2018 21st International Conference on Intelligent Transportation Systems (ITSC)
Maui, Hawaii, USA, November 4-7, 2018

Zero-shot Deep Reinforcement Learning Driving Policy Transfer for Autonomous Vehicles based on Robust Control

Zhuo Xu*, Chen Tang* and Masayoshi Tomizuka

Abstract— **Although deep reinforcement learning (deep RL) methods have lots of strengths that are favorable if applied to autonomous driving, real deep RL applications in autonomous driving have been slowed down by the modeling gap between the source (training) domain and the target (deployment) domain. Unlike current policy transfer approaches, which generally limit to the usage of uninterpretable neural network representations as the transferred features, we propose to transfer concrete kinematic quantities in autonomous driving. The proposed robust-control-based (RC) generic transfer architecture, which we call RL-RC, incorporates a transferable hierarchical RL trajectory planner and a robust tracking controller based on disturbance observer (DOB). The deep RL policies trained with known nominal dynamics model are transfered directly to the target domain, DOB-based robust tracking control is applied to tackle the modeling gap including the vehicle dynamics errors and the external disturbances such as side forces. We provide simulations validating the capability of the proposed method to achieve zero-shot transfer across multiple driving scenarios such as lane keeping, lane changing and obstacle avoidance.**

I. Introduction

Learning intelligent and reliable driving policies has been an ongoing challenge for both deep learning and control. Although conventional planning-control [1] and imitation-oriented learning [2] [3] approaches are shown to have capability of controlling autonomous vehicles, deep reinforcement-learning-based (deep RL) methods are promising to tackle more complicated and interacting scenarios that conventional methods are incapable to solve, as well as rare cases outside the demonstration dataset for the supervised imitation learning.

While the vast exploration and flexible hierarchical configurations enable deep RL methods to obtain versatile policies more easily [4] [5], robustness has essentially been the main drawback that prevents the application of deep RL in autonomous driving. Concretely, pre-optimized deep RL policies are over specialized, and thus often fail when the target vehicle has dynamics variation from the training setting, or when the target vehicle is affected by force disturbances due to strong wind or body incline. These differences between the source and target settings together are called the modeling gap. Given the modeling gap being an inevitable barrier for the deployment of deep RL in autonomous driving, in this contribution we aim to bridge the modeling gap by achieving fast and safe transfer of deep RL autonomous driving policies.

Prior efforts in deep RL attempt to achieve such objective using transfer learning and meta learning. Various contributions in these areas have indeed brought source policies to work in the target settings, but their applications in autonomous driving are limited due to safety concerns. Overall, such deep RL transfer methods embed transferable representations into uninterpretable neural networks and hope for the best in the target domain, and thus are not transparent and reliable. We seek to solve the transfer problem using an alternative tool: robust control (RC), and propose a generic RL-RC transfer framework. In this framework, the deep RL policy is applied to an imaginary setting in the source domain to generate a reference trajectory for the target vehicle. A robust controller is applied to track the reference trajectory tolerating the modeling gap. This framework is generic in that it can be generalized to any deep RL control policy transfer tasks, and that various kinds of robust controllers can be used for tracking. Compared to other transfer learning methods, the proposed framework has two fundamental advantages: (i) the transfer of the interpretable kinematic features makes the transfer framework transparent and reliable; (ii) stability and response in time and frequency domains, can be well defined and analyzed.

The contribution in this paper is four-fold. First, we purpose a generic approach for deep RL driving policy transfer. Second, we implement a hierarchical RL model which serves as the transferable trajectory planner. Third, we develop a disturbance-observer-based (DOB) robust tracking controller so as to actively reject the disturbances induced by the modeling gap. Finally, we report simulation results validating that the RL-RC architecture can zero-shoot transfer the policy under certain level of parameter variation and external disturbances.

The rest of this paper is organized as follows: The related work is summarized in Section II. In Section III, the generic RL-RC architecture for policy transfer is described. Then, the setting of our autonomous driving tasks, together with the implementation of all the components in the RL-RC architecture, is illustrated. In Section IV, simulation results are presented and evaluated to show the effectiveness of the proposed method, followed by the conclusion in Section V.

II. Related Work

Deep Learning for Autonomous Driving. Prior works on deep-learning-based autonomous driving policy have mostly been developed based on behavior cloning or imitation learning. From the pioneering ALVINN [6] to recent works such as [2], [3] and [7], researchers trained neural networks

[†]This work was supported in part by Berkeley DeepDrive (BDD).

[†]Z. Xu, C. Tang and M. Tomizuka are with the Department of Mechanical Engineering, University of California, Berkeley, CA 94720 USA (e-mail: zhuoxu@berkeley.edu, chen_tang@berkeley.edu, tomizuka@berkeley.edu).

*These authors contributed equally to this work.

to predict driving actions. However, the models were trained in a supervised manner, and are inevitably limited by the dataset. Deep RL-based driving policy learning methods have demonstrated their capability in simulators like TORCS. Various deep RL methods including deep Q-learning and actor-critic algorithms have been applied to train lane-keeping policies with low-dimensional feature vector or image pixels as inputs [8]–[10]. Meanwhile, designing transfer methods to bridge the modeling gao for deep RL application in autonomous driving is a open but intriguing problem.

