigma\left(w_o s_t + b_o\right) \tag{6e}$$

$$h_t = o_t \odot \tan h\left(c_t\right) \tag{6f}$$

where $s_t = [h_{t-1}, x_t]$, h_t and x_t are the output vector and the input vector respectively, i_t, f_t and $\tilde{c}_t$ are gate vectors, $\sigma(\cdot)$ represents the activation function, w denotes the linear transformation matrix, b denotes the offset vector, c_t denotes the amount of cell memory.

The detailed network structure for trajectory prediction is shown in Fig. 9, which consists of two layers containing 256 LSTM cells, one fully connected layer containing 128 neurons, and a dense output layer containing the same number of cells as the output vector. The network receives the historical trajectory, velocity, acceleration, and curvature from the dynamic obstacles and outputs the future location and velocity of the obstacles. Since the inputs to the network are the deviations from the reference trajectory, it enables the network to learn the continuous features of the historical trajectory. In addition, as the network is trained with relative coordinates, it can improve data sparsity and enhance scene adaptation.

Since the accuracy of the predicted trajectory decreases with the predicted distance, the uncertainty will be higher where the predicted trajectory is far from the dynamic obstacle. To simplify the model and reduce the computational complexity, we assumed that the variance approximately linearly varies along the trajectory. We first define the variances of the start and end of the trajectory based on the actual location of the dynamic

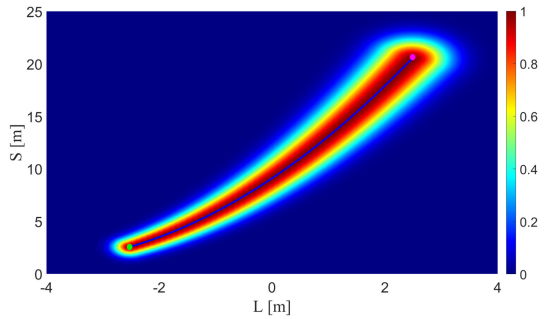


Fig. 10. Risk generated by high-speed dynamic obstacles.

obstacle and the predicted length. Then we calculate the risk cost of each discrete predicted trajectory waypoint using Eq.(5) based on the principle of linear variation of variance, and finally updated the risk map based on the principle of maximum risk cost.

We set up a prediction trajectory and generate the corresponding risk by the above algorithm, as shown in Fig. 10. The blue curve is the predicted trajectory, the green point is the start of the predicted trajectory, while the purple point is the end. It can be seen from the figure that the above algorithm can approximately represent the risk of high-speed dynamic obstacles in the future.

D. Update the Risk Map

Each grid in the risk map has a corresponding risk cost $RC^{(i,j)}$, combining the risk cost of road $RC^{(i,j)}_{road}$, static obstacles $RC^{(i,j)}_{sob}$, low-speed dynamic obstacles $RC^{(i,j)}_{ldob}$ and the high-speed dynamic obstacles $RC^{(i,j)}_{hdob}$. The calculation method of $RC^{(i,j)}$ is then given by

$$RC^{(i,j)} = \lambda_{static} RC^{(i,j)}_{static} + \lambda_{dynamic} RC^{(i,j)}_{dynamic} \tag{7}$$

subject to

$$RC^{(i,j)}_{static} = RC^{(i,j)}_{road}$$

$$RC^{(i,j)}_{dynamic} = \max\left\{RC^{(i,j)}_{sob},\ RC^{(i,j)}_{ldob},\ RC^{(i,j)}_{hdob}\right\}$$

where λ_{static} and $\lambda_{dynamic}$ are the weight parameters of road and obstacles, respectively. i denotes the row of the risk map while j denotes the column.

In Section IV-A-(2), we divided the drivable area into two types of area according to the distance from the road boundary. The area in the middle of the road ($g \in \mathcal{B}$) is mainly used to constrain the ego vehicle to drive close to the road center baseline, and the corresponding cost is usually mapped to $[0, \tau_r]$. However, driving close to the road boundary ($g \notin \mathcal{B}$) is more likely to collide with obstacles outside the road and the area is supposed to have a higher risk cost $[\tau_r, \beta_r + \tau_r]$. The maximum road risk cost at the boundary should be close to the maximum cost generated by the obstacles, indicating that the ego vehicle should neither drive beyond the edge of the drivable area nor collide with the obstacles. In this work, the risk cost of obstacles is mapped to $[0, 1]$, and the maximum cost of road risk is determined by the parameters β_r and τ_r.

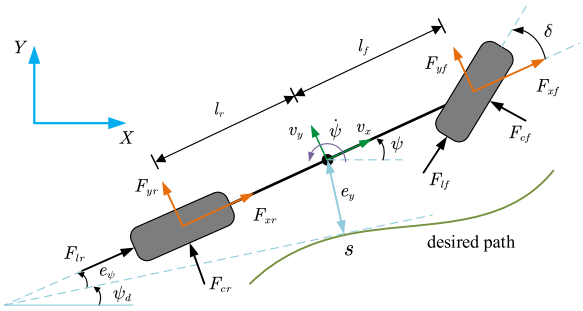


Fig. 11. Schematic of the vehicle model.

The relationship between λ_{static} and $\lambda_{dynamic}$ should satisfy $\lambda_{static}(\beta_r + \tau_r) \approx \lambda_{dynamic}$ in the application.

V. Trajectory Planning

This section begins with a nominal model of the vehicle dynamics, which is mainly based on the material presented in [35]. A learning-based model prediction trajectory planning controller is then established, and the road risk and obstacle risk are added to the controller to enhance the obstacle avoidance capability.

A. Vehicle Dynamics

We consider the following discrete-time model to describe the dynamics of the automated vehicle.

$$\dot{x} = f(x,u) + C\left(g(x,u) + \omega\right) \tag{8}$$

where f is a known nominal model and g describes the unknown dynamics. We assume that both f and g are differentiable functions, and ω follows an independent distributed Gaussian distribution.

A bicycle model is used to describe the nominal dynamics as depicted in Fig. 11, resulting in

$$\dot{v}_x = \frac{1}{m}\left(m v_y \dot{\psi} + 2F_{xf} + 2F_{xr}\right) \tag{9a}$$

$$\dot{v}_y = \frac{1}{m}\left(-m v_x \dot{\psi} + 2F_{yf} + 2F_{yr}\right) \tag{9b}$$

$$\ddot{\psi} = \frac{1}{I_z}\left(2l_f F_{yf} - 2l_r F_{yr}\right) \tag{9c}$$

$$\dot{e}_\psi = \dot{\psi} - \dot{\psi}_d \tag{9d}$$

$$\dot{e}_y = v_y \cos\left(e_\psi\right) + v_x \sin\left(e_\psi\right) \tag{9e}$$

$$\dot{s} = v_x \cos\left(e_\psi\right) - v_y \sin\left(e_\psi\right) \tag{9f}$$

where $x = [v_x, v_y, \dot{\psi}, e_\psi, e_y, s]^T$ is the state of the system. v_x and v_y denote the longitudinal and lateral velocity. $\dot{\psi}$ is the yaw rate around the vehicle's center of gravity (CoG). e_ψ and e_y are the deviation error from the desired path of vehicle's orientation and lateral position, respectively. s is the longitudinal position along the desired path. ψ_d denote the angle of the tangent of the desired path in the fixed coordinate frame. l_r and l_f are the distance from the CoG to the front and the rear axles, respectively. m and I_z are the vehicle's mass and the vehicle's momentum of inertia around its vertical axis. F_{xf} and F_{xr} denote the total longitudinal forces of the front and rear tires, while F_{yf} and F_{yr} represent the lateral forces of the front and rear tires. The tires are modeled by a simplified Pacejka tire model [36], and in this paper, the exact tire model is referred to [35]. From [35], the longitudinal tire forces are linear with the braking/throttle ratio α under the assumption of a small tire slip angle, where $\alpha = 1$ represents maximum throttle, while $\alpha = -1$ represents maximum braking. In addition, we assume that only the steering angle of the front wheels can be controlled and both front wheels have the same steering angle. i.e., $\delta_1 = \delta_2 = \delta$ and $\delta_3 = \delta_4 = 0$. We take α and δ as the control inputs, resulting in: $u = [\alpha, \delta]^T$.

