2017 IEEE Intelligent Vehicles Symposium (IV)
June 11-14, 2017, Redondo Beach, CA, USA

Master General Parking Skill via Deep Learning*

Yi-Lun Lin[1], Li Li[2], *Fellow, IEEE*,
Xing-Yuan Dai[3], Nan-Ning Zheng[4], *Fellow, IEEE*, Fei-Yue Wang[5], *Fellow, IEEE*

***Abstract*— Parking is one basic function of autonomous vehicles. However, parking still remains difficult to be implemented, since it requires to generate a relatively long-term series of actions to reach a certain objective under complicated constraints. One recently proposed method used deep neural networks(DNN) to learn the relationship between the actual parking trajectories and the corresponding steering actions, so as to find the best parking trajectory via direct recalling. However, this method can only handle a special vehicle whose dynamic parameters are well known. In this paper, we use transfer learning technique to further extend this direct trajectory planning method and master general parking skills. We aim to mimic how human drivers make parking by using a specially designed deep neural network. The first few layers of this DNN contain the general parking trajectory planning knowledge for all kinds of vehicles; while the last few layers of this DNN can be quickly tuned to adapt various kinds of vehicles. Numerical tests show that, combining transfer learning and direct trajectory planning solution, our new approach enables automated vehicles to convey the knowledge of trajectory planning from one vehicle to another with a few try-and-tests.**

I. INTRODUCTION

Autonomous vehicles are designed to complete most driving tasks that human drivers can do[1]. Among all these tasks, parking problem remains to be an essential one, not only because it is a common scenario that every driver meets in their daily life, but also because it reflects a fundamental relationship underlying all driving tasks: a relationship between control actions and trajectories under constraint of vehicle dynamics.

In general, solving the parking control problem can be defined as finding a control strategy or a series of detailed control actions that guides a vehicle to move from initial situation to the given final position and orientation.

*This work was supported in part by the National Natural Science Foundation of China (Grant No. 91520301). (Corresponding author: Li Li.)

[1]Yilun Lin is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences linyilun2014@ia.ac.cn

[2]Li Li is with Faculty of Department of Automation, Tsinghua University, China. Corresponding author li-li@tsinghua.edu.cn

[1]Xing-Yuan Dai is with the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China and University of Chinese Academy of Sciences daixingyuan2015@ia.ac.cn

[5]Nan-Ning Zheng is with Faculty of the Institute of Artificial Intelligence and Robotics (IAIR), Xi'an Jiaotong University, China nnzheng@mail.xjtu.edu.cn

[5]Fei-Yue Wang is with Faculty of the State Key Laboratory of Management and Control for Complex Systems, Institute of Automation, Chinese Academy of Sciences, China feiyue.wang@ia.ac.cn

Various approaches have been proposed to address this issue, one of the most widely used methods is trajectory planning method [2], [3]. It considers an equivalent formation of the parking control problem as finding a valid trajectory, which can be tracked by vehicle so as to move from the given initial position and orientation to the given final position and orientation. Noticing that, a certain parking control strategy will determine a trajectory, it is clear that as soon as valid trajectories are planned, steering actions can be decided under some constrains on action space. Usually, these approaches can be categorized into two kinds: indirect trajectory planning and direct trajectory planning.

By designing a reference parking trajectory, usually belongs to a set of special curves(e.g. polynomial curves [1], β-spline curves [4]) that an autonomous vehicle could approximately follow, indirect trajectory planning methods simplify the trajectory planning problem by curves' distinctive geometric properties.

Though such simplification makes the problem more tractable, it may violate the dynamic constrain of vehicles in certain situation, and results in a wider gap between the designed curves and the actual trajectories.

To fix this problem, direct trajectory planning method was first proposed in [5]. It enumerates all the possible parking trajectories that a vehicle can make and learns to set up a direct relationship between any initial/final state pair and the corresponding steering action as well as the parking trajectory. Based on this learning result, the autonomous vehicle will soon recall the desired steering action/parking trajectory, if an initial/final state pair is given; See Fig. 1 for an illustration.