Deep RL Policy Transfer. One category of transfer learning uses source domain randomization to embed robustness. The EPOpt algorithm [11] randomly samples dynamic parameters from a prior distribution and optimizes the policy for cases with worst-performing dynamic parameters. [12] uses RNN to take in historical paths, expecting the policy to be adaptive by implicitly identifying the parameters. Similar idea has been adopted in [13], but an on-line system identification module is explicitly introduced to identify the parameters. The estimated parameters are fed into a parameter-universal policy. All these works are limited in that they only handle the discrepancy of dynamic parameters, while robustness against external disturbances is excluded from consideration.

Another line of work is through model adaptation. [14] trains a neural network for dynamics model and adapts its local linear model on-line for model-based control methods. In [15], a deep inverse dynamics model of the target system is trained, and transfer is achieved by executing inverse model outputs to reach the nominal state generated by performing source policy in source environment. One problem of such approaches is that, unlike tracking by feedback control, a feasible and accurate inverse model for feedforward tracking is hardly guaranteed in the target domain. Also, a neural network approximating system dynamics is non-trivial to find in practice. Furthermore, fine-tuning in the target domain is not desirable due to safety.

A method developed on a similar idea to the one in this paper is in [16], in which a MPC controller is designed to stabilize the target system around the nominal trajectory generated by consecutively applying the policy in the source system. Theorems on tube-based MPC ensure that the states are bounded under certain modeling error. However, the bound cannot be explicitly found and no asymptotic stability can be guaranteed. Moreover, solving on-line optimization problem is computationally expensive, while robust controllers are usually easier and faster for the trajectory tracking problem of automated vehicles.

DOB for Vehicle Lateral Control. Vehicle lateral control for trajectory tracking has been well-established with many existing methods [17]. DOB is a robust control technique to reject disturbances with guaranteed robust stability for linear system [18]. Moreover, given a stabilizing nominal controller, Q-filter in DOB can be an arbitrary stable filter to make the sensitivity function shaped as desired [19]. It has been applied to robust lateral trajectory tracking and its effectiveness has been verified by experiments [20].

III. Methodology

A. Architecture

The proposed RL-RC policy transfer architecture consists of a deep RL-based high-level planning module and a RC-based low-level tracking controller. The overall methodology consists of the off-line deep RL policy training and the on-line policy transfer. In the off-line source domain, we pretrain a deep RL policy mapping the perception input to control commands, which can drive the source vehicle to produce a trajectory completing the control task.

For the on-line transfer in the target domain, Fig. 1 shows how the system works to transfer the source driving policies. In general, the RL-RC system performs closed-loop tracking of a finite-horizon previewed trajectory generated by the pretrained policy. Concretely, at each time step, the target agent obtains the perception observation. Then according to this observation, the system constructs an imaginary source agent in the same setting as the target agent. In the imaginary source domain, the pretrained policy is used to control the imaginary source agent to perform the driving task for a finite horizon, resulting in a trajectory of kinematic states, which serves as the reference for the target vehicle. Given the reference trajectory, the target agent uses a closed-loop robust tracking controller to produce the actual control command for the target vehicle. As the target environment makes one step forward, the observation for the new timestep is collected, and the system repeats the same procedure. In this architecture, the kinematic features are transfered without change, and the modeling gap is compensated by RC.

To generate the reference trajectory in the imaginary source domain, one can use the cascaded high-level planner, which consists of the pretrained policy and the source environment, or train a deep RL policy network that directly maps perception input to trajectory of kinematic states. We choose the latter approach so as to optimize the policy for the whole procedure, enabling both dynamical feasibility and optimality of the planned trajectory.

There are two key underlying assumptions for the RL-RC system: (i) the trajectory planned by the imaginary source agent is comparably satisfying for the target task; (ii) it is feasible for the target vehicle to track the trajectories produced by the source vehicle. These assumptions are reasonable as the source and target vehicles and settings are similar, and the RL-RC is only effective for such cases.

Our proposed RL-RC framework is generic in many aspects:

1) This system can perform policy transfer for various kinds of tasks in autonomous driving. It can also be applied in other robotics and control scenarios if the assumptions are satisfied and a robust controller can be designed. In this paper, we evaluate the RL-RC approach for three typical driving tasks: lane keeping (LC), lane changing (LC), and obstacle avoidance (OA).
2) The proposed method places no restrictions on the structures of the deep RL policy. In this paper we

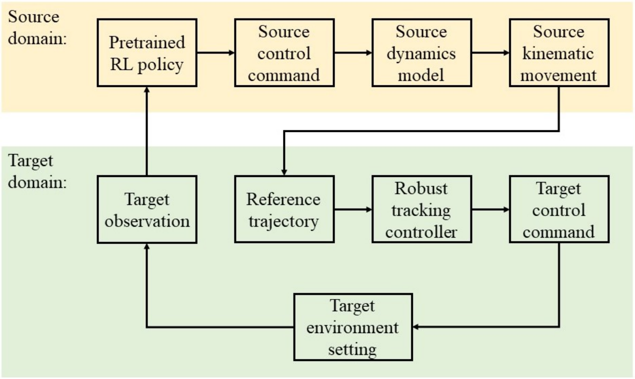

Fig. 1. Target domain on-line implementation of the RL-RC architecture

use a hierarchical RL model to generate reference trajectories in the imaginary source domain.