This model is discretized with a fixed sampling time T_s by Euler forward scheme to be utilized as the discrete-time MPC formulation.

$$x\left(k+1\right) = f\left(x(k), u(k)\right) + C\left(g\left(x(k), u(k)\right) + \omega(k)\right) \tag{10}$$

where $x(k) \in \mathbb{R}^{n_x}$ is the state and $u(k) \in \mathbb{R}^{n_u}$ is the input to the system at time step $k \in \mathbb{N}$, $f : \mathbb{R}^{n_x} \times \mathbb{R}^{n_u} \to \mathbb{R}^{n_x}$ is the nominal model, $\omega(k) \sim \mathcal{N}(0, \Sigma_\omega)$ is Gaussian white noise with zero mean and variance Σ_ω.

B. Gaussian Process Regression

Due to AVs needs high real-time performance, the nominal system model is simplified to reduce the complexity of the system. Although the nominal system model is sufficient for the operation of AVs, to achieve higher performance, we use a learning-based method to identify the residual model uncertainty so as to improve the performance and enable the automatic model adaptation.

In the following, we will use GP regression [37] to infer the unknown function $g(x,u)$ based on a training set of previously collected measurements of states x_i and inputs u_i. In this paper, we will use the following notations to describe the current data set as

$$\mathcal{D}=\left\{\boldsymbol{Z}=[z_1,\ldots,z_m]^T \in \mathbb{R}^{m\times n_z}, \boldsymbol{y}=[y_1,\ldots,y_m]^T \in \mathbb{R}^{m\times n_p}\right\}$$

where $z_i = [x_i; u_i]$, $n_z = n_x + n_u$, and we identify an unknown function $g : \mathbb{R}^{n_z} \to \mathbb{R}^{n_p}$ from the current data set $\mathcal{D}$

$$y_i = C^\dagger\left(x_{i+1} - f\left(x_i, u_i\right)\right) = g\left(z_i\right) + \omega_i \tag{11}$$

where $C^\dagger$ is the Moore-Penrose pseudo-inverse, and ω_i is the Gaussian white noise corresponds to the process noise in Eq.(10) with zero mean and diagonal variance $\Sigma_\omega = \text{diag}([\sigma_1^2, \ldots, \sigma_{n_p}^2])$.

In this paper, we assume that each output dimension $p \in \{1, \ldots, n_p\}$ is learned individually from $\boldsymbol{Z}$, and the GP can be written as

$$[\boldsymbol{y}]_{\cdot,p} \sim \mathcal{N}\left(\boldsymbol{0}, K\left(\boldsymbol{Z}, \boldsymbol{Z}\right)\right) \tag{12}$$

where $[\boldsymbol{y}]_{\cdot,i}$ is the ith column of matrix $\boldsymbol{y}$, $K(\boldsymbol{Z}, \boldsymbol{Z})$ is the Gram matrix using a kernel function $k(\cdot,\cdot)$ on the input $\boldsymbol{Z}$, i.e.

$[K(\boldsymbol{Z},\boldsymbol{Z})]_{i,j}=k(z_i,z_j)$. We use the popular squared exponential kernel throughout this paper

$$k\left(z_i,z_j\right)=\sigma_f^2\exp\left(-\frac{1}{2}\left(z_i-z_j\right)^T\Lambda^{-1}\left(z_i-z_j\right)\right) \tag{13}$$

where σ_f^2 is the signal variance and $\Lambda\in\mathbb{R}^{n_z\times n_z}$ is a positive diagonal length scale matrix.

With the combination of the data set $\mathcal{D}$ and the test input z^*, we can get the joint distribution in output dimension p as

$$\begin{bmatrix}[\boldsymbol{y}]_{\cdot,p}\\ [y^*]_p\end{bmatrix}\sim\mathcal{N}\left(\boldsymbol{0},\begin{bmatrix}K\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I} & K\left(\boldsymbol{Z},z^*\right)\\ K\left(z^*,\boldsymbol{Z}\right) & K\left(z^*,z^*\right)\end{bmatrix}\right) \tag{14}$$

where y^* is the predicted output, $[K(\boldsymbol{Z},z^*)]_i=k(z_i,z^*)$, $K(z^*,\boldsymbol{Z})=[K(\boldsymbol{Z},z^*)]^T$, $K(z^*,z^*)=k(z^*,z^*)$.

Given the training data $\mathcal{D}$ and the conditional distribution

$$[y^*]_p\left|\boldsymbol{Z},[\boldsymbol{y}]_{\cdot,p},z^*\sim\mathcal{N}\left(m_p^r(z),\Sigma_p^r(z)\right)\right. \tag{15}$$

where

$$m_p^r(z)=K\left(z^*,\boldsymbol{Z}\right)\left(K\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I}\right)^{-1}[\boldsymbol{y}]_{\cdot,p} \tag{16a}$$

$$\Sigma_p^r(z)=K\left(z^*,z^*\right)-K\left(z^*,\boldsymbol{Z}\right)\left(K\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I}\right)^{-1}K\left(\boldsymbol{Z},z^*\right) \tag{16b}$$

Combining the individual output dimensions and the resulting multivariate GP approximation of $g(z)$ is given by

$$r(z)\sim\mathcal{N}\left(m^r(z),\Sigma^r(z)\right) \tag{17}$$

where $m^r(z)=[m_1^r(z);\cdots;m_{n_p}^r(z)]$, $\Sigma^r(z)=\mathrm{diag}([\Sigma_1^r(z);\cdots;\Sigma_{n_p}^r(z)])$.

The hyper-parameters $\theta=(\sigma_f^2,\Lambda,\sigma_p^2)$ are critical in model estimation, and in this paper, the hyper-parameters of the GP model are learned by maximizing the log-likelihood function given by

$$\log p\left([\boldsymbol{y}]_{\cdot,p}\left|\boldsymbol{Z},\theta\right.\right)=-\frac{1}{2}[\boldsymbol{y}]_{\cdot,p}^T\left(K\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I}\right)^{-1}[\boldsymbol{y}]_{\cdot,p}-\frac{1}{2}\log\left|K\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I}\right|-\frac{m}{2}\log\left(2\pi\right) \tag{18}$$

It is a nonlinear non-convex optimization problem that we can use some nonlinear solvers to minimize the value subject to the hyper-parameters, such as conjugate gradient (CG). In addition, the hyperparameters using maximum likelihood optimization before operation based on the historical data and fix them online during the driving.

$$\theta^*=arg\min\left(-\log p\left([\boldsymbol{y}]_{\cdot,p}\left|\boldsymbol{Z},\theta\right.\right)\right) \tag{19}$$

C. Sparse Approximations

Due to the high computational complexity of GP when dealing with large-scale data, sparse GPs (SPGP) algorithms, in this work, will be utilized to relieve the computational complexity, and we will use the popular sparse approximation approaches Fully Independent Training Conditional (FITC) [38]. Given a set of $n(n\ll m)$ inducing inputs $\boldsymbol{Z}_{ind}=[z_1,\ldots,z_n]^T$ and using a shorthand notation

$$Q\left(A,B\right)\triangleq K\left(A,Z_{ind}\right)K\left(Z_{ind},Z_{ind}\right)K\left(Z_{ind},B\right)$$

the approximate posterior distribution is given by

$$m_p^{SP}(z)=K\left(z^*,\boldsymbol{Z}_{ind}\right)\boldsymbol{\Gamma} \tag{20a}$$

$$\Sigma_p^{SP}(z)=K\left(z^*,z^*\right)-Q\left(z^*,z^*\right)+K\left(z^*,\boldsymbol{Z}_{ind}\right)\Omega K\left(\boldsymbol{Z}_{ind},z^*\right) \tag{20b}$$

To decrease the computational complexity, $\boldsymbol{\Gamma}$ and Ω will be calculated offline as

$$\Omega=\left[K\left(\boldsymbol{Z}_{ind},\boldsymbol{Z}_{ind}\right)+K\left(\boldsymbol{Z}_{ind},\boldsymbol{Z}\right)\Theta^{-1}K\left(\boldsymbol{Z},\boldsymbol{Z}_{ind}\right)\right]^{-1} \tag{21a}$$

$$\Theta=\mathrm{diag}\left[K\left(\boldsymbol{Z},\boldsymbol{Z}\right)-Q\left(\boldsymbol{Z},\boldsymbol{Z}\right)+\sigma_p^2\boldsymbol{I}\right] \tag{21b}$$

$$\boldsymbol{\Gamma}=\Omega K\left(\boldsymbol{Z}_{ind},\boldsymbol{Z}\right)\Theta^{-1}[\boldsymbol{y}]_{\cdot,p} \tag{21c}$$

Combining the individual output dimensions, similar to Eq.(17), the sparse GP approximation is given by

$$r^{SP}(z)\sim\mathcal{N}\left(m^{SP}(z),\Sigma^{SP}(z)\right) \tag{22}$$

with $m^{SP}(z)=[m_1^{SP}(z);\cdots;m_{n_p}^{SP}(z)]$, $\Sigma^{SP}(z)=\mathrm{diag}([\Sigma_1^{SP}(z);\cdots;\Sigma_{n_p}^{SP}(z)])$.