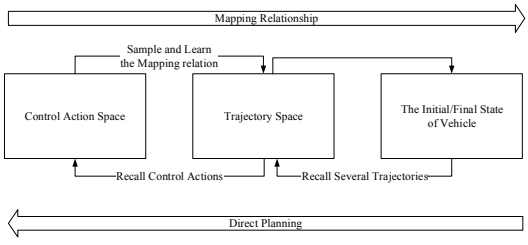

Fig. 1. The solution framework of direct trajectory planning for autonomous parking

The core of direct trajectory planning method is to establish the relationship between an initial/final state pair and the parking trajectory precisely. A neural network is proposed in [5] to achieve this goal. However, training such a network is computationally intensive since the action space increase

exponentially over the time length of trajectories, and huge amounts of data will be needed to make controller having an acceptable performance. Moreover, this method failed to learn general parking skill. Once the vehicle yaw dynamic has change, the trajectory-steering mapping relationship will be changed as well.

In this paper, we proposed a novel approach to tackle problems existing in previous method. Inspired by human drivers, our model will first gain a general understanding of trajectory-steering mapping relationship regardless of vehicle dynamic constraint, then learn the mapping relationship between vehicle, trajectory and steering actions given data of specific vehicle type. The model can be pre-trained on one kind of vehicle, then perform parking actions on the other kind of vehicles with additional few trainings. Combining transfer learning and direct trajectory planning solution, our new approach enables automated vehicles to master general parking skills and adapt different vehicles with a few try-and-tests.

We will illustrate the general idea of this method in *Section II*, propose a feasible implementation in *Section III*, then set up an experiment to test its performance in *Section IV*. Finally, we briefly conclude the paper in *Section V*.

II. Direct Trajectory Planning Using Parallel Learning

Drivers make a lot efforts to learn driving at the beginning, but can easily learn to drive different vehicles with similar operating characteristics in different situations once they master the skill. A plausible hypothesis of human driving behaviour is that, the very beginning training built up a general understanding of the relation between control actions and its possible trajectories, then with the guidance of this understanding, human drivers establish a mapping mechanism of steering-possible trajectories for a specific vehicle type with try-and-tests. In parking scenario, drivers predict a trajectory according to target position first, then use the mapping mechanism to find the optimal control action.

Inspired by the way human learn to drive, direct trajectory planning methods were proposed to mimic this plan-then-execute behavior pattern. Constrained by computational power, previous work [5] used ideal model such as bicycle model for trajectories sampling. Since the change of vehicle dynamics, weather/road conditions, the number of passengers and other factors can all affect the values of parameters and thus influence the accuracy of the ideal model, previous methods can only be used in limited circumstances, otherwise the sampled and stored mapping relation will not hold in practice.

With development of computational power, using data-driven approach to deal with problems used to be handled by model-based methods has become a trend in intelligent vehicle field. However, traditional data-driven approaches are extremely data hungry and not be general[6]. To enhance the generality of direct trajectory planning solution without severely constrained by the amount of the data, we apply parallel learning technique on vanilla framework.

Parallel learning is a method first proposed in [7], which combines data-driven approach with predictive learning and transfer learning. By building a parallel system upon domain knowledge and existing data, parallel learning methods can learn from limited size of dataset without compromising the performance of data-driven models.

Compared with model-based method, parallel learning is easier to adapt the change of dynamics and environments. Instead of making assumption on the actual dynamic models and what factors may affect its behaviour, such methods focus on the parameters that can be measured, make the best guest about corresponding results depend on given dataset, and continuously improve their performance by obtaining records needed, just as human driver will do under limited observation.

