A Novel Learning Framework for Sampling-Based Motion Planning in Autonomous Driving

Yifan Zhang,[1*] Jinghuai Zhang,[1*] Jindi Zhang,[1] Jianping Wang,[1] Kejie Lu,[2] Jeff Hong[3]
[1]City University of Hong Kong, [2]University of Puerto Rico at Mayaguez, [3]Fudan University
{yif.zhang, jzhang538-c, jd.zhang}@my.cityu.edu.hk, jianwang@cityu.edu.hk,
kejie.lu@upr.edu, hong_liu@fudan.edu.cn

Abstract

Sampling-based motion planning (SBMP) is a major trajectory planning approach in autonomous driving given its high efficiency in practice. As the core of SBMP schemes, sampling strategy holds the key to whether a smooth and collision-free trajectory can be found in real-time. Although some bias sampling strategies have been explored in the literature to accelerate SBMP, the trajectory generated under existing bias sampling strategies may lead to sharp lane changing. To address this issue, we propose a new learning framework for SBMP. Specifically, we develop a novel automatic labeling scheme and a 2-Stage prediction model to improve the accuracy in predicting the intention of surrounding vehicles. We then develop an imitation learning scheme to generate sample points based on the experience of human drivers. Using the prediction results, we design a new bias sampling strategy to accelerate the SBMP algorithm by strategically selecting necessary sample points that can generate a smooth and collision-free trajectory and avoid sharp lane changing. Data-driven experiments show that the proposed sampling strategy outperforms existing sampling strategies, in terms of the computing time, traveling time, and smoothness of the trajectory. The results also show that our scheme is even better than human drivers.

1 Introduction

In the past decade, autonomous driving has gained significant developments with the joint effort from academia and industry. In 2018, Google's driverless car, Waymo, started the taxi service, which was the first one to put autonomous driving technology into commercial practice. Despite such promising developments, seeing surroundings robustly, perceiving objects in real-time, and acting safely on par with or even better than human drivers remain as the big challenges in achieving full autonomy.

Motion planning, which aims to generate a collision-free trajectory from the current location to the immediate destination for a driverless vehicle, is the key to acting safely. In the literature, there are four main categories of motion planning algorithms, graph search-based planners (Montemerlo et al. 2008), sampling-based planners (Kuwata et al. 2009; Karaman and Frazzoli 2011), interpolating curve planners (Labakhua et al. 2008), and numerical optimization approaches (Ziegler et al. 2014). Among them, sampling-based motion planning (SBMP) is becoming a mainstream motion planner in recent years (Ma et al. 2015; Banzhaf et al. 2017; Lim et al. 2018) given its capability of solving high-dimensional motion planning problems in a shorter time. Moreover, SBMP does not need precise obstacle geometry while other methods do.

The key challenge in adopting SBMP to autonomous driving is to ensure that the planner can respond accurately to the changing environment. The updating frequency of sensors on autonomous vehicles is usually set as $10Hz$ (Kuwata et al. 2008), which means that the planner needs to finish its decision in $100ms$ in order to be responsive to changes in the surrounding environment. According to (Ma et al. 2015), SBMP runs longer than 100ms in complex environments. To finish the planning task in $100ms$, a planner will have no choice but to sample fewer points, which may fail to find a feasible solution. Even if a motion planner can finish its task in time, it may bring other issues. For example, when a planner always tries to find an optimal path with the minimum cost, e.g., the shortest traveling time, or highest fuel efficiency, it may lead to jerky velocity over different planning time windows, sharp lane changes, and ride-line driving. Such actions will sacrifice the comfort of passengers. Moreover, the generated driving behavior may disturb other drivers and may cause accidents.

In this paper, we tackle the aforementioned issues in SBMP through a novel learning framework. Specifically, we first aim to allow motion planning to "see" ahead through predicting the intention of surrounding vehicles because the prediction will allow a motion planner to (1) have a longer time for planning and (2) generate smooth velocity over a longer time period. Then, we propose a new bias sampling approach with prediction and imitation learning, which can help generate a smooth and collision-free trajectory in a shorter time. Our main contributions are summarized as follows:

- Firstly, we refine the existing automatic labeling strategy for data preprocessing to correctly extract driving scenar-

*The authors contributed equally to this work.

ios and propose the first 2-stage prediction model that greatly improves the accuracy of prediction on vehicles' intentions.

- We leverage conditional variational autoencoder (CVAE) to design a new sampling strategy based on imitation learning and the prediction result so that only sample points near the human-driving trajectory will be sampled. The proposed sampling strategy will lead to faster motion planning and help generate a smooth and collision-free trajectory that is on par with and even better than the trajectory generated by human drivers.
- Finally, we evaluate our model and compare the performance with others, from the perspectives of the prediction accuracy, the success rate of finding collision-free trajectory, the computation time of planning, and the quality of trajectory.

The rest of the paper is organized as follows. In Section 2, we first review the general process of the SBMP algorithm and briefly introduce our framework. Then, we present our 2-stage intention prediction model and evaluate its performance in Section 3. Next, in Section 4, we leverage the prediction results and imitation learning to design a new sampling strategy. Finally, we evaluate our sampling strategy in Section 5 and conclude the paper in Section 6.