D. Model Learning

Given the nominal model $f(x_k,u_k)$ presented in Eq.(9), we use GP to derive the unknown function $g(z_k)$ of the system from the previously collected system state and input data. The training data y_k for learning the GP model is created by the error between the nominal model predictions and measurements

$$y_k=g\left(z_k\right)+\omega_k=C^\dagger\left(x_{k+1}-f\left(x_k,u_k\right)\right) \tag{23}$$

where $z_k=[x_k;u_k]\in\mathbb{R}^{n_z}$.

It is assumed that only the states v_x, v_y and $\dot{\psi}$ are influenced by the uncertainty and noise, while the remaining three states are calculated by kinematic relations. In this paper, we will use the GP function $r(z_k)$ to estimate the unknown function $g(z_k)$, and result in the following stochastic model

$$x_{k+1}=f\left(x_k,u_k\right)+C\left(r\left(z_k\right)+\omega_k\right) \tag{24}$$

where $C=[\boldsymbol{0}_{3\times3};\boldsymbol{I}_{3\times3}]$.

GP is typically accurate in one-step predictions, but we have to use the stochastic output as input in the next prediction when predicting forward over the MPC prediction horizon, which introduces uncertainty into the discrete-time system. Therefore, the predicted states are stochastic variables, and we assumed that they follow the Gaussian distributions at each step, i.e. $x_k\sim\mathcal{N}(\mu_k^{(x)},\Sigma_k^{(x)})$. It is assumed that the vehicle state and the residual model uncertainty are jointly Gaussian distributed at each iteration

$$\begin{bmatrix}x_k\\ r_k\end{bmatrix}\sim\mathcal{N}\left(\mu_k,\Sigma_k\right)=\mathcal{N}\left(\begin{bmatrix}\mu_k^{(x)}\\ m_k^r\left(z_k\right)\end{bmatrix},\begin{bmatrix}\Sigma_k^{(x)} & S\\ \nabla_x m_k^r\left(z_k\right)\Sigma_k^{(x)} & \Sigma_k^r\end{bmatrix}\right) \tag{25}$$

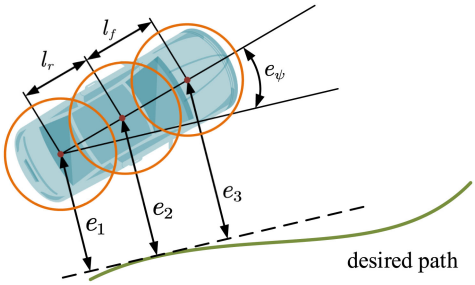


Fig. 12. Circle decomposition of the vehicle shape and localization of the vehicle with a given desired path.

where S denotes the corresponding elements by symmetry.

To evaluate the posterior of a GP with Gaussian input, the mean and variance will be calculated by applying the prediction step of the Extended Kalman Filter (EKF) [32]. This results in predicted mean $\mu_{k+1}^{(x)}$ and variance $\Sigma_{k+1}^{(x)}$ given by

$$\mu_{k+1}^{(x)} = f\left(\mu_k^{(x)}, u_k\right) + Cm_k^r\left(z_k\right) \tag{26a}$$

$$\Sigma_{k+1}^{(x)} = \left[\nabla_x f\left(\mu_k^{(x)}, u_k\right)\ C\right] \Sigma_k \left[\nabla_x f\left(\mu_k^{(x)}, u_k\right)\ C\right]^T \tag{26b}$$

E. Application of Risk Map

The risk of ego vehicle is mainly associated with the occupancy grid by its geometric shape in the risk map. For reducing the computational complexity, we use the geometric shape composed of three circles to replace the real geometric shape of the vehicle, as depicted in Fig. 12.

According to the orientation deviation e_ψ and lateral deviation e_y of the ego vehicle from the desired path, we can easily get the specific location of the ego vehicle. As shown in Fig. 12, we make use of the center of the rear axle, the CoG, and the center of the front axle to make three circles with equal radius, and use these three circles to approximate the geometric shape of the ego vehicle. The distances between the centers of the three circles and the desired path are approximately given by the following formula

$$e_1 = e_y - l_r \sin\left(e_\psi\right) \tag{27a}$$

$$e_2 = e_y \tag{27b}$$

$$e_3 = e_y + l_f \sin\left(e_\psi\right) \tag{27c}$$

where l_r and l_f are the distance from the CoG to the front and the rear axles, respectively.

In this way, the geometric shape of the whole vehicle can be determined uniquely at each step, and the occupancy grid set $\mathbb{V}_k$ of the ego vehicle at each sampling time k can be easily obtained. We take the highest cost C_k as the risk cost of the ego vehicle at step k

$$C_k = \max\left(RC_k^{(i,j)}\right),\ (i,j) \in \mathbb{V}_k \tag{28}$$

where $\mathbb{V}_k \in \mathbb{R}^{m\times 2}$, m is the number of occupancy grids at step k.

F. Constraints Formulation

1) Chance constraints: The system is subject to state constraints $\mathcal{X} \in \mathbb{R}^{n_x}$, and input constraints $\mathcal{U} \in \mathbb{R}^{n_u}$ with the following form

$$\mathcal{X} = x \in \{\mathbb{R}^{n_x} \,|H^x x \le b^x\} \tag{29a}$$

$$\mathcal{U} = u \in \{\mathbb{R}^{n_u} \,|H^u u \le b^u\} \tag{29b}$$

Due to the random disturbance in the process, the constraints $x \in \mathcal{X},\ u \in \mathcal{U}$ may not be satisfied and we convert it to chance constraints

$$Pr\left(x_k \in \mathcal{X}\right) \ge 1 - \epsilon_x \tag{30a}$$

$$Pr\left(u_k \in \mathcal{U}\right) \ge 1 - \epsilon_u \tag{30b}$$

where ϵ_x and ϵ_u are the probability of violation.

2) Safety constraints:

i) Risk boundary constraints: Fig. 12 shows that different vehicle postures can be obtained by expanding to both sides along the vertical direction of the desired path. At each predicted position, the lateral safety area of the ego vehicle can be obtained by restricting the vehicle body within the road risk area, and combining Eq. (27), the following risk boundary constraints are given by

$$e_{i,\min} \le e_i \le e_{i,\max}\ \ i = 1,2,3 \tag{31}$$

ii) Slip constraints: as the tire forces are linearized when modeling, the slip angle needs to be limited to the linear region of the tire forces.

$$\alpha_{\min}^{slip} < \alpha_f^{slip} < \alpha_{\max}^{slip} \tag{32a}$$

$$\alpha_{\min}^{slip} < \alpha_r^{slip} < \alpha_{\max}^{slip} \tag{32b}$$

where α_f^{slip} and α_r^{slip} are the front tire slip angle and rear tire slip angle, respectively.

To increase the chance of finding the solution of the optimal control problem, we use the slack variables ε to relax the risk boundary constraints and the slip constraints. In this way, Eq.(31) and Eq.(32) can be written in shorthand as

$$g\left(\mu^{(x)}, u\right) \le \varepsilon \tag{33}$$

where ε indicates that a partial violation is allowed.

