# Vehicular Performance Modeling for Longitudinal Motion Planning to Satisfy Arrival Requirements

Ty Nguyen[1], Dung Nguyen[1], and Tsz-Chiu Au[1]

*Abstract*— **Motion planning with predictable timing and velocity will enable a number of interesting applications such as autonomous intersection management (AIM). These planning algorithms depend on an accurate model of the performance of the vehicular controllers, which can be highly non-linear. Au et al. proposed a motion planning algorithm to satisfy the arrival requirements in AIM. However, they assumed that the performance models are given for every road and did not discuss how to learn these models. In this paper, we propose an instance-based learning approach to learn the performance models automatically, and argue that instance-based learning is suitable for this learning task because performance models for different roads can have a high correlation with each other. Moreover, an exploration strategy based on the principle of least effort is given to speed up the learning process. Our experiments showed that the instance-based learning method with distance-based exploration strategy offers a faster learning rate than the artificial neural network methods.**

## I. INTRODUCTION

Motion planning algorithms often assume an accurate model of physical systems for the reliable execution of motion plans. However, this assumption does not hold in many real world situations, causing errors in plan execution. For example, Au et al. [1] considered the problem of controlling a mobile robot (e.g., an autonomous vehicle) to arrive at a given position on a one dimensional trajectory at a specific arrival time and a specific arrival velocity. This motion control with *arrival requirements* is fundamental in a number of multi-robot systems, in particular autonomous intersection management (AIM) which coordinates vehicles to enter an intersection in unison, leading to a much lower traffic delay than traffic signals and stop signs [2]. In some sport games such as robot soccer, the question of whether a player can move to a target position at certain time and at certain velocity to hit a ball is important in role assignment and formation positioning [3]. The motion planning algorithms in [1] depend on an accurate model of the vehicle's performance on every road in order to plan ahead to meet the arrival requirements. However, [1] assumed the performance models are given and did not discuss how to build them. Since a vehicle will run in a wide variety of road conditions, we could not obtain the performance models for running on all possible roads ahead of time. Therefore, it is necessary to employ some machine learning techniques to conduct vehicle performance profiling automatically while driving on an unfamiliar road.

[1]School of Electrical and Computer Engineering, Ulsan National Institute of Science and Technology, UNIST-gil 50, Eonyang-eup, Ulij-gun, Ulsan, Republic of Korea {tynguyen,dung,chiu}@unist.ac.kr

In this paper, we consider two approaches for learning a performance model of a vehicle. The first approach is artificial neural networks, which is a popular method for function approximation. The second approach is the instance-based learning approach, which relies on the correlation between performance models for different roads. In addition, to improve the rate of convergence of these learning algorithms, we propose a new exploration strategy that gathers samples with the least effort first. A fast learning algorithm can reduce the model error that would have an impact on the performance of the motion planner that utilizes the performance model on the go. Hence, we conduct experiments to evaluate these approaches and find out which one can learn faster.

This paper is organized as follows. After presenting the related work in Section II, we give the definition of a performance model in Section III. Then we define a longitudinal motion planning problem that utilizes a performance model in planning in Section IV. Section V gives the details of the two learning approaches, and Section VI describes the exploration strategy. Finally, we present experimental results in Section VII before we conclude in Section VIII.

## II. RELATED WORK

The acceleration profiles of a vehicle can be used for motion planning in autonomous vehicles, estimation of fuel consumption and emission, crash simulation, etc. Effectively modeling the acceleration behavior of vehicles is essential to the control of autonomous or semi-autonomous vehicles, which in turn affects the performance of transportation systems. For instance, modeling vehicle accelerations is important in motion planning in AIM as discussed in the introduction. Thus, researchers have been trying to develop models to predict the acceleration profiles of a vehicle. In practice, there are several difficulties in obtaining a set of precise equations describing the accelerating behavior of a vehicle under a particular traffic condition, including the imperfect vehicle dynamics, noise and the complexity in modeling the environment characteristics. Therefore, data-driven methods have been serving as an alternative approach to acquire a model of the accelerating behavior. Machine learning is the key to build these models, as discussed in a survey of model learning for robot control presented by Nguyen-Tuong and Peters [4].

In general, data-driven methods can be divided into two main categories: parametric approaches and nonparametric approaches. In parametric approaches, a model can fall into the categories of either kinematics models or dynamics models. Kinematics models consider the mathematical relation-

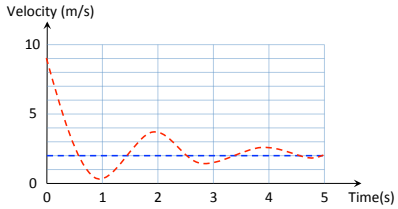
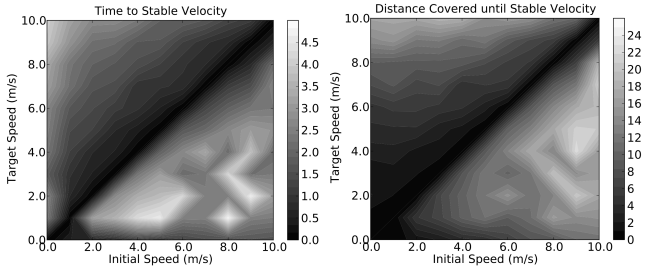



Fig. 1. An example of the velocity response of a PID controller. Overshooting occurs when the vehicle decelerates to a velocity that is close to zero.