Another advantage of parallel learning is its transferability. Since the dynamic of vehicles can be various for different types or in different environments, the steering-trajectory mapping relationship changed from time to time, and the data collected can be outdated easily. Transfer learning[8] can be beneficial in such cases. Given a source domain D_S and learning task T_S, a target domain D_T and learning task T_T, transfer learning aims to help improve the learning of the target function $f_T(\cdot)$ in D_T using the knowledge in D_S and T_S, where $D_S \neq D_T$, or $T_S \neq T_T$. We can refer the trajectories under constraint of specific types of vehicles' dynamic to D_S, and the corresponding control actions to park in given position and orientation to T_S. Then D_T and T_T is the trajectories and actions of other types of vehicles, or vehicles under different environment. By forming knowledge as neural network, parallel learning takes benefit from its representation ability to gain a good transferability.

In the context of parking scenario context, we build a parallel system by distilling knowledge from actions-trajectories pairs collected from both specific type of vehicle and a theoretical model. The knowledge will be presented as the structure of a generative model, who takes control sequence as input and output the corresponding trajectory. Mixed data generated from both actual and artificial system will be used to update the initial knowledge, and a general parking skill can then be drawn from it, which results in control actions and corresponding trajectories. Such process will cycle until it converges to attain an acceptable performance. See Fig. 2 for a visualization.

To implement this framework, we use a deep neural network to learn the mapping relationship from velocity and steering sequence to trajectories under complex dynamic constraints, and try to generate trajectories corresponding to actions that have not happened. Deep neural network is a non-linear function approximator capable of modelling complex dynamics in an end-to-end style without making a strong assumption on the underlying models[9]. It also exhibits a phenomenon that, it tends to learn first-layer features in a very general way, while the features computed by the last layers depend greatly on the chosen dataset and task[10]. These features make it capable to build a general parking model. The network will be first pre-trained on the

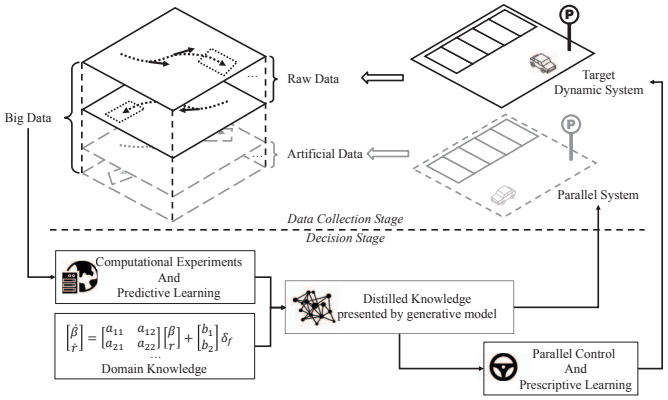

Fig. 2. A parallel learning framework to solve parking problem

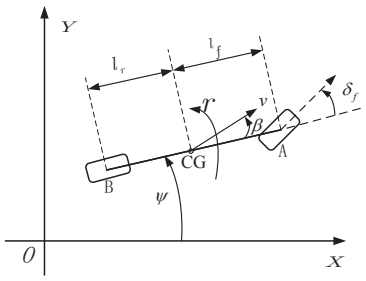

Fig. 3. A simplified model of the front steering vehicle when vehicle going forward

source domain, which has sufficient records, then transfer the knowledge to the target type. Once the performance of the network exceeds threshold, we can use actions-trajectories pairs generated by the network to perform a feed-forward control.

III. Implementation of General Parking Model

A. Vehicle Dynamics Variables and Trajectories Sampling

Without losing generality, we consider the parking problem of front steering vehicles. The parking problem of full steering vehicles can be solved in a similar manner [1]. We characterize vehicle's dynamics, actions and corresponding trajectory by the following variables listed in Table I