G. GP-MPC Formulation

By minimizing the expected value of a Gaussian distributed quadratic function, we define a stochastic MPC problem over a finite receding horizon. The resulting optimization problem can then be formulated as

$$\underset{\Delta u,\varepsilon}{minimize} \sum_{k=0}^{N_p-1} \lambda C_{t+k,t} + \left\|\mu_{t+k,t}^{(x)} - x_{t+k,t}^{ref}\right\|_Q^2 + \left\|u_{t+k,t}\right\|_R^2 + \left\|\Delta u_{t+k,t}\right\|_S^2 + \beta^T \varepsilon_k \tag{34}$$

TABLE I
VEHICLE PARAMETERS

Symbol	Description	Value
m	Vehicle mass	$2060kg$
I_z	Vehicle yaw inertia	$3430kg \cdot m^2$
l_f	Distance from CoG to front axle	$1.41m$
l_r	Distance from CoG to rear axle	$1.37m$
C_f	Front-tire cornering stiffness	$84000N/rad$
C_r	Rear-tire cornering stiffness	$84000N/rad$

$$
\begin{aligned}
&s.t.\\
&\mu^{(x)}_{t+k+1,t} = f\\
&\left(\mu^{(x)}_{t+k,t}, u_{t+k,t}\right) + Cm^{r}_{t+k,t},\ k = 0, 1, \ldots, N_p - 1\\
&Pr\left(\mu^{(x)}_{t+k,t} \in \mathcal{X}\right) \geq 1 - \epsilon_x,\ k = 0, 1, \ldots, N_p - 1\\
&Pr\left(u_{t+k,t} \in \mathcal{U}\right) \geq 1 - \epsilon_u,\ k = 0, 1, \ldots, N_c - 1\\
&g\left(\mu^{(x)}_{t+k,t}, u_{t+k,t}\right) \leq \varepsilon_k,\ k = 0, 1, \ldots, N_p - 1\\
&\Delta u_{t+k,t} = u_{t+k,t} - u_{t+k-1,t}\\
&\Delta u_{\min} \leq \Delta u_{t+k,t} \leq \Delta u_{\max},\ k = 0, 1, \ldots, N_c - 1\\
&\Delta u_{t+k,t} = 0,\ k = N_c, \ldots, N_p\\
&u_{t-1,t} = u\left(t-1\right)\\
&\mu^{(x)}_{t,t} = \mu^{(x)}(t)
\end{aligned}
\tag{35}
$$

where t denotes the current time instant and $t+k, t$ index denotes the predicted value at k steps ahead of t. N_p and N_c are the prediction horizon and control horizon. $\Delta u_{\min}$ and $\Delta u_{\max}$ are the lower and upper bounds of the control inputs changes. In equation (32), the first term $C_{t+k,t}$ is the risk cost and λ is the weight; The second term is the quadratic cost on the state tracking error, i.e., $\|A-B\|^2_Q \triangleq (A-B)^T Q(A-B)$, and Q is the weighting matrices. Similarly, the third and fourth terms are the quadratic cost on the control inputs and their changes, and R, S are the corresponding weight matrices. The last term ε_k is the slack variables and β is the weight.

VI. SIMULATION RESULTS

In this section, the effectiveness of the risk map and the comparison between GP-MPC and nominal MPC are tested in the simulation environment. The algorithms are implemented in MATLAB, while the highly reliable full-vehicle dynamics are given by Carsim. The nonlinear optimization problem in MPC controllers is solved by the open-source toolbox CasADi [39], and the main parameters in the simulation environment are listed in Tables I and II.

A. Effectiveness of Risk Map

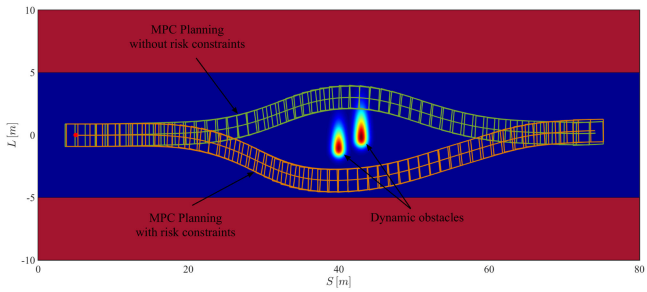
In this section, we will verify the effectiveness of the risk map in improving the performance of trajectory planning. For comparison, we set up two MPC planners: one considers risk constraints, and the other does not. Except for risk constraints,

TABLE II
DESIGN PARAMETERS

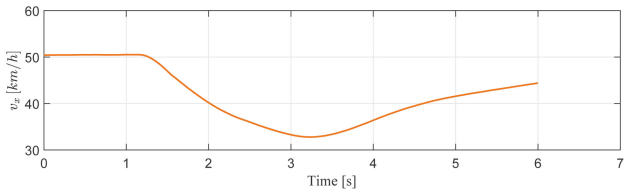
Parameter	Value	Units	Parameter	Value	Units
$u_{\max}$	$[0.5, 0.2]$	–	T_s	100	ms
$u_{\min}$	$[-0.5, -0.2]$	–	N_p	30	–
$\Delta u_{\max}$	$[0.3, 0.02]$	–	N_c	30	–
$\Delta u_{\min}$	$[-0.3, -0.02]$	–	$\alpha^{slip}_{\min}$	-4	deg
Q	$(0.01, 5, 1, 2, 1, 0)$	–	$\alpha^{slip}_{\max}$	4	deg
R	$(0.1, 1)$	–	β	500	–
S	$(1, 10)$	–	λ	10	–
μ	1.0	–	ε	0.05	–

TABLE III
REAL-TIME DESIGN PARAMETERS

Parameter	Value	Units	Parameter	Value	Units
$u_{\max}$	$[0.5, 0.15]$	–	T_s	100	ms
$u_{\min}$	$[-0.5, -0.15]$	–	N_p	30	–
$\Delta u_{\max}$	$[0.3, 0.02]$	–	N_c	10	–
$\Delta u_{\min}$	$[-0.3, -0.02]$	–	$\alpha^{slip}_{\min}$	-4	deg
Q	$(0.001, 5, 1, 2, 1, 0)$	–	$\alpha^{slip}_{\max}$	4	deg
R	$(0.1, 1)$	–	β	500	–
S	$(1, 10)$	–	λ	10	–
μ	0.8	–	ε	0.05	–



(a)



(b)

Fig. 13. Experimental results for low-speed dynamic obstacles avoidance.(a) Bird's eye view of the AVs. (b) The longitudinal velocity of MPC planning with risk constraints.

the two planners are equipped with the same parameters. We set up two scenarios for the two planners to test the obstacle avoidance capability and the corresponding simulation results are shown in Figs. 13 and 14.

1) Obstacle Avoidance in Scenario 1: As shown in Fig. 13, in this scenario, the starting location of the ego vehicle is located at the red point, and two low-speed dynamic obstacles crossing the road are in front of the vehicle. The MPC planner that ignored risk constraints treated all obstacles as static obstacles and planned a convenient path for controller execution. However, it increased the probability of collision with obstacles. Instead, the

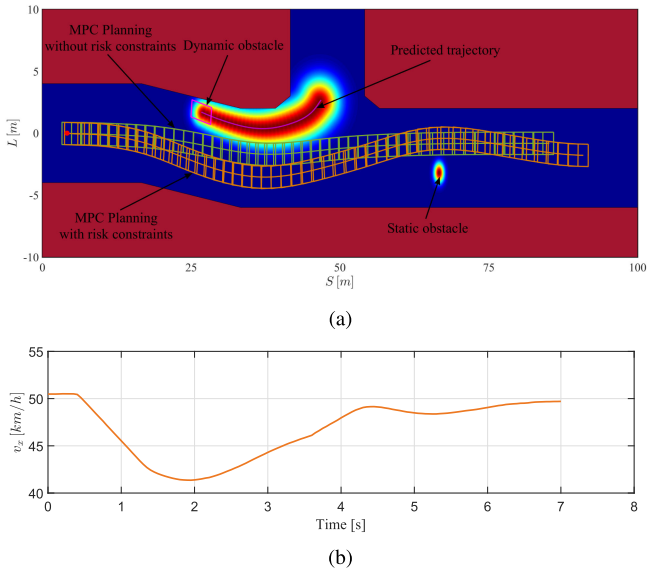


Fig. 14. Experimental results for both static and high-speed dynamic obstacles avoidance. (a) Bird's eye view of the AVs. (b) The longitudinal velocity of MPC planning with risk constraints.

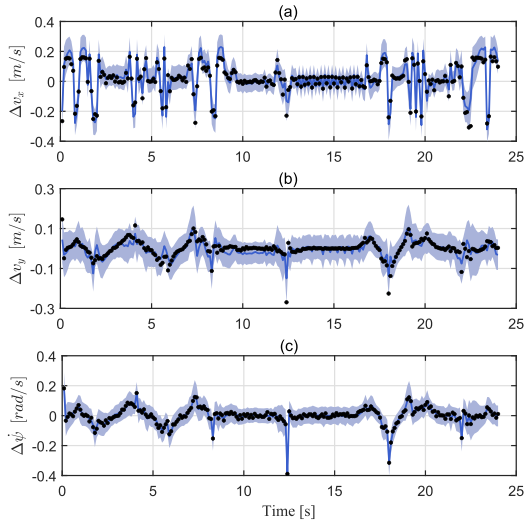


Fig. 15. Recorded model error and GP error prediction during a lap. (a) Prediction error in v_x. (b) Prediction error in v_y. (c) Prediction error in $\dot{\psi}$. The black dots are the model error under process noise. The blue line is the mean predicted model error while the light blue is the 2σ confidence interval.