2 The Background and New Framework

In this section, we first explain the basic modules of SBMP, then present the proposed learning framework for SBMP.

A vehicle's state at any given time is defined as a vector $[p_x, p_y, v, \theta] \in \mathbb{R}^4$ where p_x and p_y are the coordinates of the center of the vehicle's rear axle, v is the instantaneous velocity, and θ is the heading angle of the vehicle with respect to the road direction (y-axis). SBMP aims to find the best trajectory that connects a vehicle's initial state, x_{init}, to any possible goal state, x_{goal}, in a goal region, $\mathcal{X}_{goal}$. To this end, there are many intermediate states, x_s, which are on the feasible trajectories. Each of such intermediate states is defined as a sample point. SBMP generally consists of sampling phase and planning phase introduced as follows.

- ***Sampling phase*:** This procedure discretizes the state space and generates the sample points. Although the uniform sampling strategy is commonly used, many bias sampling strategies have been proposed, such as goal biasing (LaValle 2006), bias Gaussian sampling (Kuwata et al. 2009), and informed sampling (Gammell, Srinivasa, and Barfoot 2014). The key advantage of bias sampling is that, by eliminating unnecessary sample points, a bias sampling scheme may reduce the search space and speed up the planning algorithm.
- ***Planning phase*:** Given the sample points, the obstacle information, and the differential constraints, this procedure generates a collision-free roadmap (Kavraki, Kolountzakis, and Latombe 1996) or tree (Janson et al. 2015) connecting x_{init} and x_{goal}. If a tree is constructed in this phase, the algorithm stops because the final trajectory can be obtained directly. If a roadmap is constructed where there are several feasible trajectories, graph search algorithms, such as Dijkstra (Dijkstra 1959), A* (Hart, Nilsson, and Raphael 1968), D* (Stentz 1997), can be applied to find the optimal one connecting x_{init} and x_{goal}.

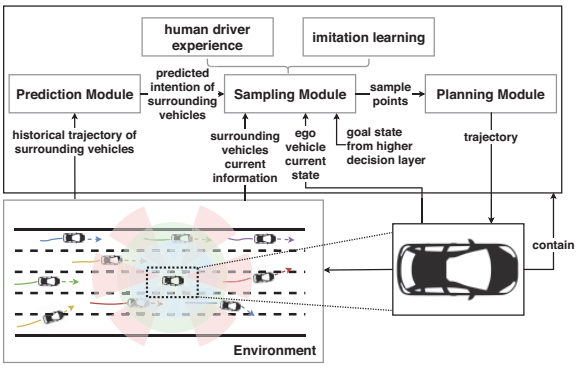

Figure 1: Framework of Proposed Motion Planning.

Among aforementioned two procedures in SBMP, sampling strategy in the sampling phase plays the key role in quickly finding a trajectory. In this paper, we design a new bias sampling strategy to reduce the computation time in the planning phase by integrating prediction and imitation learning. As shown in Fig. 1, our proposed SBMP includes four modules, namely, prediction module, imitation learning module, sampling module, and planning module, respectively, where we make contributions to the first three modules.

3 Prediction of Vehicle's Intention

In this section, we introduce the prediction module to allow the planning algorithm to see ahead, optimize the trajectory and relax the constraints on its computational time. Specifically, we first present an overview for the prediction module in Section 3.1. Next, in Section 3.2, we analyze existing automatic labeling schemes and propose our own scheme. Then in Section 3.3, we present details of our learning model with the training strategy, and conduct numerical experiments that validate the advantages of the proposed model.

3.1 Overview of the Prediction Module

The objective of the prediction module is to accurately predict the intention of a vehicle based on its history trajectory. Many deep learning models have been proposed for prediction in the past few years (Su et al. 2018). In general, existing learning models take a vehicle's past features as inputs, including position, velocity, acceleration, etc. On the other hand, the output of the models can be a classification of future behavior, such as car following, lane-changing-left and lane-changing-right in a recent study (Su et al. 2018).

Despite the promising results of existing deep learning-based prediction models, we notice that the accuracy is usually below 90% for lane changing (Su et al. 2018), which could compromise the path planning in autonomous driving. This is because an unexpected lane changing of a vehicle may affect the safety of neighboring vehicles and may in-

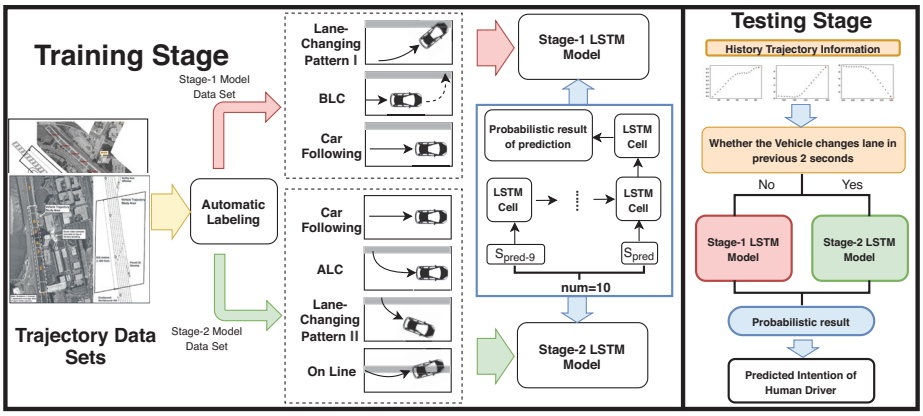

Figure 2: Overview of training process and testing process of our prediction model.

terfere the planned path of other vehicles. To improve the accuracy, we refine the prediction module as follows:

- We classify the history trajectory of a vehicle into different stages and use one learning model to predict in each stage. This feature is based on our analysis on the existing vehicle trajectory datasets, in which we can identify different moving behaviors before, during, and after changing lanes.
- To train different models for different stages, we develop a novel automatic labeling scheme so as to train different models using different labeled data.