3) Any kind of RC method is compatible with the RL-RC architecture. Among various RC algorithms, we use a disturbance-observer-based (DOB) tracking controller to actively reject the disturbance induced by modeling error.

B. Environment

In both the source domain and the target domain, we simulate the vehicle dynamics using nonlinear bicycle model discretized using forward Euler method with $\Delta t = 0.02s$. The model is illustrated in Fig. 2. The longitudinal acceleration a_x and the derivative of steering angle $\dot{\delta}$ are inputs. The seven state variables are longitudinal speed v_x, lateral speed v_y, yaw rate ω_z, global coordinates X, Y, yaw angle ψ and steering angle δ. To obtain simulated vehicle with high-fidelity, we include tire force saturation and restrict magnitudes of $a_x, \delta, \dot{\delta}$ to reasonable ranges. Moreover, we omit longitudinal powertrain dynamics, assuming nearly perfect longitudinal tracking by a low-level longitudinal controller.

Apart from the vehicle dynamics, we also define terms related to the driving scenarios. We define lateral displacements, Δy and Δy_s, and yaw errors, $\Delta\psi$ and $\Delta\psi_s$, to represent the tracking errors relative to lane or reference trajectory (Fig. 2). Δy and $\Delta\psi$ are defined with respect to the center of gravity (CG), and Δy_s and $\Delta\psi_s$ are defined at a point S specified by a look-ahead distance d_s. We denote the angle of the vehicle speed v as $\psi_v = \psi + \beta$, where β is the angle between vehicle speed and yaw. Similarly, the angle of v_s is $\psi_{v_s} = \psi + \beta_s$. The yaw errors are the angle between velocity and the tangent direction of the reference curve, specifically, $\Delta\psi = \psi_v - \psi_{ref}$ and $\Delta\psi_s = \psi_{v_s} - \psi_{s,ref}$. With the yaw error $\Delta\psi$, one can easily derive ego vehicle's speed along and perpendicular to the reference curve: $v_{\parallel} = v\cos(\Delta\psi)$ and $v_{\perp} = v\sin(\Delta\psi)$.

For the collision avoidance task, we consider a simple scenario with a single surrounding vehicle, which perfectly tracks the lane with constant speed v_{srd}. We denote the relative longitudinal and lateral position of the surrounding vehicle by Δx_{srd} and Δy_{srd}.

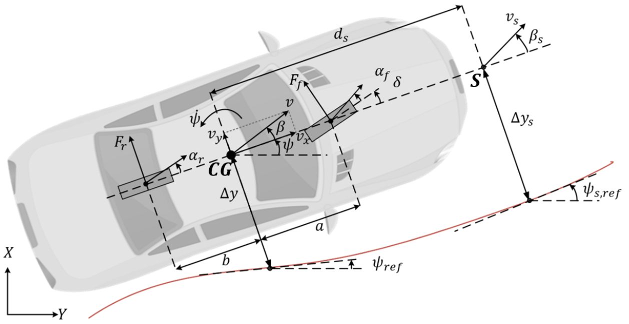

Fig. 2. Illustration of the terminology for the simulated environment and the linear tracking model for the controller

C. Hierarchical RL

So as to learn driving policy using deep RL, we first define the Markov Decision Process (MDP) for the driving tasks. Since different tasks have different observations, we define clusters of observations as entities. Concretely, we define the observation of ego vehicle to be $o_{vh} = [v_x \; v_y \; \omega_z \; \delta]^T$. For each lane, denoted l_i, $i \in N$, we define tracking observations as $o_{ref,i} = [\Delta y_i \; \Delta\psi_i \; \Delta y_{s,i} \; \Delta\psi_{s,i}]^T$. The lane that the ego vehicle is tracking is denoted using l_{i^*}. The obstacle observation is $o_{srd} = [v_{srd} \; \Delta x_{srd} \; \Delta y_{srd}]^T$. For simplification, we assume perfect and deterministic observations. For all the tasks, control command $a_{vh} = [a_x \; \dot{\delta}]^T$ is defined as the actions in the MDP. The actions are continuous and normalized by their maximum allowable values.

For performance evaluation, we define the step tracking reward function at timestep t:

$$r_{tra}(t) = v_{\parallel,i^*}(t) - |v_{\perp,i^*}(t)| - \eta \cdot \Delta y_{i^*}(t)^2, \qquad (1)$$

where $\eta > 0$ is an importance factor. The step rewards are discounted with factor γ. An episode is defined to have maximum of 1000 timesteps, The episode will be terminated by large deviation from the lane or violating collision constraint. Meanwhile, the agent will be punished with a large negative reward, denoted r_{dev} and r_{col}. The interfaces of lane keeping (LK), lane changing (LC) and obstacle avoidance (OA) are shown in Table. I.