TABLE I

NOMENCLATURES USED TO DESCRIBE VEHICLE DYNAMICS

Symbol	Meaning
$X-Y$	The coordinate system
β	Vehicle side slip angle which is between the heading of the vehicle and the velocity
ψ	The angle from the X-axis to the longitudinal axis of the vehicle body AB.
r	Yaw rate of the vehicle, $r=\dot{\psi}$
δ_f	Front steering angle
$\delta_{\max}$	The maximum absolute value of the front steering angle
v	The velocity of the vehicle
m	Mass of the vehicle
I	Inertial moment around the vertical axis through CG
l_f	Distance from point A and point CG
l_r	Distance from point B and point CG
l_{fo}	Front overhang, the lengths of vehicle which extend beyond the point A
l_{ro}	Rear overhang, the lengths of vehicle which extend beyond the point B
l_w	Overall width of vehicle
c_f	Stiffness coefficients of front tire
c_r	Stiffness coefficients of rear tire

We consider the situation of going forward only in this paper, while going backward can be dealt with in a similar manner [1]. Suppose a vehicle is operating on a flat surface, the motion of the vehicle can be simplified as Fig. 3.

As shown in Fig. 3, reference point CG is the vehicle's **center of mass**, and its coordinate value (x,y) represents the position of the vehicle. **Velocity** v is defined at the reference point **CG**. **Heading Angle** ψ refers to the angle from the X-axis to the longitudinal axis of the vehicle body **AB**. **Slide-slip Angle** β is the angle from **AB** to the direction of the vehicle velocity.

The origin of the coordinate system is set at the **CG** of the vehicle at the initial time, and the direction of the positive X-axis is assumed to point to the head of the vehicle.

To make the problem tractable for neural network, we discretize the control variables δ_f along the time axis. All the possible parking trajectories could be generated by assigning distinct steering sequences $\delta_f^N=\{\delta_f(1),\ldots,\delta_f(N)\}$ for N consequent time segments. During each time segment, the steering angle is chosen from a preselected steering set $S_\delta=\{\delta_1,\delta_2,\ldots,\delta_K\}$ and remains constant within this segment, K is the number of steering angles we considered, and for all angles in S_δ has $\delta_i\in[-\delta_{\max},\delta_{\max}]$. Therefore, the control action space is sampled into a set I consisting of K^N steering angle sequences as its elements.

Once a sufficient number of samples is obtained, a model approximates the mapping relationship between actions and trajectories can then be learned, then used to enumerate allowable control sequences and corresponding trajectories.

B. Using Deep Neural Network to Learn General Parking skill

A computational model is proposed to discover the intricate structure underlying the relationship of vehicle dynamics, control actions and parking trajectories. The mean component of this model is a deep neural network composed of multiple processing layers. Since deep neural network can learn representations of data with multiple levels of abstraction [11], we take its advantage to distil a knowledge of general relationship between steering actions and parking trajectories, then specify it to fit the certain type of vehicle and environment.

To achieve this purpose, a hierarchical generative model is set. As visualized in Fig. 4, control sequences during the parking trajectory will be fed into this model. The first few layers which marked as **general hidden layer(s)**, will learn the general relationship between control actions and trajectories, and generate a general trajectory corresponding to given velocities and control actions. Then this trajectory

will flow through another multiple layers network marked as **specific hidden layer(s)** to generate a specific parking trajectory under specific physical constrains of vehicle dynamics.

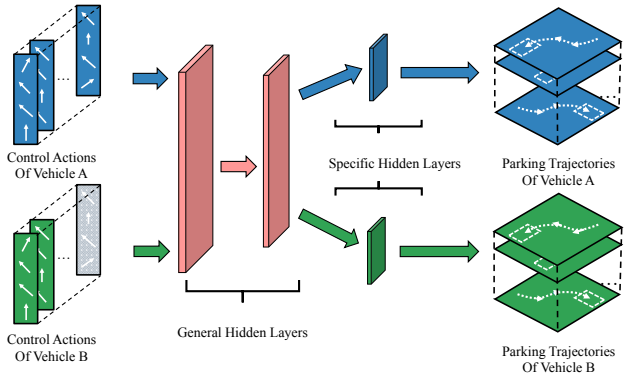

Fig. 4. Structure of Neural Network used to generate trajectories

In Fig. 4, vehicle A refers to the source domain and vehicle B the target domain. While source domain contains sufficient records for the model to learn the relationship, the target domain is what we are actually interested in yet lack of sufficient data. We apply the transfer learning technique to handle this issue. In initial stage, both general and specific hidden layers will be trained by the actions-trajectories pair of vehicle A, letting the neural network to obtain multiple levels of abstraction about the mapping relationship. Specific hidden layers will then be trained on data of vehicle B. The output of the network is a generative trajectory corresponding to the input action.