In this paper, we demonstrate the proposed design using two stages: (1) regular, and (2) after-crossing-line (vehicle crosses the line in previous 2 seconds). As shown in Fig. 2, for the first stage, we consider three scenarios: (1) typical car following, (2) car following before lane changing (BLC), and (3) lane changing pattern I (i.e., before crossing the line between two lanes). For the second stage, we consider the history trajectory from the time that the vehicle crosses the line to a certain time after it crossed the line. In this stage, we consider four possible scenarios: (1) typical car following, (2) car following after lane changing (ALC), (3) lane changing pattern II (i.e., immediately after crossing the line), and (4) driving on line. In the next subsection, we will discuss how to automatically assign labels to vehicles with different timestamps in a dataset.

3.2 Automatic Labeling

Since a typical trajectory dataset contains a large number of positions of vehicles, it is crucial to design automatic labeling scheme to classify the state of each vehicle over time. In the literature, there are mainly three types of automatic labeling schemes.

The first type of schemes (Deo and Trivedi 2018) identifies the time that a vehicle is crossing the line between two lanes, and then specifies the state of vehicle as lane-changing if and only if the vehicle is within t seconds before or after the crossing event, where t is a predefined threshold as shown in Fig. 3 (a). A major problem of this type of

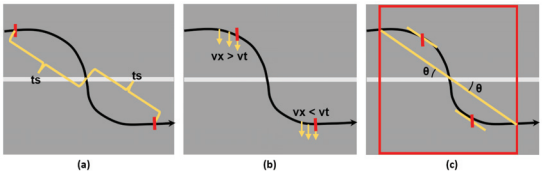

Figure 3: Illustration of three types of automatic labeling schemes. Positions of vehicle between two red vertical line segments are labeled as lane-changing.

schemes is that, with a fixed t, some car following cases may be wrongly labeled as lane-changing, as shown in Fig. 4 (a).

The second type of schemes (Nie et al. 2016; Scheel et al. 2018) assumes that a vehicle starts lane changing if its lateral velocity towards an adjacent lane exceeds a threshold v_l in N consecutive time steps before line crossing, and it ends lane changing if it crosses the line or its lateral velocity is lower than v_l after line crossing, as shown in Fig. 3 (b). Its main issues are that a vehicle may continue lane-changing after the cross-over or change its steering angle several times during lane-changing , as shown in Fig. 4 (b) (c).

A recent study in (Su et al. 2018) proposed the third type of schemes, which first identifies two positions of vehicle, corresponding to 2 seconds before and after line crossing, respectively. Next, the two points will be used to create a straight line, whose angle with the line is θ, shown as yellow line in Fig. 3 (c). Finally, it attempts to find two parallel tangent lines that have the nearest tangent points to the crossing point, and will use these two points as the starting and end points for lane changing. Although this scheme leads to better prediction performance, it also suffers the inaccurate labeling issue when a vehicle changes its steering angle multiple times during lane changing, as shown in Fig. 4 (b) (c). Moreover, this scheme cannot find the desired tangent line in some cases, as shown in Fig. 4 (d) (e).

Besides the above schemes for automatic labeling, most existing automatic labeling schemes will skip the cases in which a vehicle is driving along the line during lane changing, shown in Fig. 4 (f), which is a common situation.

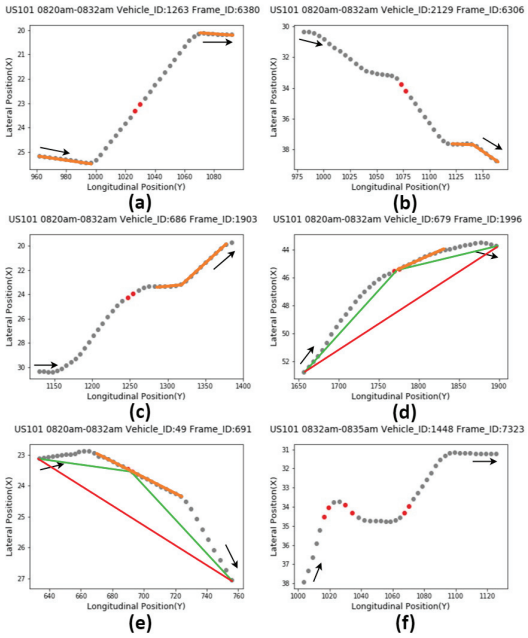

Figure 4: Some cases in real trajectory data. Each scatter point indicates a position in the trajectory. Each yellow line segment indicates positions with possible wrong labels.