In our implementation, a hierarchical RL model that can modularize distinct driving attributes is applied. The attributes refer to driving behaviors such as obstacle detection, lane selection, and lane tracking. Table II and Fig. 3 give the detailed module interfaces and their usages. Theoretically, all the three basic modules can be optimized using any kind of method including deep RL. In the implementation, both lane-selection and obstacle-detection modules are rule-based and reasonably optimized for the sake of simplicity. The lane-tracking module is optimized using model-free deep RL. The benefit of such hierarchical implementation over end-to-end training is that basic attribute modules are much easier to optimize compared to end-to-end training of complicated high-level driving policies.

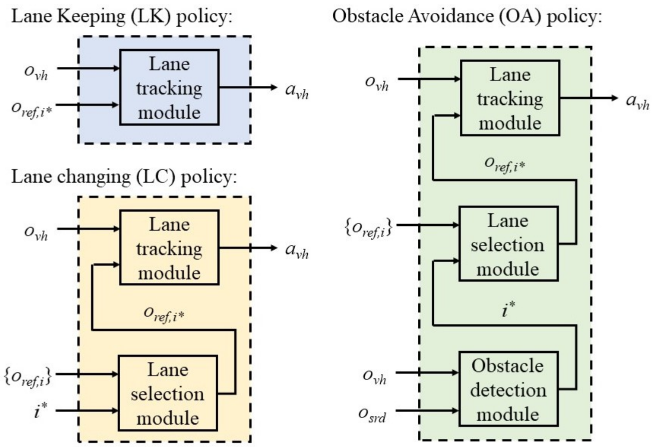

Fig. 3. Usage of hierarchical RL modules to assemble policies for the lane keeping (LK), lane changing (LC) and obstacle avoidance (OA) tasks.

D. DOB-based Tracking Controller

We design the reference trajectory as a trajectory of (X, Y, ψ_v, v_x). While the speed profile is followed directly by executing corresponding a_x, the path defined by the series of (X, Y, ψ_v) is tracked by the tracking controller. To design the controller, first we need to obtain an approximate linear model for the tracking problem. We adopt the constant speed linear bicycle model for lateral dynamics [21]:

$$\dot{x} = A \cdot x + B \cdot \delta, \tag{2}$$

where the state variable is $x = [v_y \ \psi \ \dot{\psi}]^T$. Under small angle assumption, $\beta \approx v_y/v_x$, we approximate β_s as:

$$\beta_s \approx \frac{v_x \beta + d_s \dot{\psi}}{v_x} = \frac{1}{v_x} v_y + \frac{d_s}{v_x} \dot{\psi}. \tag{3}$$

Thus:

$$\psi_{v_s} = \psi + \beta_s = \begin{bmatrix} \frac{1}{v_x} & 1 & \frac{d_s}{v_x} \end{bmatrix} \begin{bmatrix} v_y \\ \psi \\ \dot{\psi} \end{bmatrix} = C \cdot x. \tag{4}$$

Following [20], the derivative of $\psi_{s,ref}$ is approximated as $\dot{\psi}_{s,ref} \approx v_x \kappa_{s,ref}$, where $\kappa_{s,ref}$ refers to the curvature of the curve at the reference point. Consequently:

$$\Delta \dot{\psi}_s = \dot{\psi}_{v_s} - \dot{\psi}_{s,ref} \approx \dot{\psi}_{v_s} - v_x \kappa_{s,ref}. \tag{5}$$

Finally, $\Delta \dot{y}_s$ can be approximated as $\Delta \dot{y}_s \approx v_x \Delta \psi_s$.

TABLE I
DEFINITION OF DRIVING TASKS

Task	Observation	Evaluation
LK	o_{vh}, o_{ref,i^*}	$\Sigma \gamma^t r_{tra}, r_{dev}$
LC	$o_{vh}, \{o_{ref,i}\}, i^*$	$\Sigma \gamma^t r_{tra}, r_{dev}$
OA	$o_{vh}, \{o_{ref,i}\}, o_{srd}$	$\Sigma \gamma^t r_{tra}, r_{dev}, r_{col}$

TABLE II
DEFINITION OF RL MODULES

deep RL module	Input	Output
Lane tracking	o_{vh}, o_{ref,i^*}	a_{vh}
Lane selection	$\{o_{ref,i}\}$, i^*	o_{ref,i^*}
Obstacle detection	o_{vh}, o_{srd}	i^*

Using the equations above and applying forward Euler discretization, the overall tracking model can be obtained as the block diagram shown in Fig. 4, where T_s is the sampling time, $\frac{T_s z^{-1}}{1-z^{-1}}$ is the transfer function of the discretized integrator. $G_v(z^{-1})$ is the vehicle dynamics that maps δ to ψ_{v_s}, of which we can derive the nominal transfer function $G_{nv}(z^{-1})$ using (2) and (4). As shown in the diagram, $\psi_{s,ref}$ can be considered as disturbance to the system.