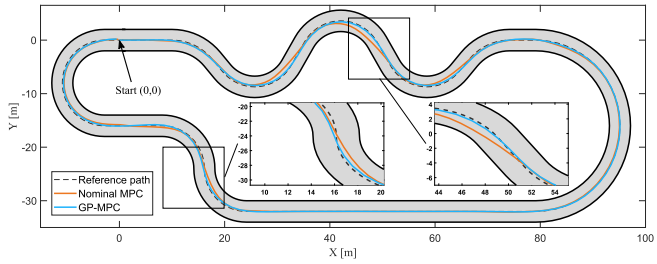


Fig. 16. The driven trajectory between nominal MPC and GP-MPC with same configuration during a lap.

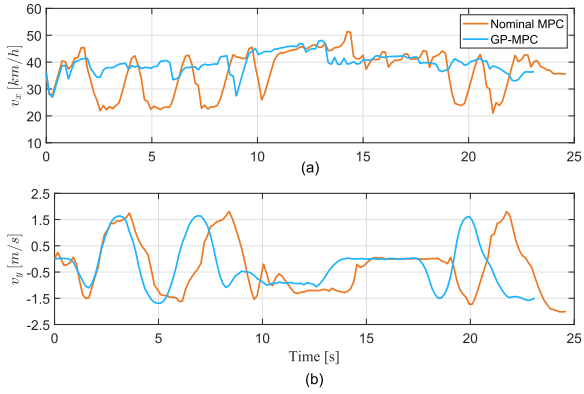


Fig. 17. The longitudinal velocity v_x and lateral velocity v_y between nominal MPC and GP-MPC with same configuration during a lap.

planner considering the risk constraints can consider the future movement trend of the obstacles, thereby obtaining a safe and feasible path.

2) Obstacle Avoidance in Scenario 2: As shown in Fig. 14, there is a high-speed dynamic vehicle at the left front side of the ego vehicle preparing to turn left. Its predicted trajectory forms a risk field in the risk map, which poses a vital challenge to the MPC planner. Similar to the previous scenario, the path generated by the MPC planner without considering risk constraints is close to the dynamic obstacles, and the only way to avoid a collision in the future is through emergency avoidance or emergency deceleration, which is undoubtedly failed planning. In contrast, the planner considering risk constraints can consider the future movement trend of the dynamic vehicle, thus planning a safe path.

Through the simulation experiments of the above two scenarios, in an uncertain environment, combining the MPC planner with risk constraints can significantly improve scene adaptability and driving safety.

B. GP-MPC Validation

1) Verification of GP Prediction: To demonstrate the effectiveness of learning uncertain disturbances with GP, a path tracking experiment along a curvy reference path is tested in the road course shown in Fig. 16. In the process of tracking, the vehicle state, control input and real prediction errors at each sampling time are taken as the training data of the GP model. Through the real-time updated GP model, the mean and variance of the prediction error at each time step can be obtained. Fig. 15 shows the actual error and GP prediction error of v_x, v_y and $\dot{\psi}$ in the simulation, and the results show that the real model errors can be well fitted by the mean and uncertainty estimate of GP.

2) Verification of GP-MPC Controller: To demonstrate the learning performance, we used the GP-MPC controller and the nominal MPC controller to perform a path tracking experiment in the road course, illustrated in Fig. 16. The experimental results show that the GP-MPC controller performs better than the nominal MPC controller in sharp-turning scenarios. Fig. 17(a) shows

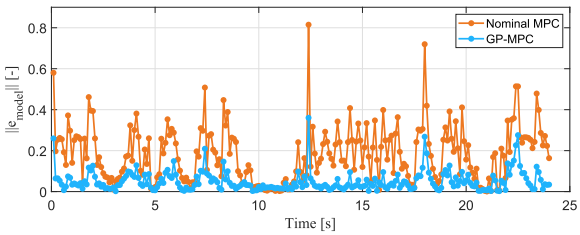


Fig. 18. The prediction 2-norm error between nominal MPC and GP-MPC during a lap.

Fig. 19. Our automated vehicle test platform.

that the GP-MPC controller has higher and more stable longitudinal velocity than the nominal MPC controller. Fig. 17(b) shows that the lateral velocity of the GP-MPC controller changes more smoothly in the curve. To quantify the performance of the GP-MPC controller, we compare the 2-norm error $\|e_{model}\|$, i.e., $\|e_{model}\| = \|x(k+1) - f(x_k, u_k)\|$ of the two controllers in the system dynamics at each time step, and the results are shown in Fig. 18.

Through the above experiments, it can be clearly observed that the GP-MPC controller improves the accuracy of the dynamic model and has a better performance by learning the model prediction error online.

VII. Experiment Results

A. Experimental Platform

To verify the effectiveness of the algorithm in real unstructured environments, we use a modified 4WD car as an experimental platform. As shown in Fig. 19, the ego vehicle is equipped with the necessary sensors for automated driving, including an IBEO four-layer laser scan instrument, a Velodyne HDL-64E lidar, two Velodyne HDL-32E lidar, four high-resolution cameras, and a differential GPS/INS system. The onboard computer is equipped with an Intel i7 2.2 GHz processor and 16 GB of DDR4 2400 MHz memory. We use the Ubuntu 16.04 LTS system with ROS Kinetic as the development and operation environment.

B. Parameter Evaluation

To evaluate the performance of the controller, several experiments have been performed under a double lane change maneuver with different sampling times, $T_s \in \{0.05\,s, 0.1\,s, 0.15\,s\}$,

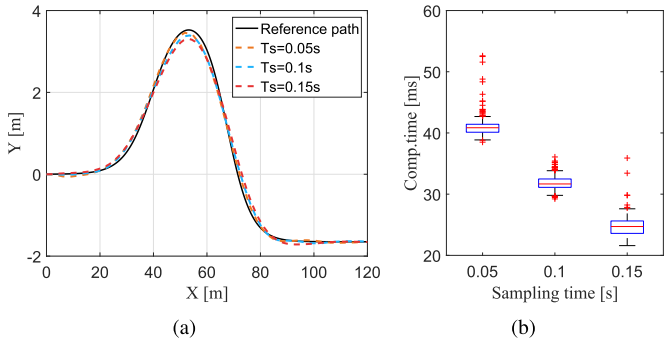


Fig. 20. Comparison of experimental results and computation time for different sampling times. The prediction and control horizons are $N_p = 30$ and $N_c = 10$, respectively, and the initial velocity is 20 m/s. (a) Comparison of path tracking results. (b) Comparison of computation time. The red line in the box plot represents the median. The computation time for the 75th and 25th percentiles are represented by the top and bottom edges of the box, respectively. The red crosses denotes the outliers.

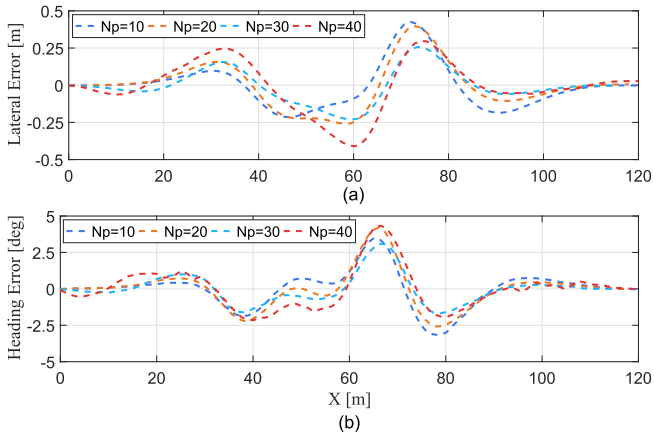


Fig. 21. Comparison of experimental results for different prediction horizons. The control horizon is $N_c = 10$, sampling time is $T_s = 0.1\,s$, and the initial velocity is 20 m/s. (a) Comparison of lateral error. (b) Comparison of heading error.

prediction horizons, $N_p \in \{10, 20, 30, 40\}$, and control horizons, $N_c \in \{5, 10, 20\}$.

As shown in Fig. 20(a), the shorter sampling time T_s results in smaller trajectory tracking error and better control performance. However, Fig. 20(b) shows that smaller sampling times lead to higher computation times for the controller. When T_s is 0.05 s, the average computation time is close to the cycle time of the controller with a few isolated moments exceeding 0.05 s. Therefore the sampling time should take into account the trade-off between computational efficiency and control performance, and in this work the sampling time was chosen to be $T_s = 0.1\,s$.