The performance of trajectory generative model are measured by the Euclid distance from the real position for generative coordinate and the rectilinear distance for generative orientation. The objective function of the entire model is

$$L = \gamma_1 \|(X,Y) - (\tilde{X},\tilde{Y})\|_2 + \gamma_2 \|\Psi - \tilde{\Psi}\|_1 \qquad (1)$$

γ_1, γ_2 is loss weights used to specify relative importance of generative coordinate and orientation accuracy. These hyper-parameters may be varied under different setting.

Once the performance of the generative model exceeds the threshold, we use it to enumerate all the possible trajectories and store them to build the direct trajectory planning model, since the relationship between optimal control actions and parking destination is plain straight forward given generative data.

C. Combining Feedback Control to deal with rare situation

Besides the feed-forward controller provided by deep neural network, combining a feedback controller could be helpful in this general parking solution under rare situation. First, without evidence accumulation, cold start problem may occur to the data-driven models. Relationship between steering and trajectory for different types of vehicles under some situations, such as extreme weather, might not happen frequently and left sufficient data to train a usable model. Therefore, the generative model cannot draw a solid inference in these rare circumstances and an expedient solution will be needed. Another reason is due to the complexity of the traffic scenario, the parking situation might change after the trajectory has been planned. A feedback controller can deal with minor change and avoid planning the trajectory once more.

Such combination can be created in many ways. For example, take the difference between the desired state and the practical state in every moment as the negative feedback signal and the planned input steering sequence given by an imperfect model as the feed-forward signal, an adjusted steering angle sequence can be obtained to make the practical trajectory overlap the planned trajectory as much as possible. Adjustments and adjusted trajectories will back propagate to generator and controller neural networks for further training. Given steering sequence $\hat{u}$ send by neural network, the control signal $u = \{\delta_1, \delta_2, \ldots, \delta_n\}$ can be calculated by (2)

$$u = \hat{u} - \gamma \cdot K(x - \hat{x}) \qquad (2)$$

where K is the feedback matrix and γ the weight of feedback control signal.

Given control signal u, system state $x = [X, Y, \psi]$ is the real parking trajectory of vehicle corresponding to control actions while x' is the trajectory generated by generative model. The real actions-trajectories pairs $< u, x >$ will be feedback to controller for a more precise control, and prediction error $e = x' - x$ will be used to modify the generative model's weights for a better prediction.

By assigning proper feedback matrix K, we can guarantee the practical stability (convergence to a bounded error) of the tracking error [12]. Ensemble such a feedback control with deep neural network and decay the weight of it with the increasing of network's performance, a smooth control can be performed to deal with the rare situation. Since how to design such a feedback control does not fit with the main theme of this paper, we do not further discuss it here.

IV. Simulation Experiments

We set up a numerical simulation to exam our methods. By using a simplified model of vehicle dynamic, we simulate parking scenarios met by an autonomous vehicle and test the convergence speed of our model respect to the size of dataset. The stability and accuracy of such model are examined as well.

A. Experimental Setup

We adopted the well-known bicycle model [13] to generate parking trajectories for the following experiment. It should be noticed, though there exists non-linear dynamics that cannot be accurately modelled by bicycle model as pointed out in [14], such simplification will not compromise our model's ability since it does not rely on the theoretical model describing the parking trajectories. Provided enough data which roughly reflect the nature of vehicle motion, this novel approach can update and eventually have a satisfactory performance on guiding vehicle parking.