The robust controller is designed based on the nominal model based on the source vehicle. First, we design a proportional feedback controller as: $u_c = -k_1 \Delta \psi_s - k_2 \Delta y_s$. For analysis, we can further write the control law as:

$$u_c = -k_2 \left(\frac{k_1}{k_2} + \frac{v_x T_s z^{-1}}{1 - z^{-1}} \right) \Delta \psi_s = -k_2 C_1(z^{-1}) \Delta \psi_s \tag{6}$$

where $C_1(z^{-1}) = \frac{k_1}{k_2} + \frac{v_x T_s z^{-1}}{1-z^{-1}}$. In the implementation, we hold constant $\frac{k_1}{k_2}$, and tune k_2 to achieve stability of the closed-loop system using root locus. We inspect stability for various v_x such that the resulting controller can stabilize the vehicle for the range of velocity in the given driving tasks.

Then a DOB is added to the nominal feedback controller. The block diagram for the overall closed-loop system is shown in Fig. 5. DOB is inserted between $C_1(z^{-1})$ and k_2 so that disturbance due to modeling error between Δy_s and $\Delta \psi_s$ can be rejected. $P_n(z^{-1})$ and $\hat{P}_n(z^{-1})$ are the nominal plant and nominal plant without delay, defined as:

$$P_n(z^{-1}) = z^{-2} \hat{P}_n(z^{-1}) = G_{nv}(z^{-1}) C_1(z^{-1}) \tag{7}$$

In our case, the nominal plant $P_n(z^{-1})$ has two timesteps' delay. We make $\hat{P}_n(z^{-1})$ free of delay so that $\hat{P}_n^{-1}(z^{-1})$ is realizable. The closed-loop sensitivity function with DOB is:

$$S = \frac{1}{1 + k_2 G_v C_1 + z^{-2} (\frac{G_v}{G_{nv}} - 1) Q} (1 - z^{-2} Q). \tag{8}$$

We design Q as a second-order low-pass filter to reject low-frequency disturbances, because $\kappa_{s,ref}$ should have relatively low frequency components for smooth reference trajectory.

When modeling errors exist, we can use the robust control theorem to ensure robust stability [19]. However, modeling uncertainty between the linear model and the nonlinear model nonlinear model is involved in our problem, which complicates the analysis. Future efforts will be made to approximate the uncertainty bound for robustness specification.

One last note is that P is time-varying as v_x is not constant in general. In our implementation, we designed the DOB based on the final steady speed of the vehicle in simulations for simplification. In practice, an adaptive DOB can be designed.

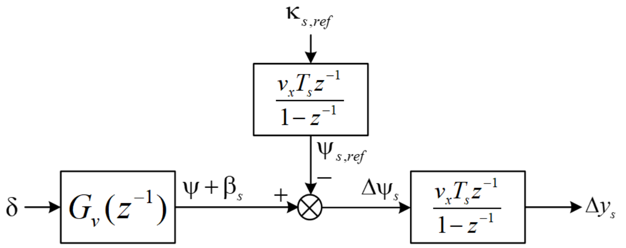

Fig. 4. Block diagram of linear tracking model

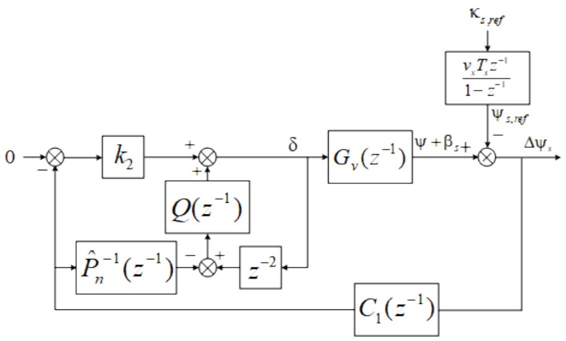

Fig. 5. Block diagram of closed-loop system

IV. Simulation Results

A. Training of Hierarchical RL Model

In the off-line training phase, the nominal vehicle dynamics is applied, and the environment includes parallel sinusoidal lanes with lane width of 3 meters. Both the policy network and the value network are three layer fully-connected neural networks. We used the proximal policy optimization (PPO) [22] algorithm as the model-free deep RL algorithm. Training techniques such as advantage normalization and reparameterization were applied.

In the training phase, the agent achieves successful driving for 1000 timesteps in around 7000 iterations, and the policy optimization converges in around 12000 iterations. After the RL modules are optimized, they can be flexibly assembled to complete different driving tasks, using structures shown in Fig. 3. Each deep RL policy is then cascaded with the source environment to get the transferable trajectory planner. The performances of the hierarchical RL policies for all the three tasks are satisfying, which gives us confidence of the transfered deep RL-based high-level trajectory planner. The results will be presented later in Subsection IV-C, for direct comparison with the transfer system.