The prediction horizon N_p has a significant effect on the performance of controller, and when a longer N_p is chosen, more information on the future vehicle dynamics can be obtained to predict longer distances. Fig. 21 shows that when N_p is in the range of $[10, 30]$, both the lateral error and heading angle error decrease as the prediction horizon N_p increases. However, when N_p is 40, the control performance begins to degrade compared to $N_p = 30$. Consequently, both the large and small prediction

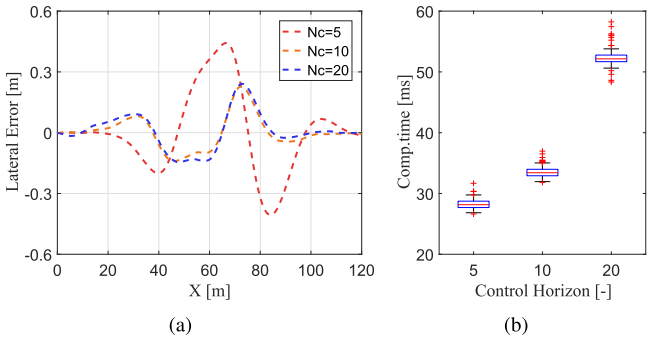


Fig. 22. Comparison of experimental results and computation time for different control horizons. The prediction horizon is $N_p = 30$, sampling time is $T_s = 0.1\,s$, and the initial velocity is 15 m/s. (a) Comparison of lateral error. (b) Comparison of computation time.

Fig. 23. Satellite view of the off-road course by Google Earth.

horizons will degrade the control performance, and in this work we choose $N_p = 30$.

Generally, the relationship $N_c \leqslant N_p$ needs to be satisfied between the prediction horizon N_p and control horizon N_c. Fig. 22 shows that small control horizon ($N_c = 5$) fails to achieve good control performance, and a longer N_c can improve the tracking accuracy. However, when N_c is increases to a certain range, it will have little improvement on the control performance and will incur considerable computational cost. In this study, we choose $N_c = 10$.

C. Trajectory Prediction

To get an accurate trajectory prediction model, we collect a large amount of driving data in a real off-road environment to train our trajectory prediction model. As shown in Fig. 23, the off-road driving dataset was collected at a test site of larger than 5 square kilometers. Due to different drivers having different driving styles, we use more than ten drivers to complete the dataset to make it contain different driving behaviors. The dataset contains more than 3000 trajectories, with an average length of approximately 30 - 50 m.

To verify the effectiveness of the trajectory prediction algorithm, we performed an experiment to predict the future motion trajectory of dynamic vehicles before an intersection, as shown in Fig. 24(b). For dynamic vehicles in this scenario, there will be different driving behaviors in the future for different historical trajectories, such as slowing down to turn left (Fig. 24(c)) or driving straight at a uniform velocity (Fig. 24(d)). The future trajectory of the dynamic vehicle is predicted by the LSTM network, and the results are shown in Fig. 24(a) and Fig. 24(e). The blue rectangle in the figure represents the dynamic vehicle, the red curve is the predicted future trajectory of the dynamic vehicle, and the colored curve is the historical trajectory of the dynamic vehicle while the change of color indicates the change of its historical velocity.

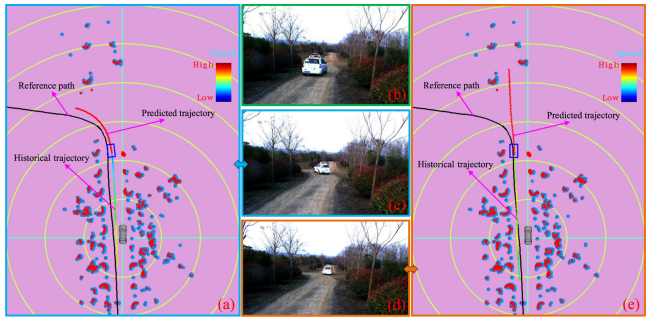


Fig. 24. The prediction results of different historical trajectories for high-speed vehicles in the same scenario.

From the historical trajectory in Fig. 24(a), it can be clearly observed that the speed of the dynamic vehicle slowly decreased before the intersection. Combined with the location and curvature of the historical trajectory, a predicted left-turn trajectory at the intersection was obtained by the LSTM network. In contrast, as shown in Fig. 24(c), the dynamic vehicle moved forward at a high uniform speed before the intersection, and the corresponding predicted trajectory is moving straight ahead. The actual driving behaviors of the dynamic vehicle at the intersection are shown in Fig. 24(c) and Fig. 24(d). Compared with the predicted results, it can be seen that the outputs of the LSTM network match the actual results, which can effectively predict the future motion trend of the dynamic vehicles.

D. Verifications of Collision Avoidance

To validate the proposed trajectory planning algorithm, field tests were carried out on our automated vehicle platform in an unstructured environment. The nominal model is discretized with a Runge-Kutta 4th-order integration using a sampling time of $T_s = 100\,ms$. The core algorithm is implemented in C++, and ROS is used for communication. In addition, we use the HPIPM solver in the open-source Control Toolbox (CT) [40] to improve the solving efficiency. To comprehensively evaluate the reliability of the proposed method, the following three test scenarios are defined in an off-road environment.

1) Scenario 1: The ego vehicle is moving forward at an initial speed of 20 km/h. There are static obstacles on the left and right sides of the road ahead. This scenario is intended to validate the effectiveness of static obstacle avoidance for the trajectory planning controller in unstructured environments.

2) Scenario 2: The ego vehicle is moving forward at an initial speed of 21 km/h. The obstacles are composed of static obstacles on the left and a slow-moving pedestrian on the right. This scenario is intended to validate the avoidance effect for both static and low-speed dynamic obstacles in unstructured environments.

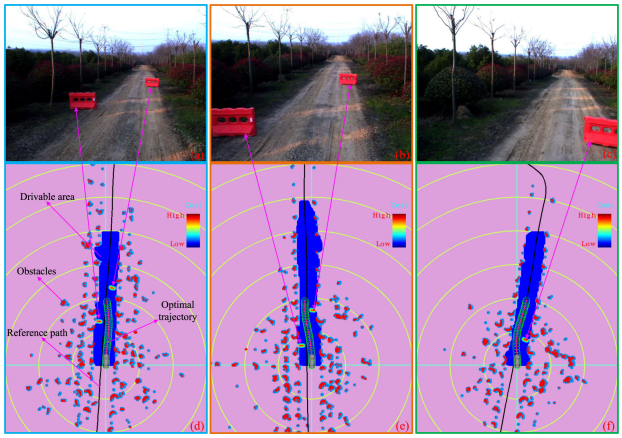


Fig. 25. Scenario 1: The snapshots of the local trajectory generation scenario for static obstacles avoidance at three consecutive instants of time.

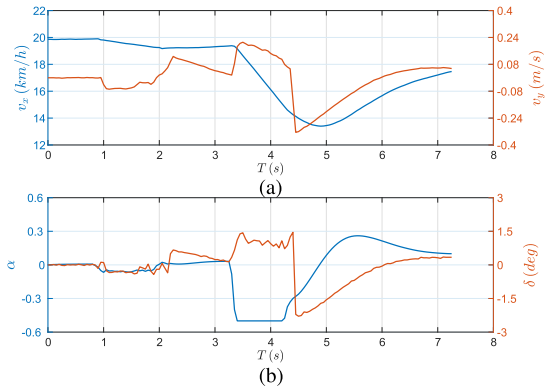


Fig. 26. Experimental results in scenario 1: (a) Longitudinal velocity v_x and lateral velocity v_y. (b) Braking/throttle ratio α and steering angle δ.

3) Scenario 3: The ego vehicle is moving forward at an initial speed of 29 km/h. There are three static obstacles on the right side of the road ahead, and on the left side is a dynamic vehicle moving at high speed. This scenario is intended to test the trajectory prediction results of high-speed dynamic obstacles and the planning effect of the trajectory planning controller in complex scenarios.