Parameters of bicycle model are selected from two common types of vehicles, where vehicle type A is sedan and

type B is SUV. Lateral/yaw dynamics of a typical model for each type are listed in Table II.

TABLE II
FEATURES OF TYPICAL VEHICLE MODELS

	Type A	Type B
l_f(mm)	1108	1003.2
l_r(mm)	1662	1638.8
l_{fo}(mm)	960	870
l_{ro}(mm)	1080	750
l_w(mm)	1820	1780
m(kg)	1441	1424
c_f(N/rad)	75000	75000
c_r(N/rad)	78000	75000
$I(kg.m^2)$	2200	2200

For presentation simplicity, we only consider the case that the vehicle keeps $v = 1m/s$ throughout the whole parking process, and the parking behavior of the vehicle must be accomplished in 12s, which is uniformly divided into 4 segments. So the time interval in each segment is 3s.

The steering angles are selected from the set $S_\delta = \{-0.6, -0.4, -0.2, 0, 0.2, 0.4, 0.6\}$. Setting $N = 4$ and $K = 7$, we can generate $7^4 = 2401$ trajectories for each type, as shown in Fig. 5

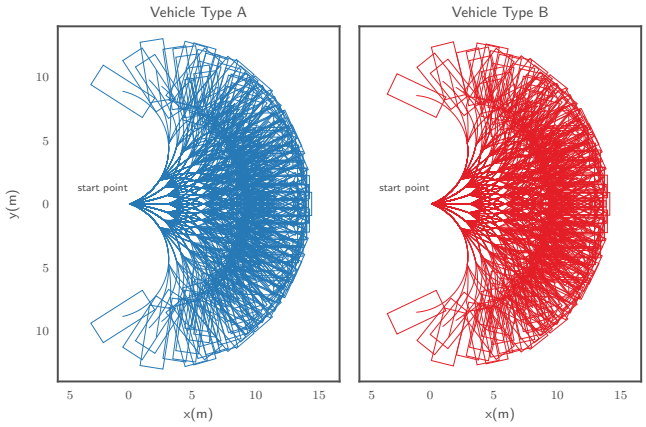

Fig. 5. Trajectories of different type vehicle. To make the plot clear, only about 1/10 of the generated 2401 trajectories are drawn;

Fig. 6 shows a sampled trajectory with steering sequence $\delta_f^4 = \{0.6, 0.6, 0.4, 0.6\}$. With same control actions, the difference of vehicle dynamics results in different trajectories.

The neural network we set up in this experiment takes a vector consists of $\{v_1, v_2, \ldots, v_t\}$ and $\{\delta_1, \delta_2, \ldots, \delta_t\}$ as input, where $t = 4$ is the length of time segments. One hidden layers consists of 120 Rectified Linear Unit (ReLU)[15] served as general hidden layer, followed by two layers with same structure as specific hidden layers. The output of network is corresponding trajectory, decomposed into $(\tilde{X}, \tilde{Y}) = \{(\tilde{x_1}, \tilde{y_1}), (\tilde{x_2}, \tilde{y_2}), \ldots, (\tilde{x_t}, \tilde{y_t})\}$ and $\tilde{\Psi} = \{\tilde{\psi_1}, \tilde{\psi_2}, \ldots, \tilde{\psi_t}\}$. Since most parking space in the city is $2.5m \times 5m$, the error should be less than $0.2m$ around the ending position of generative trajectories.

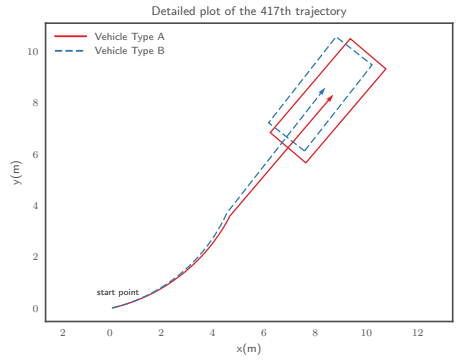

Fig. 6. Example of different types of vehicles' trajectories

We use stochastic gradient descent(SGD) to optimize the network. The learning rate α is set to 0.1, and decays $1e-6$ in every epoch. Gradients will be clipped to a maximum value of 1 and a minimum value of -1 to prevent gradient explosion. It should be pointed out that, using fine-tuned hyper-parameters with SGD or other methods such as adaptive moment estimation(Adam)[16] can achieve a better result. The setting in our paper is mainly for reproducibility and serves as a baseline.