B. Analysis of DOB-based Tracking Controller

Given the parameters of the nominal vehicle model, we specified the look-ahead distance as $d_s = 15\mathrm{m}$ and designed the controller. We analyzed the closed-loop system at $v_x = 20\mathrm{m/s}$, which is approximately the final steady speed in our simulations. Fig. 6 shows the step responses of Δy_s and $\Delta\psi_s$ given a step input of previewed reference curvature $\kappa_{s,ref}$ with a magnitude of $10^{-3}m^{-1}$. DOB enables smaller tracking errors for both Δy_s and $\Delta\psi_s$. Particularly, steady state error of Δy_s is eliminated by adding DOB.

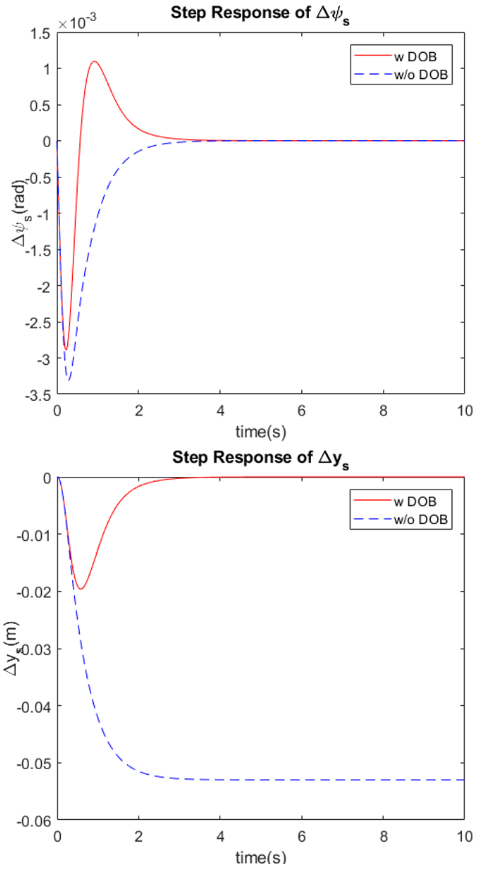

Fig. 6. Step responses of Δy_s and $\Delta\psi_s$ given a step input of previewed reference curvature $\kappa_{s,ref}$ with a magnitude of $10^{-3}m^{-1}$

C. Performance of Overall System

In this subsection, we evaluated the performance of the proposed RL-RC transfer architecture, and compared it with the baseline RL policies. We tested the models in all the three tasks (i.e., LK, LC, and OA). For source domain tests, we ran the baseline policies and their corresponding RL-RC systems on the nominal vehicle settings with random initial conditions for 10 times. For target domain tests, both driving strategies were deployed and tested from random initial conditions in 10 target settings with modeling gap.

We considered two types of modeling gap: (i) Variation in model parameters. We randomly generated test environments by adding zero-mean and uniformly distributed errors to the parameters of the vehicle model, including vehicle geometry, mass, rotational inertia, tire model parameters, and friction factors. We evaluated the performance on 10 different target vehicles. (ii) External side-force caused by the side slope of the road or wind. We added a constant external force along Y axis in the target environments. We conducted 10 tests for each given magnitude of the external force. We recorded the total discounted rewards and episodic lengths for each test episode, and calculated their means and standard deviations for each testing category.

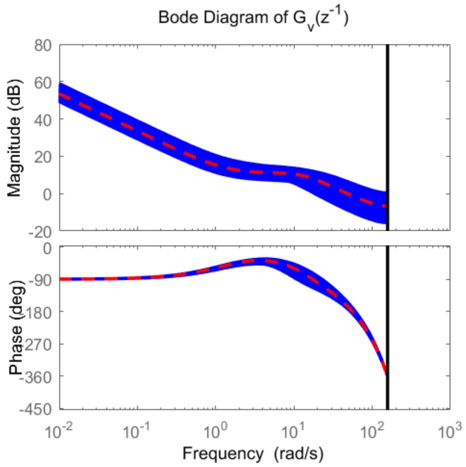

Fig. 7. Bode plots of $G_v(z^{-1})$ with parameter variation bounded by 20%. The blue curves correspond to 100 samples of $G_v(z^{-1})$ with uniformly distributed parameter variation bounded by 20%. The red dash curve corresponds to the nominal model.

Table. III gives the overall performance comparison of the baseline RL policy and RL-RC method. Fig. 9 shows two examples illustrating the driving behaviors of the two methods. The target settings have parameter variation bounded by 20% of the nominal values or side force of a magnitude of 5000N. A demonstration video under the same target settings is also provided online [1]. To give an intuition of the effects of such modeling gap, Fig. 7 shows the bode plots of 100 samples of $G_v(z^{-1})$ with uniformly distributed parameter variation bounded by 20%. The 5000N side force contributes a maximum lateral acceleration of $2.78\mathrm{m/s}^2$ to the nominal vehicle.