The above three typical scenarios are chosen to verify the capability of the proposed trajectory planning controller for real-time obstacle avoidance in unstructured environments. Figs. 25, 27 and 29 show three consecutive snapshots of the real-time planning results, where (a-c) are images collected by the onboard cameras and (d-f) are the corresponding real-time local planning maps. We use an occupancy grid map with a resolution of $10\,cm \times 10\,cm$ to represent the local environment. The grid with red and light blue are static obstacles outside the road. The blue area is the drivable area extracted by the perception system. The colored grids in the drivable area correspond to the obstacles in the image, and the color of the grid represents the corresponding risk value. The black curve is the global reference path, while the red curve is the local planning trajectory. The

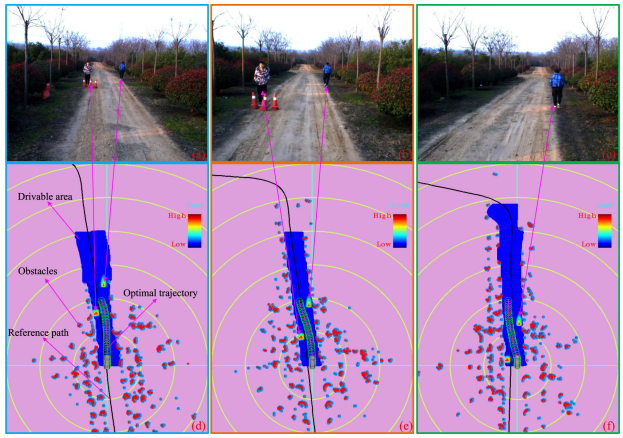


Fig. 27. Scenario 2: The snapshots of the local trajectory generation scenario while interacting with both static and low-speed dynamic obstacles at three consecutive instants of time.

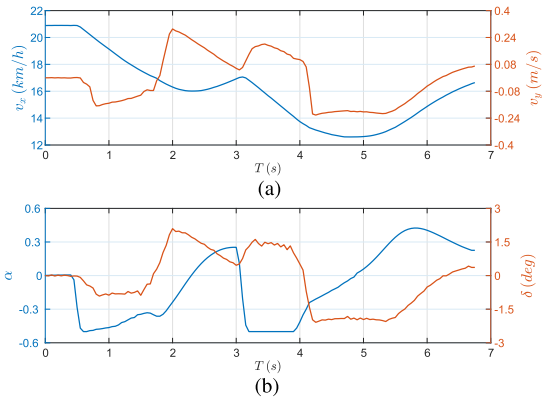


Fig. 28. Experimental results in scenario 2: (a) Longitudinal velocity v_x and lateral velocity v_y. (b) Braking/throttle ratio α and steering angle δ.

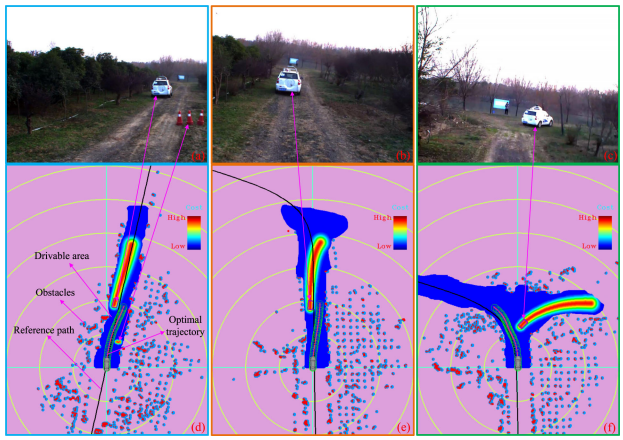


Fig. 29. Scenario 3: The snapshots of the local trajectory generation scenario while interacting with both static and high-speed dynamic obstacles at three consecutive instants of time.

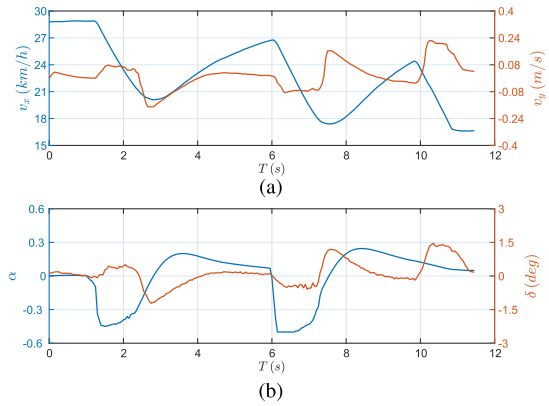


Fig. 30. Experimental results in scenario 3: (a) Longitudinal velocity v_x and lateral velocity v_y. (b) Braking/throttle ratio α and steering angle δ.

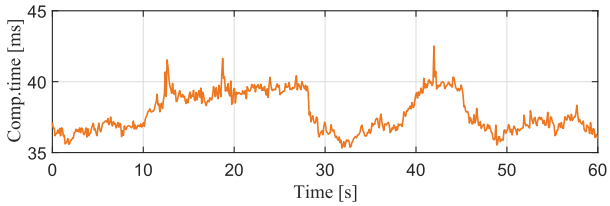


Fig. 31. Computational time at each time step in the experiments.

experimental longitudinal velocity, lateral velocity, and control inputs are shown in Figs. 26, 28, and 30.

Fig. 25 demonstrates the ability of the automated vehicle to safely and continuously avoid static obstacles, while Fig. 27 shows the safety avoidance effect on both static and low-speed dynamic obstacles. It can be seen from the results of Figs. 26 and 28 that the ego vehicle can promptly decrease the longitudinal velocity during the obstacle avoidance process, and accelerate again by increasing the throttle-braking ratio α when leaving obstacles, and maintain a small lateral velocity by steering in small increments. As seen in Figs. 29 and 30, the automated vehicle is still able to avoid static and high-speed dynamic obstacles safely and successfully return to the reference path. Fig. 31 shows the computational time at each time step in the experiments. The average computation time is 37.78 ms and the maximum is 42.51 ms, both of which are lower than the controller's cycle time of 100 ms.

VIII. Conclusion and Future Work

This paper has developed and evaluated a learning-based model predictive trajectory planning controller for automated driving in unstructured, dynamic environments with obstacle avoidance. First, a hierarchical risk map is designed to assess the risk of roads and different types of obstacles for the uncertainties arising from unstructured environments. A reliable LSTM network is employed to predict the future trajectories of high-speed dynamic obstacles. Second, trajectory planning and tracking control are integrated into one module with GP learning the nonlinear residual model uncertainty, and collision-free and kinematically-feasible local trajectories are quickly obtained by the constrained optimization techniques. The simulations demonstrate the effectiveness of the risk map in trajectory planning and the feasibility of GP in reducing the model uncertainty. The framework has been implemented on our automated vehicle, and successful experimental results in real unstructured roads have demonstrated that the approach can effectively address various complex scenarios.

Although the strategy proposed in this paper mainly focused on applications in unstructured environments, our strategy can be extended to be used in structured environments by incorporating high precision maps. In addition, a more accurate risk-cost equation can be assigned by further subdividing the properties of the obstacles. Therefore, the trajectory planning strategy will expand its usage scenarios and improve the accuracy of risk map construction through continuous study in the future.

References

[1] H. Marzbani, H. Khayyam, C. N. TO. V. Quoc, and R. N. Jazar, "Autonomous vehicles: Autodriver algorithm and vehicle dynamics," *IEEE Trans. Veh. Technol.*, vol. 68, no. 4, pp. 3201–3211, Apr. 2019.

[2] C. Pek and M. Althoff, "Fail-safe motion planning for online verification of autonomous vehicles using convex optimization," *IEEE Trans. Robot.*, vol. 37, no. 3, pp. 798–814, Jun. 2021.

[3] S. Grigorescu, B. Trasnea, T. Cocias, and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *J. Field Robot.*, vol. 37, no. 3, pp. 362–386, 2020.

[4] O. Sharma, N. C. Sahoo, and N. B. Puhan, "Recent advances in motion and behavior planning techniques for software architecture of autonomous vehicles: A state-of-the-art survey," *Eng. Appl. Artif. Intell.*, vol. 101, 2021, Art. no. 104211.

[5] L. Claussmann, M. Revilloud, D. Gruyer, and S. Glaser, "A review of motion planning for highway autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 5, pp. 1826–1848, May 2020.

[6] J. Shin, D. Kwak, and T. Lee, "Robust path control for an autonomous ground vehicle in rough terrain," *Control Eng. Pract.*, vol. 98, 2020, Art. no. 104384.

[7] A. Roshanianfard, N. Noguchi, H. Okamoto, and K. Ishii, "A review of autonomous agricultural vehicles (the experience of hokkaido university)," *J. Terramechanics.*, vol. 91, pp. 155–183, 2020.

[8] W. Schwarting, J. Alonso-Mora, and D. Rus, "Planning and decision-making for autonomous vehicles," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 1, no. 1, pp. 187–210, 2018.