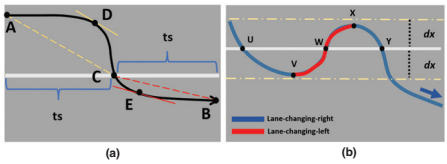

Figure 5: Illustration of our proposed automatic labeling method on lane-changing cases.(a) Regular Lane-changing. (b) Lane-changing with on line driving.

To improve the prediction accuracy, we propose a novel automatic labeling scheme as follows. Similar to the scheme in (Su et al. 2018), we first identify two points A and B, corresponding to t seconds before and after line crossing, respectively. Next, we draw two lines to connect each point to the line crossing point C, as shown in Fig. 5 (a). In this manner, we can find two angles θ_1 and θ_2. We then find one tangent line using angle θ_1 that leads to a tangent point D that is nearest to A, and let D be the starting point of lane changing. Similarly, we use θ_2 to find another tangent line that leads to a tangent point E that is nearest to B, and let E be the end point of lane changing.

Based on the above procedure, our automatic labeling scheme provides two levels of labels. In the first level, we specify the classical labels. In the second level, we provide labels that match with our prediction needs. Specifically, in a regular lane changing case, such as Fig. 5 (a), positions before D and after E are labeled as car following, and the positions between D and E are labeled as lane-changing-right for the first level. In the second level, we consider positions between A and D as BLC, positions between D and C as lane-changing pattern I, between C and E as lane-changing pattern II, and between E and B as ALC. For the on-line driving case, we first specify the second level label as on-line drive if (1) the vehicle crossed line multiple times in a short period t_x and (2) the maximum distance to the line is smaller than a threshold d_x. For example, in Fig. 5 (b), positions between U and Y are labeled as on-line driving. Next, we identify some turning points and determine the first level labels. For example, in Fig. 5 (b), positions before V and after X are labeled as lane-changing-right, and the positions between V and X are labeled as lane-changing-left.

3.3 2-Stage Model, Training Strategy, and Numerical Results

As shown in Fig. 2, our prediction module consists of two learning models. We choose Long Short-Term Memory (LSTM) architecture for both of them, each with same network settings as (Su et al. 2018). The prediction module will take the lateral and longitude position, velocity, acceleration, heading angle, and lane_ID of past 10 frames as inputs for each of the adjacent vehicles and output their intentions.

To train the two models, we use 6 sequences of trajectory data in two public datasets: NGSIM US-101 (Federal Highway Administration 2007) and I-80 (Federal Highway Administration 2006). Each sequence comprises trajectories of multiple vehicles in 15 minutes. We use data in the first 12 minutes for training and the last 3 minutes for testing, with all data labeled by our automatic labeling scheme. As shown in Fig. 2, we train the Stage-1 model with three possible cases, and the Stage-2 model with four cases. In reality, the frequencies of these cases are not the same and the car following case is the dominating one in a common traffic scenario. To improve the performance of the neural network, we increase the number of BLC and lane-changing pattern I cases so that the Stage-1 model can learn more about lane changing. The same strategy is also applied to train Stage-2 model. To train both models, we select learning rate to be 0.00125 and use softmax cross entropy as the loss function.

To evaluate the performance of our 2-Stage model, we conduct extensive experiments. Due to limited space, we only present the results from two experiments to compare the performance of our model with the Social-LSTM model (SLM) applied in (Su et al. 2018) using the the same labeled data for training and testing. In the first experiment, we utilize the testset with all the vehicles randomly selected, each at random time epoch. The results are summarized in Table 1, in which we can find that both the SLM and ours achieve high accuracy in different classification tasks. Moreover, our model achieves about 3% better accuracy in all cases. Since the prediction before and after lane changing is more important to our system, we conduct the second experiment, in which we choose vehicles randomly but only at time epoch when the vehicle is 4 seconds before or after the line crossing. As shown in Table 2, in such a more challenging scenario, we notice that the accuracy of the SLM decreases considerably, especially for the accuracy of car following. By comparison, our model still can achieve high accuracy for all three categories. These results clearly demonstrate the advantages and potentials of the proposed model.

Table 1: Comparison results on testing set with each testing scenario randomly selected in all testing scenarios.

Method	Real Label	Predict Label		
		Following	Left	Right
Social LSTM Model (Su et al. 2018)	Following	93.96%	2.43%	3.61%
	Left	11.72%	87.00%	1.28%
	Right	8.99%	1.33%	89.68%
2-stage Model	Following	**96.67%**	1.33%	2.00%
	Left	7.77%	**91.78%**	0.45%
	Right	7.56%	0.50%	**91.94%**

Table 2: Comparison results on testing set with each testing scenario randomly selected in ± 4 s interval of cross-over.