Table III and Fig. 9 show the effectiveness of the RL-RC in three aspects: (i) The robustly satisfying performance of the deep RL policies in the source setting is validated, which gives us confidence of the transfered trajectory planner to generate reasonable reference for the tracking controller. (ii) The poor performance of the baseline RL policy in the target settings indicates the incapability of the direct policy transfer. When the baseline policy is applied to a different vehicle or when the target vehicle is affected by a side force, the baseline policy often fails and the target vehicle eventually loses control. (iii) The consistent performance of the RL-RC in both the source and the target domain validates the proposed approach can zero-shoot the transfer tasks. This is thanks to the robust and satisfying tracking performance of the DOB-based tracking controller.

Fig. 8 further compares the performance of the baseline RL policy and RL-RC system under modeling gap of different magnitudes. The curves show the mean value of the reward, and the error-bar indicates the interval with a maximum deviation of one standard deviation from the mean value. The proposed RL-RC framework obtains consistent reward with parameter variation up to 20% or side force of magnitude up to 5000N, whereas the performance of baseline RL policy keeps getting worsen as the modeling gap increases. Decrease in reward can be observed for all tested magnitude of disturbances.

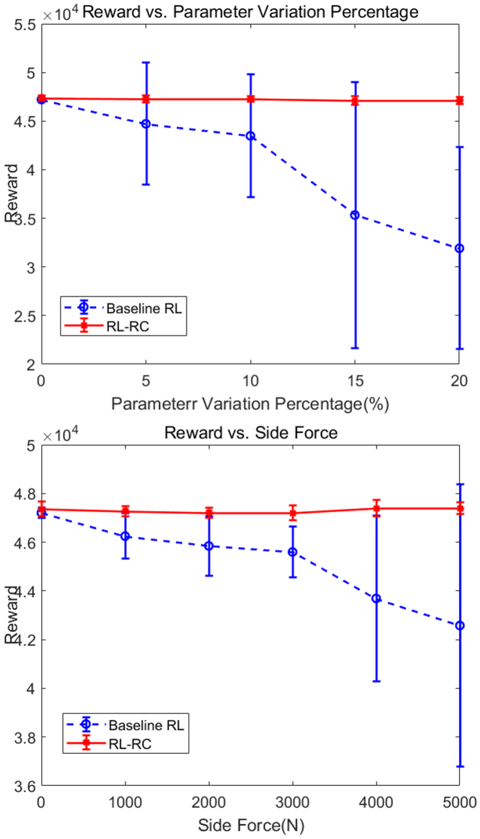

Fig. 8. Performance (episodic return) of baseline RL policy and RL-RC architecture under increasing modeling gap in Lane Changing (LC) task

V. Conclusion

We proposed a generic framework for deep RL driving policy transfer using RC techniques. The deep RL policy trained in the source domain is used to generate a finite-horizon previewed reference trajectory of interpretable kinematic features. In the target domain, a robust controller is designed to track the reference trajectory so that disturbances induced by the modeling gap can be rejected. We implemented the framework for three driving tasks in term of a hierarchical RL policy and a DOB-based robust tracking controller. Simulation results showed that the proposed RL-RC architecture can achieve zero-shot consistent performance in the target domain with certain level of parameter variation or external side force. Meanwhile, performance of the baseline RL policy is devastated by the modeling gap. Future efforts will be made to investigate the uncertainty bound introduced by model linearization so that robust stability of the DOB-based controller can be guaranteed. Moreover, the proposed method will be implemented to achieve policy transfer from simulated environment to a real vehicle. Also the current method only considers the variation of the system dynamics.

[1] https://berkeley.box.com/v/ITSC2018

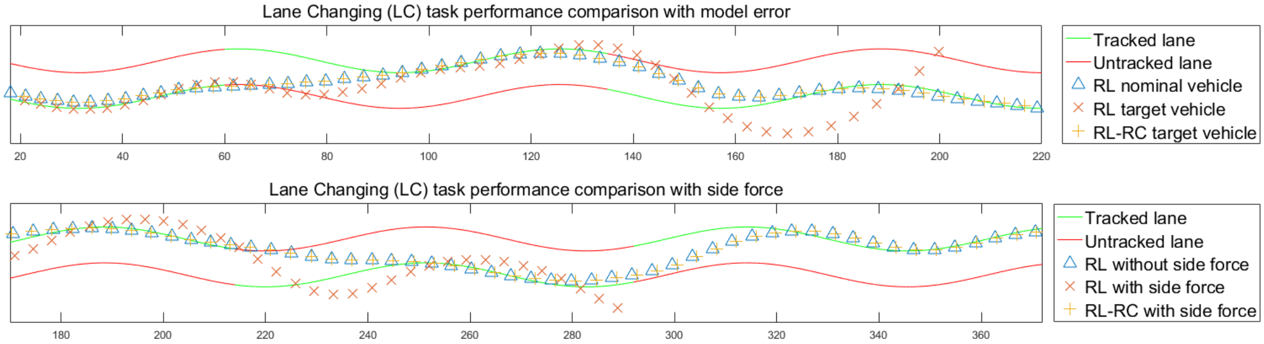

Fig. 9. Comparison of the driving behaviors for the RL and the RL-RC with modeling gap in the Lane Changing (LC) task

Furthermore, other kinds of modeling gap can affect the policy transfer performances (e.g. number of obstacles and behavior of surrounding vehicles). We will extend the current method to solve more general policy transfer problem for autonomous driving tasks.