[9] X. Li, Z. Sun, D. Cao, D. Liu, and H. He, "Development of a new integrated local trajectory planning and tracking control framework for autonomous ground vehicles," *Mech. Syst. Signal Process.*, vol. 87, pp. 118–137, 2017.

[10] Y. Rasekhipour, A. Khajepour, S. K. Chen, and B. Litkouhi, "A potential field-based model predictive path-planning controller for autonomous road vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 18, no. 5, pp. 1255–1267, May 2017.

[11] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[12] Y. Huang *et al.*, "A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach," *IEEE Trans. Ind. Electron.*, vol. 67, no. 2, pp. 1376–1386, Feb. 2020.

[13] J. Cui and B. Zhang, "VeRA: A simplified security risk analysis method for autonomous vehicles," *IEEE Trans. Veh. Technol.*, vol. 69, no. 10, pp. 10494–10505, Oct. 2020.

[14] T. Puphal, M. Probst, and J. Eggert, "Probabilistic uncertainty-aware risk spot detector for naturalistic driving," *IEEE Trans. Intell. Veh.*, vol. 4, no. 3, pp. 406–415, Sep. 2019.

[15] J. Kim and D. Kum, "Collision risk assessment algorithm via lane-based probabilistic motion prediction of surrounding vehicles," *IEEE Trans. Intell. Transp. Syst.*, vol. 19, no. 9, pp. 2965–2976, Sep. 2018.

[16] H. Guo *et al.*, "Safe path planning with gaussian process regulated risk map," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2019, pp. 2044–2051.

[17] B. Patle *et al.*, "A review: On path planning strategies for navigation of mobile robot," *Defence Technol.*, vol. 15, no. 4, pp. 582–606, 2019.

[18] M. Mohanan and A. Salgoankar, "A survey of robotic motion planning in dynamic environments," *Robot. Auton. Syst.*, vol. 100, pp. 171–185, 2018.

[19] D. Dolgov, S. Thrun, M. Montemerlo, and J. Diebel, "Path planning for autonomous vehicles in unknown semi-structured environments," *Int. J. Robot. Res.*, vol. 29, no. 5, pp. 485–501, 2010.

[20] D. Ferguson, T. M. Howard, and M. Likhachev, "Motion planning in urban environments," *J. Field Robot.*, vol. 25, no. 11-12, pp. 939–960, 2008.

[21] R. Kala and K. Warwick, "Multi-level planning for semi-autonomous vehicles in traffic scenarios based on separation maximization," *J. Intell. Robot. Syst.*, vol. 72, no. 3, pp. 559–590, 2013.

[22] U. Graf, P. Borges, E. Hernández, R. Siegwart, and R. Dubé, "Optimization-based terrain analysis and path planning in unstructured environments," in *Proc. Int. Conf. Robot. Automat.*, 2019, pp. 5614–5620.

[23] S. Karaman, M. R. Walter, A. Perez, E. Frazzoli, and S. Teller, "Anytime motion planning using the RRT," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2011, pp. 1478–1483.

[24] B. Ichter, E. Schmerling, T.-W. E. Lee, and A. Faust, "Learned critical probabilistic roadmaps for robotic motion planning," in *Proc. IEEE Int. Conf. Robot. Automat.*, 2020, pp. 9535–9541.

[25] C. Wang *et al.*, "Autonomous mobile robot navigation in uneven and unstructured indoor environments," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst.*, 2017, pp. 109–116.

[26] J. Wang, B. Li, and M. Q. Meng, "Kinematic constrained bi-directional RRT with efficient branch pruning for robot path planning," *Expert Syst. Appl.*, vol. 170, 2021, Art. no. 114541.

[27] M. N. Zafar and J. Mohanta, "Methodology for path planning and optimization of mobile robots: A review," *Procedia Comput. Sci.*, vol. 133, pp. 141–152, 2018.

[28] S. Dixit *et al.*, "Trajectory planning for autonomous high-speed overtaking in structured environments using robust MPC," *IEEE Trans. Intell. Transp. Syst.*, vol. 21, no. 6, pp. 2310–2323, Jun. 2020.

[29] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. Hrovat, "MPC-based approach to active steering for autonomous vehicle systems," *Int. J. Veh. Auton. Syst.*, vol. 3, no. 2-4, pp. 265–291, 2005.

[30] X. Di and R. Shi, "A survey on autonomous vehicle control in the ERA of mixed-autonomy: From physics-based to ai-guided driving policy learning," *Transp. Res. Part C. Emerg. Technol.*, vol. 125, 2021, Art. no. 103008.

[31] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger, "Learning-based model predictive control: Toward safe learning in control," *Annu. Rev. Control Robot. Auton. Syst.*, vol. 3, no. 1, pp. 269–296, 2020.

[32] L. Hewing, J. Kabzan, and M. N. Zeilinger, "Cautious model predictive control using gaussian process regression," *IEEE Trans. Control Syst.*, vol. 28, no. 6, pp. 2736–2743, Nov. 2020.

[33] J. Kabzan, L. Hewing, A. Liniger, and M. N. Zeilinger, "Learning-based model predictive control for autonomous racing," *IEEE Robot. Autom. Lett.*, vol. 4, no. 4, pp. 3363–3370, Oct. 2019.

[34] C. J. Ostafew, A. P. Schoellig, and T. D. Barfoot, "Robust constrained learning-based NMPC enabling reliable mobile robot path tracking," *Int. J. Robot. Res.*, vol. 35, no. 13, pp. 1547–1563, 2016.

[35] Y. Gao, A. Gray, H. E. Tseng, and F. Borrelli, "A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles," *Veh. Syst. Dyn.*, vol. 52, no. 6, pp. 802–823, 2014.

[36] H. B. Pacejka and E. Bakker, "The magic formula tyre model," *Veh. Syst. Dyn.*, vol. 21, no. S1, pp. 1–18, 1992.

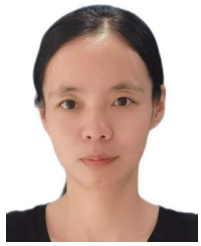
[37] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, MA, USA: MIT Press, 2006.

[38] J. Quiñonero-Candela and C. E. Rasmussen, "A unifying view of sparse approximate Gaussian process regression," *J. Mach. Learn. Res.*, vol. 6, pp. 1939–1959, 2005.

[39] J. A. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi: A software framework for nonlinear optimization and optimal control," *Math. Program. Comput.*, vol. 11, no. 1, pp. 1–36, 2019.

[40] M. Giftthaler, M. Neunert, M. Stäuble, and J. Buchli, "The control toolbox-an open-source c library for robotics, optimal and model predictive control," in *Proc. IEEE Int. Conf. Simulation, Modeling, Program. Auton. Robots*, 2018, pp. 123–129.

Zhiyuan Li is currently working toward the Ph.D. degree in detection technology and automatic equipment with the University of Science and Technology of China, Hefei, China. He is also with the Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei, China. His current research interests include motion planning and behavior planning for robotic systems and automated vehicles.



Pan Zhao received the Ph.D. degree in detection technology and automatic equipment from the University of Science and Technology of China, Hefei, China, in 2012. Since 2012, she has been an Associate Professor with the Institute of Intelligent Machines, Hefei Institutes of Physical Science, Hefei, China. Her research focuses on self-driving vehicles, such as optimal control, motion planning, and mobile robot navigation.

Chunmao Jiang is currently working toward the Ph.D. degree in detection technology and automatic equipment from the University of Science and Technology of China, Hefei, China. He is also with the Hefei Institutes of Physical Science, Chinese Academy of Sciences, Hefei, China. His current research interests include optimal control, model learning, and motion planning for robotic systems, especially for automated driving car and unmanned ground vehicles.

Weixin Huang received the M.S. degree in integrated circuits engineering from Shenzhen University, Shenzhen, China, in 2016. He is currently working toward the Ph.D. degree in detection technology and automatic equipment with the University of Science and Technology of China, Hefei, China. His research focuses on self-driving vehicles, such as 3D LiDAR data processing, object recognition and tracking, and deep learning.

Huawei Liang received the Ph.D. degree in detection technology and automation from the University of Science and Technology of China, Hefei, China, in 2007. He is currently a Principal Investigator and the Deputy Director with the Institute of Intelligent Machines, Hefei Institutes of Physical Science Chinese Academy of Science, Hefei, China. He has been engaged in the robotics, intelligent vehicle technology, detection technology and automation device, pattern recognition and intelligent system, control theory, and control engineering.