Method	Real Label	Predict Label		
		Following	Left	Right
Social LSTM Model (Su et al. 2018)	Following	73.20%	16.93%	9.87%
	Left	11.76%	86.28%	1.96%
	Right	9.50%	1.95%	88.55%
2-stage Model	Following	**83.03%**	10.36%	6.61%
	Left	7.73%	**90.97%**	1.30%
	Right	8.32%	1.60%	**90.08%**

4 Sampling with Imitation Learning and Prediction

Many works (Li, Song, and Ermon 2017; Kuefler et al. 2017) learn from human drivers' trajectory and teach autonomous vehicles to drive like human drivers. The question that we aim to answer is: can an autonomous vehicle drive even better than human drivers given the capability of "seeing" ahead spatially and temporally? To this end, we aim to find the most appropriate set of sample points including but not limited to the sample points near the human-driving trajectory and other sample points which will lead to a better trajectory than human-driving one with high probability. In other words, we need to learn the distribution of sample points on and near human-driving trajectory for any given environment, initial state, and goal state. We apply generative models, which are good at learning from the given data and generating new data, to learn such a distribution.

In the rest of this section, we first introduce the basics of CVAE. Then, we describe how to apply the CVAE model to our problem and its training process. Finally, we present how to integrate the prediction module and the well-trained CVAE model to generate sample points.

4.1 Basics of CVAE

Variational Autoencoder (VAE) (Kingma and Welling 2013) is a mainstream generative model. It can generate new samples that follow the same distribution of the training data. In this paper, we choose conditional variational autoencoder (CVAE) (Sohn, Lee, and Yan 2015), an extension of VAE, as the generative model. With sampled data x, denote the latent variable of x as z. Conditioned on c, we train CVAE to maximize the objective function (Sohn, Lee, and Yan 2015) for a given sample x:

$$E_{q_\phi(z|x,c)}[\log p_\psi(x|z,c)] - D_{KL}[q_\phi(z|x,c)||p_\psi(z|c)] \quad (1)$$

where ϕ and ψ are parameters of encoder and decoder functions respectively. Given sample x, we firstly utilize an encoder to capture the distribution of its latent variable z conditioned on c and approximates the distribution to a function $p_\psi(z|c)$. After decoding the latent variable z conditioned on c, we hope to maximize the expectation of log likelihood $\log p_\psi(x|z,c)$ in order to regenerate x.

The procedure of training a CVAE is to map the distribution over latent variable z to a known distribution $N(0,I)$ (Doersch 2016) and make generated $\bar{x}$ based on latent variable close to x as much as possible. Therefore, we let the encoder output two values, μ and σ^2. We minimize the KL divergence between $N(0,I)$ and $N(\mu,\sigma^2)$. Once trained, given c, we can sample from $N(0,I)$ to generate $\bar{x}$ conditioned on c.

4.2 Our Training Model

The network architecture used for training and sampling of CVAE is shown in Fig. 6. Besides the encoder and decoder networks, we need an additional convolutional neural network (CNN) to preprocess our 3D environment information, which involves current and future situations. The surrounding environment only contains the neighboring vehicles in a fixed range of distance. In the training phase, we use the ground truth to describe the future environment. Now we introduce c, x, loss function and the training process.

Condition c In our model, the condition c includes the environment information, the initial state x_{init}, and the goal state x_{goal}. We define the environment information in every 100ms as a frame. For each frame, we encode it into a occupancy matrix as follows. Let the width of the lane be w, the lateral distance of the ego vehicle to the lane center be d, the ego vehicle position be $[p_x, p_y]$, and the neighboring range be r. Considering the camera and Lidar sensing ability, we set $r = 150$ feet. If the ego vehicle is in the right of the lane center, d is negative and vice versa. We draw a rectangle with length $2 * r$ and width $3 * w$. The center of this rectangle is $[p_x - d, p_y]$. This rectangle is divided evenly into several small grids. As the vehicle size is usually around 9 feet * 16 feet, the size of a grid is set as 4 feet * 6 feet which is smaller than the vehicle size to make it sensible to the vehicle movement even in a short time period. Therefore, this rectangle is divided into $\frac{3*w}{4} \times 50$ grids. Let $(pos_i, pos_j),\ i < \frac{3*w}{4},\ j < 50$ be the center of each grid.

Given the 1s (10 frames) predicted trajectory of surrounding vehicles, the raw information is encoded in a 3D occupancy matrix $M_{env} \in \mathbb{R}^{m \times n \times 10}$ as follows:

$$M_{new}[i][j][t] = \begin{cases} 1, if\ (pos_i, pos_j)\ is\ occupied\ at\ time\ t \\ 0, if\ (pos_i, pos_j)\ is\ free\ at\ time\ t \end{cases}$$

where $m = \frac{3*w}{4},\ n = 50$.

This 3D occupancy grid matrix is input to the CNN model for feature extraction. The output is a k-dimension vector $v_{env} \in \mathbb{R}^k$. Therefore, the condition variable can be represented as a $(k+8)$-dimension vector which contains $v_{env}, x_{init}, x_{goal}$ as shown in the most left part in Fig. 6.