To examine the performance of such method, we compare transferred deep neural network(**TDNN**) described in *Section III.B* with two other models. Both of them have the same structure, but the layers' functionalities are different. One is a pre-trained network(**PDNN**) with three specific layers and pre-trained on vehicle type A, another is a normal network(**DNN**) with the same structure as PDNN but without pre-trained.

Three networks will be trained on the control-trajectory pairs of vehicle type B. For each training epoch, the data size obtained by the network is constrained to 1, and this trajectory is sampled uniformly from all possible situations. This constraint simulates the actual driving environment, where drivers will perform specific parking actions only a few times a day. In such situation, the model can only sample very limited size of data from the target domain.

B. Results and Discussion

Models' performances are evaluated by the accuracy of trajectories they generate given all possible steering sequences. The mean and standard deviation values of prediction errors made by each network are shown in Fig. 7.

It can be found that, pre-trained on vehicle type A will improve models' performance on vehicle type B, letting both TDNN and PDNN have lower mean prediction errors compared than DNN. However, without general hidden layers, PDNN is more sensitive to the randomness brought by actions-trajectories pair sampling compared with TDNN. Though PDNN has met the same amount of data as TDNN, its final performance is still inferior to TDNN.

Such results point out the difference between data augmentation and transfer learning, that without a constrained on layers' functionality, the underlying knowledge may not

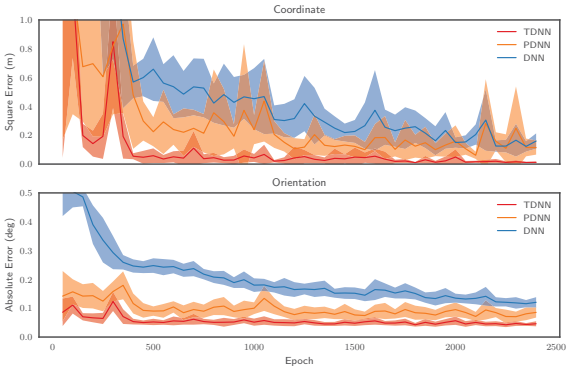

Fig. 7. Mean and Standard deviation of Prediction Error

be learned. The general hidden layers, transferring general parking knowledge and served as a generalization, make TDNN attain an acceptable level of performance with a tiny fraction of target domain data, and perform more accurate and stable. A detailed plot of a sampled trajectory generated by different models is shown in Fig. 8. It indicates that, combining with parallel learning, deep neural network can generate higher accurate trajectories than other methods.