References

[1] J. Levinson, J. Askeland, J. Becker, J. Dolson, D. Held, S. Kammel, J. Z. Kolter, D. Langer, O. Pink, V. Pratt, *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 163–168, IEEE, 2011.

[2] M. Bojarski, D. Del Testa, D. Dworakowski, B. Firner, B. Flepp, P. Goyal, L. D. Jackel, M. Monfort, U. Muller, J. Zhang, *et al.*, "End to end learning for self-driving cars," *arXiv preprint arXiv:1604.07316*, 2016.

[3] H. Xu, Y. Gao, F. Yu, and T. Darrell, "End-to-end learning of driving models from large-scale video datasets," *arXiv preprint*, 2016.

[4] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[5] Z. Xu, H. Chang, and M. Tomizuka, "Cascade attribute learning network," *arXiv preprint arXiv:1711.09142*, 2017.

[6] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in neural information processing systems*, pp. 305–313, 1989.

[7] U. Muller, J. Ben, E. Cosatto, B. Flepp, and Y. L. Cun, "Off-road obstacle avoidance through end-to-end learning," in *Advances in neural information processing systems*, pp. 739–746, 2006.

[8] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," *arXiv preprint arXiv:1509.02971*, 2015.

[9] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*, pp. 1928–1937, 2016.

[10] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[11] A. Rajeswaran, S. Ghotra, B. Ravindran, and S. Levine, "Epopt: Learning robust neural network policies using model ensembles," *arXiv preprint arXiv:1610.01283*, 2016.

[12] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," *arXiv preprint arXiv:1710.06537*, 2017.

[13] W. Yu, C. K. Liu, and G. Turk, "Preparing for the unknown: Learning a universal policy with online system identification," *arXiv preprint arXiv:1702.02453*, 2017.

[14] J. Fu, S. Levine, and P. Abbeel, "One-shot learning of manipulation skills with online dynamics adaptation and neural network priors," in *Intelligent Robots and Systems (IROS), 2016 IEEE/RSJ International Conference on*, pp. 4019–4026, IEEE, 2016.

[15] P. Christiano, Z. Shah, I. Mordatch, J. Schneider, T. Blackwell, J. Tobin, P. Abbeel, and W. Zaremba, "Transfer from simulation to real world through learning deep inverse dynamics model," *arXiv preprint arXiv:1610.03518*, 2016.

[16] J. Harrison, A. Garg, B. Ivanovic, Y. Zhu, S. Savarese, L. Fei-Fei, and M. Pavone, "Adapt: zero-shot adaptive policy transfer for stochastic dynamical systems," *arXiv preprint arXiv:1707.04674*, 2017.

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

[18] W.-H. Chen, J. Yang, L. Guo, and S. Li, "Disturbance-observer-based control and related methodsan overview," *IEEE Transactions on Industrial Electronics*, vol. 63, no. 2, pp. 1083–1095, 2016.

[19] X. Chen and M. Tomizuka, "Control methodologies for precision positioning systems," in *American Control Conference (ACC), 2013*, pp. 3704–3711, IEEE, 2013.

[20] C. Rathgeber, F. Winkler, D. Odenthal, and S. Muller, "Lateral trajectory tracking control for autonomous vehicles," in *Control Conference (ECC), 2014 European*, pp. 1024–1029, IEEE, 2014.

[21] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[22] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

TABLE III

Comparison of the performances of baseline RL policies and RL-RC architecture

Task	Baseline RL policy			RL-RC architecture		
	Source	Target - model errors	Target - side force	Source	Target - model errors	Target - side force
LK	1000 ± 0 49418.0 ± 270.7	926.3 ± 177.8 44644.5 ± 9538.6	1000 ± 0 47102.9 ± 542.2	1000 ± 0 48688.1 ± 221.3	1000 ± 0 48956.0 ± 362.8	1000 ± 0 48959.2 ± 110.4
LC	1000 ± 0 47209.7 ± 213.3	775.8 ± 239.3 33990.2 ± 12007.3	963.0 ± 111.0 42578.0 ± 5800.9	1000 ± 0 47363.2 ± 303.2	1000 ± 0 47150.9 ± 361.1	1000 ± 0 47385.2 ± 241.4
OA	1000 ± 0 47348.1 ± 318.4	749.0 ± 239.1 33283.7 ± 11846.1	735.3 ± 256.5 31655.7 ± 12017.1	1000 ± 0 47661.8 ± 214.7	1000 ± 0 47521.2 ± 319.1	1000 ± 0 47694.8 ± 374.8

[1] LK stands for lane keeping. LC stands for lane changing. OA stands for obstacle avoidance.

[2] Data is presented in form of $\mathrm{mean} \pm \mathrm{std}$. In each cell, episodic length and total reward are listed from top to bottom.