Variable x As the environment information in c has the temporal dimension, x also needs to have time-sequential sample points corresponding the frame in the 3D environment matrix. The variable x is a vector $x \in \mathbb{R}^{10*4+1*4}$ which

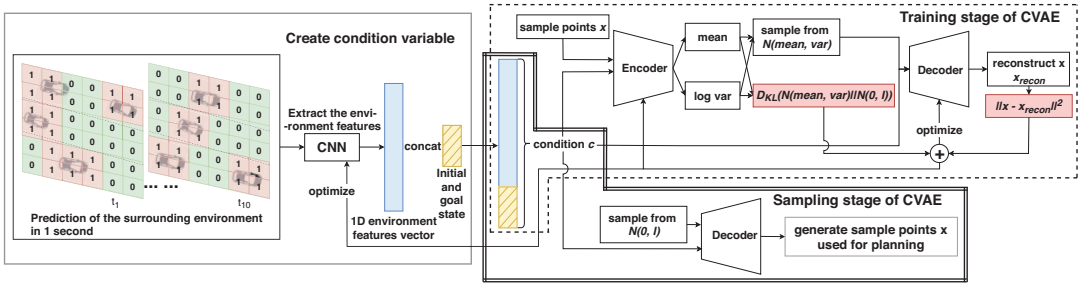

Figure 6: Architecture of our model applying CVAE.

contains 10 states corresponding to the 10 future frames and 1 state out of the predicted range. Therefore, x can be represented as $x = [x_1, x_2, ..., x_{10}, x_s]$ where x_s is a state out of the predicted range but before the goal state. x_1 is the state of the ego vehicle in the 1st predicted frame.

We take 40 frames (4 s) trajectory and its corresponding environment as a training case which means human driver finishes the trajectory in 4 s. Lots of training cases of different lane-changing scenarios are retrieved evenly. If we use these training cases to train our model directly, the model can only be applied to the cases where the travelling time must be around 4 s. Obviously, it is unpractical. To handle this problem and augment the training data, one lane-changing trajectory is used to generate multiple training cases with a different initial state but the same goal state.

Loss function We use the KL divergence, an important term of the CVAE objective function, as one part of the loss function. Another part is the reconstruction error between the input x and the output of the decoder $\bar{x}$. The final loss function is:

$$||\bar{x} - x||^2 + D_{KL}[q_\phi(z|x, c)||p_\psi(z|c)]. \quad (2)$$

The loss function penalizes the large divergence between the distribution of latent variable $N(\mu, \sigma^2)$ and $N(0, I)$ for approximating $N(\mu, \sigma^2)$ to $N(0, I)$ where μ and σ^2 are output by encoder. Besides, it encourages $\bar{x}$ to be similar with x as much as possible.

Training process The training process is illustrated in Fig. 6. The 3D environment matrix is transformed to a 1D vector by CNN. This vector is concatenated with the initial and the goal state to constitute the condition c. Then the sample points x are concatenated by the condition c and mapped to the latent space by the encoder. The encoder outputs two values, one is mean μ and the other is logarithmic variance $\log \sigma^2$ so that we can sample from $N(\mu, \sigma^2)$ which is the latent variable z. z concatenates with the same condition c. The decoder projects this vector from the latent space to get $\bar{x}$. The loss value is calculated using the Eq. (2) to optimize CNN, encoder network and decoder network.

4.3 Our Sampling Model

After training, the decoder is capable of using $N(0, I)$ and c to generate $\bar{x}$ as shown in the sampling stage in Fig. 6, where c can be obtained by the same way introduced in the training process, for given environment frames, initial state, and goal state. The initial and goal state can be obtained from the location system and higher decision layer, respectively. Therefore, to obtain c, an input of the decoder in the sampling stage, we only need to construct the environment frames, especially the future frames.

For each future frame, we need every surrounding vehicle's location information and heading angle to complete our 3D environment matrix. The location of a vehicle in each frame is calculated using the velocity and heading angle. We can use the current instantaneous velocity to estimate the velocity in the next 1 second. The heading angle may change a lot in 1 second, especially, in lane-changing cases. We leverage the output of our prediction module to estimate the heading angle. Since our prediction module can provide probabilities of lane-changing-left P_{left}, lane-changing-right P_{right}, and car following P_{follow} for each surrounding vehicle, we sort these three probabilities and define the largest one as P_m where the subscript m represents the major intention. The second largest one as P_s where the subscript s represents the secondary intention. We define the normal range of heading angle for lane-changing-left, lane-changing-right, and car following as $[\theta_{follow-}, \theta_{follow+}]$, $[\theta_{left-}, \theta_{left+}]$ and $[\theta_{right-}, \theta_{right+}]$ respectively, according to the statistics of NGSIM real data. For further estimation, let $\theta_{follow} = \frac{\theta_{follow-}+\theta_{follow+}}{2}$, $\theta_{left} = \frac{\theta_{left-}+\theta_{left+}}{2}$ and $\theta_{right} = \frac{\theta_{right-}+\theta_{right+}}{2}$. Denote the current heading angle θ, estimated heading angle θ_{new}. We estimate the heading angle of a vehicle as follows.

$$\theta_{new} = \begin{cases} \dfrac{\theta + \theta_m}{2}, & \theta \notin [\theta_{m-}, \theta_{m+}], P_m \geq 80\% \\ \dfrac{\theta + \theta_m}{2} + \eta * \theta_s, & \theta \notin [\theta_{m-}, \theta_{m+}], P_m < 80\% \\ \theta + \eta * \theta_s, & \theta \in [\theta_{m-}, \theta_{m+}], P_m < 80\% \end{cases}$$

where η is the weight.