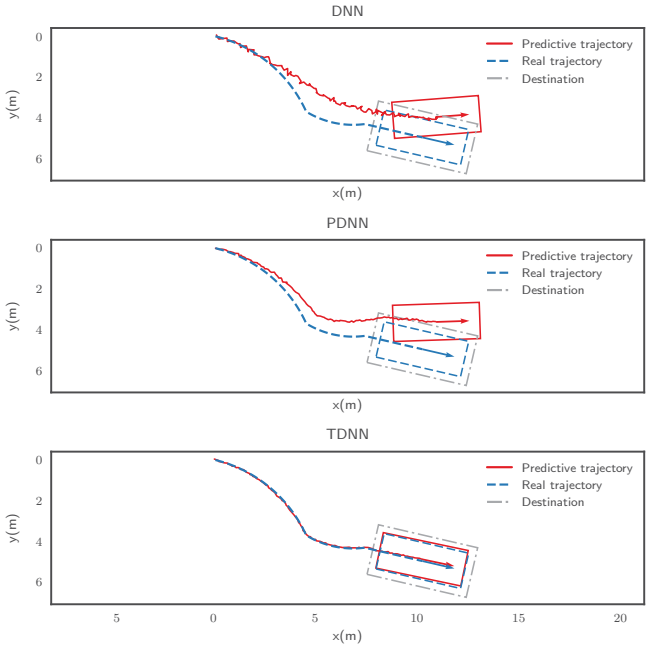

Fig. 8. Example of Trajectories generated by different models

We can use this deep neural network to enumerate all the possible parking trajectories that a vehicle can make in high resolution and learn to set up a direct mapping from any initial/final state pair to the corresponding steering action.

V. Conclusions

In this paper, we address the parking control problem by combining direct trajectory planning method with parallel learning. Based on the study of the reference vehicle and its trajectories corresponding to control actions, a deep neural network learns the general relationships between operations and movements. With acquaintance of this knowledge, parking trajectories of all possible steering actions in various vehicles can be generated, then refined as a general parking skill.

Tests prove the high reliability and efficiency of proposed method. In addition, unlike classic models such as the bicycle model, the generative model used in this method can be updated in its entire life, or forming a collective intelligence by sharing weights with similar models, which possesses a good application prospect in the Internet of Vehicle field.

To the best of our knowledge, this is the first attempt to apply parallel learning and transfer learning trick in intelligent vehicle motion control field. It brings new hope to build more intelligent automated vehicles so as to implement more difficult tasks.

References

[1] L. Li and F.-Y. Wang, *Advanced motion control and sensing for intelligent vehicles*. Springer Science & Business Media, 2007.

[2] H. Vorobieva, S. Glaser, N. Minoiu-Enache, and S. Mammar, "Automatic parallel parking in tiny spots: Path planning and control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 1, pp. 396–410, Feb. 2015.

[3] X. Du and K. K. Tan, "Autonomous reverse parking system based on robust path generation and improved sliding mode control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 16, no. 3, pp. 1225–1237, June 2015.

[4] F. Gómez-Bravo, F. Cuesta, A. Ollero, and A. Viguria, "Continuous curvature path generation based on β-spline curves for parking manoeuvres," *Robotics and autonomous systems*, vol. 56, no. 4, pp. 360–372, 2008.

[5] L. Li and F. Y. Wang, "Parking guidance system for front wheel steering vehicles using trajectory generation," in *Proc. IEEE Int. Conf. Intelligent Transportation Systems*, vol. 2, Oct. 2003, pp. 1770–1775 vol.2.

[6] T. M. Mitchell, "Machine learning. 1997," *Burr Ridge, IL: McGraw Hill*, vol. 45, no. 37, pp. 870–877, 1997.

[7] L. Li, Y.-L. LIN, D.-P. CAO, L.-N. ZHENG, and F.-Y. WANG, "Parallel learning a new framework for machine learning," vol. 43, no. 1, pp. 1–8, June 2017.

[8] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, no. 10, pp. 1345–1359, Oct. 2010.

[9] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural networks*, vol. 2, no. 5, pp. 359–366, 1989.

[10] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in neural information processing systems*, 2014, pp. 3320–3328.

[11] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[12] J. C. Doyle, K. Glover, P. P. Khargonekar, and B. A. Francis, "State-space solutions to standard H_2 and H_∞ control problems," *IEEE Transactions on Automatic control*, vol. 34, no. 8, pp. 831–847, 1989.

[13] G. Rill, *Road Vehicle Dynamics: Fundamentals and Modeling*. CRC Press, 2011.

[14] L. Li, F.-Y. Wang, and Q. Zhou, "Integrated longitudinal and lateral tire/road friction modeling and monitoring for vehicle motion control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 7, no. 1, pp. 1–19, Mar. 2006.

[15] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, 2010, pp. 807–814.

[16] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014. [Online]. Available: http://arxiv.org/abs/1412.6980