5 Experiments

To evaluate and compare our sampling strategy with other sampling strategies, we adopt FMT* (Janson et al. 2015) as the planning algorithm and conduct data-driven simulations, in which we apply the cost function and trajectory generation method suggested in (Webb and van den Berg 2013). We focus on the success rate of finding a collision-free trajectory, computation time of the planning algorithm,

Table 3: Success rate versus replanning time interval.

Replanning time interval	200ms	400ms	600ms	800ms	900ms
uniform	0%	0%	0%	0%	0%
bias Gaussian	70%	60%	30%	0%	0%
ours	100%	100%	100%	90%	70%

Table 4: Computation time in different replanning steps.

time step (ms)	bias Gaussian (ms)	ours (ms)
0	158	19
300	199	254
600	217	236
900	945	235
1200	279	180
1500	203	167
1800	76	199
2100	55	34
2400	8	32
2700	11	2
3000	5	arrived
3300	arrived	-
average time	196	136

the travelling time of driving over the trajectory, and the acceleration variation. Here, the acceleration variation represents the smoothness (Ziegler et al. 2014), and a smaller value implies that passengers feel more comfortable. Due to limited space, we compare our scheme with two strategies: uniform sampling and bias Gaussian sampling (Kuwata et al. 2009), which is the default bias sampling strategy applied in autonomous driving and has been tested on road.

The CNN is designed using 16 kernels with size 5. The encoder and decoder have the same architecture which contains two full connected layers where the first one has 512 hidden units and the second one has 128 hidden units. They share the same CNN. As NGSIM US-101 dataset (Federal Highway Administration 2007) includes 3 sub datasets, we use two for training and one for testing. We utilize Adam optimizer with a learning rate of 0.0001 and a batch size of 256 to train the model for 500 epochs. Since both prediction module and imitation learning module are trained with NGSIM dataset, we select lane-changing cases from the intersection of test dataset to evaluate the performance. Here we randomly select one lane-changing case for illustration. The initial state is before lane-changing and the goal state is after lane-changing. The travelling time for human driver is 4 s. There are 11 surrounding vehicles at the initial state. In this following, we refer the computation time used for planning algorithm simply as computation time.

We first compare the success rate using different sampling strategies under different replanning time intervals. The number of sample points is set as 1000. We run FMT* algorithm under each sampling strategy 10 times. From Table 3, we can see that our sampling strategy performs much better than other ones. With the prediction module, our sampling strategy has higher chance to provide sample points which lead to collision-free trajectory.

Then, we compare the acceleration variation using different sampling strategies. Under each given replanning time interval, we choose the trajectory with the shortest travelling time for each sampling strategy. We calculate the acceleration for each trajectory. As shown in Fig. 7, the acceleration variation of trajectory generated by our sampling strategy is always smaller than that by bias Gaussian sampling strategy for all replanning time intervals, which demonstrates that our sampling strategy helps generate a more smooth trajectory than bias Gaussian. Moreover, we find that a longer time interval leads to a smaller variance for both strategies, which implies that a longer replanning interval alleviates the jerky velocity problem and brings more comfort to passengers.

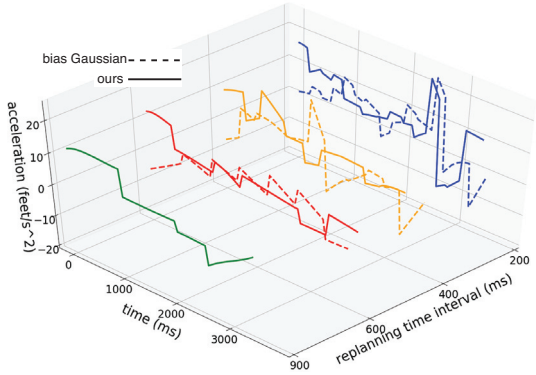

Figure 7: Acceleration variation with different replanning time intervals.

Finally, we consider the travelling time and computation time when FMT* succeeds under both bias Gaussian and our sampling strategies. We set the replanning time interval as 300ms and select the trajectory with the shortest travelling time for each sampling strategy. The results in Table 4 show that our strategy takes less time (3 s) than a human driver (4 s) and bias Gaussian (3.3 s) to reach the goal state. Table 4 shows that the computation time for both of them first increase then decrease as the ego vehicle approaching the goal state, which is mainly caused by the varying of search space over time. The results show that the average computation time of replanning with bias Gaussian sampling strategy is much longer than ours, and that our computation time is always less than the replanning time interval 300ms.

6 Conclusion

In this paper, we have proposed a new learning framework to enhance SBMP in autonomous driving, which only samples the points leading to a high-quality collision-free trajectory. Specifically, we proposed new labeling and training models to improve the accuracy of predicting a vehicle's intention. We then integrated the prediction of surrounding vehicles and imitation learning to generate the collision-free sample points near the human-driving trajectory that are used for the planning algorithm. Data-driven experiments show that our sampling strategy not only accelerates the computation, but also alleviates the jerky velocity. Moreover, our sampling strategy can generate a trajectory that leads to less driving time than that by human drivers.

7 Acknowledgements

This work was partially supported by NSFC-Guangdong Joint Fund under project U1501254, by Hong Kong Research Grant Council under GRF project 11204917, and by NSF under Grant CNS-1730325.

References

Banzhaf, H.; Palmieri, L.; Nienhuser, D.; Schamm, T.; Knoop, S.; and Zollner, J. M. 2017. Hybrid curvature steer: A novel extend function for sampling-based nonholonomic motion planning in tight environments. In *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, 1–8.

Deo, N., and Trivedi, M. M. 2018. Multi-modal trajectory prediction of surrounding vehicles with maneuver based lstms. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, 1179–1184.

Dijkstra, E. W. 1959. A note on two problems in connexion with graphs. *Numerische mathematik* 1(1):269–271.

Doersch, C. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Federal Highway Administration. 2006. US highway 80 dataset.

Federal Highway Administration. 2007. US highway 101 dataset.

Gammell, J. D.; Srinivasa, S. S.; and Barfoot, T. D. 2014. Informed rrt*: Optimal sampling-based path planning focused via direct sampling of an admissible ellipsoidal heuristic. In *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2997–3004.

Hart, P. E.; Nilsson, N. J.; and Raphael, B. 1968. A formal basis for the heuristic determination of minimum cost paths. *IEEE transactions on Systems Science and Cybernetics* 4(2):100–107.

Janson, L.; Schmerling, E.; Clark, A.; and Pavone, M. 2015. Fast marching tree: A fast marching sampling-based method for optimal motion planning in many dimensions. *The International Journal of Robotics Research* 34(7):883–921.

Karaman, S., and Frazzoli, E. 2011. Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research* 30(7):846–894.

Kavraki, L. E.; Kolountzakis, M. N.; and Latombe, J.-C. 1996. Analysis of probabilistic roadmaps for path planning. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 4, 3020–3025.

Kingma, D. P., and Welling, M. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kuefler, A.; Morton, J.; Wheeler, T.; and Kochenderfer, M. 2017. Imitating driver behavior with generative adversarial networks. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, 204–211.

Kuwata, Y.; Fiore, G. A.; Teo, J.; Frazzoli, E.; and How, J. P. 2008. Motion planning for urban driving using rrt. In *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1681–1686.

Kuwata, Y.; Teo, J.; Fiore, G.; Karaman, S.; Frazzoli, E.; and How, J. P. 2009. Real-time motion planning with applications to autonomous urban driving. *IEEE Transactions on Control Systems Technology* 17(5):1105–1118.

Labakhua, L.; Nunes, U.; Rodrigues, R.; and Leite, F. S. 2008. Smooth trajectory planning for fully automated passengers vehicles: Spline and clothoid based methods and its simulation. In *Informatics in control automation and robotics*. Springer. 169–182.

LaValle, S. M. 2006. *Planning algorithms*. Cambridge university press.

Li, Y.; Song, J.; and Ermon, S. 2017. Infogail: Interpretable imitation learning from visual demonstrations. In *Advances in Neural Information Processing Systems 30*. Curran Associates, Inc. 3812–3822.

Lim, W.; Lee, S.; Sunwoo, M.; and Jo, K. 2018. Hierarchical trajectory planning of an autonomous car based on the integration of a sampling and an optimization method. *IEEE Transactions on Intelligent Transportation Systems* 19(2):613–626.

Ma, L.; Xue, J.; Kawabata, K.; Zhu, J.; Ma, C.; and Zheng, N. 2015. Efficient sampling-based motion planning for on-road autonomous driving. *IEEE Transactions on Intelligent Transportation Systems* 16(4):1961–1976.

Montemerlo, M.; Becker, J.; Bhat, S.; Dahlkamp, H.; Dolgov, D.; Ettinger, S.; Haehnel, D.; Hilden, T.; Hoffmann, G.; Huhnke, B.; et al. 2008. Junior: The stanford entry in the urban challenge. *Journal of field Robotics* 25(9):569–597.

Nie, J.; Zhang, J.; Wan, X.; Ding, W.; and Ran, B. 2016. Modeling of decision-making behavior for discretionary lane-changing execution. In *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, 707–712.

Scheel, O.; Schwarz, L.; Navab, N.; and Tombari, F. 2018. Situation assessment for planning lane changes: Combining recurrent models and prediction. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2082–2088.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. In *Advances in Neural Information Processing Systems 28*. Curran Associates, Inc. 3483–3491.

Stentz, A. 1997. Optimal and efficient path planning for partially known environments. In *Intelligent Unmanned Ground Vehicles*. Springer. 203–220.

Su, S.; Muelling, K.; Dolan, J. M.; Palanisamy, P.; and Mudalige, P. 2018. Learning vehicle surrounding-aware lane-changing behavior from observed trajectories. *2018 IEEE Intelligent Vehicles Symposium (IV)* 1412–1417.

Webb, D. J., and van den Berg, J. 2013. Kinodynamic rrt*: Asymptotically optimal motion planning for robots with linear dynamics. In *2013 IEEE International Conference on Robotics and Automation*, 5054–5061.

Ziegler, J.; Bender, P.; Dang, T.; and Stiller, C. 2014. Trajectory planning for bertha — a local, continuous method. In *2014 IEEE Intelligent Vehicles Symposium Proceedings*, 450–457.