(a) Stable Time (b) Stable Distance

Fig. 2. Stable time and stable distance for the vehicle in [1]. The light color means longer time and distance, as indicated in the bars beside the graphs.

ship between acceleration, speed, and distance that a vehicle traveled. The most simple kinematics models are the constant acceleration model, the linear decay model [5], and the dual-regime model [6], to name a few. These models basically attempt to empirically construct mathematical expressions that describe how the vehicle accelerates. However, the determinant factor of the acceleration process—the tractive force provided by the engine and the opposing resistance forces are ignored. For this reason, these kinematics models cannot provide reasonable fitting to field data.

Both of the tractive effort and resistance forces which act on the vehicle's body and control the vehicle's motion can be taken into account to develop vehicle dynamics models. Rakha et al. was the first to bring forth a constant power model and a variable power model to determine the performance of trucks [7]. Many efforts have been followed by [8] and [6] to calibrate the dynamics models. Although these dynamics models provide a good fit to the field data, it is hard to decide which breaking points are appropriate for different regimes, not to mention that these breaking points are subject to variation as data sets change. Besides, these models need intensive calibration before using them.

The downsides of the parametric models are twofold. First, the majority of parametric models only predict the maximum acceleration capabilities of a vehicle. Second, due to the limit of the number of parameters, it is hard to estimate highly nonlinear terms with measurement noise. For these reasons, nonparametric estimation can be a suitable alternative. For example, Kim and Oh adopted an artificial neural network model to predict the next state of the vehicle given the current vehicle state, the current steering angle of the wheels, and the vehicle's velocity [9]. The neural network model is associated with the hybrid learning scheme. In addition, Park et al. introduced a speed prediction algorithm with a model that is trained with the historical traffic data and capable of predicting the vehicle speed profile by using current traffic information [10].

## III. PERFORMANCE MODELS

The goal of modeling vehicle performance is to allow long-term planning of vehicle's movement without knowing the details of vehicle dynamics and controls. This separation of high-level planning from lower-level vehicle controls enables our planning procedures, called setpoint schedulers, to work with a wide variety of vehicle hardware with different underlying control mechanisms. Longitudinal control of autonomous vehicles are usually achieved by throttle and braking systems coupled with sensors such as odometers and speedometers using PID-controllers. A setpoint is the target velocity given to the PID-controllers so as to control the vehicle to reach the velocity. However, due to the complexity of the system, it is often hard to tune the PID gains to achieve a smooth transition after changing the setpoint. For example, if the autonomous vehicle at UT Austin decelerates from 9 m/s to 2 m/s, it takes 4.7 s to stabilize and the stable distance is 19.3 m—-a rather long stable time and stable distance. This problem is due to overshooting as illustrated in Fig. 1.

Fortunately, for planning purpose it is not necessary to take every detail of the vehicular behavior into account. Given the current velocity $v$ and a setpoint $\hat{v}$, the setpoint scheduler only needs to know how long the PID controllers will take to stabilize at $\hat{v}$ after setting the setpoint to $\hat{v}$, and how much the vehicle will move before its velocity is stabilized. Thus we propose to estimate two functions $T^{\text{stable}}$ and $D^{\text{stable}}$, where $T^{\text{stable}}(v,\hat{v})$ is the time the vehicle takes to stabilize at $\hat{v}$ and $D^{\text{stable}}(v,\hat{v})$ is the distance the vehicle travels after setting the setpoint to $\hat{v}$ for a period of $T^{\text{stable}}(v,\hat{v})$. We call $T^{\text{stable}}(v,\hat{v})$ and $D^{\text{stable}}(v,\hat{v})$ the *stable time* and the *stable distance*, respectively. The *performance model* of the vehicle is the pair $(T^{\text{stable}}, D^{\text{stable}})$. Fig. 2 shows the performance model of a vehicle in a table format.

The performance model in Fig. 2 is built via an empirical performance profiling of the PID controllers for the brake and throttle actuators. Our previous work assumed this profiling is given [11]. This paper discusses in detail how this profiling can be achieved by online machine learning techniques.

## IV. SETPOINT SCHEDULING PROBLEMS

Let us define a setpoint scheduling problem that utilizes a performance model. Suppose a vehicle is moving on a one-dimensional trajectory such as a road. We are interested to control the vehicle to arrive at a destination on a road at a given arrival time $t_{\text{end}}$ and at a given arrival velocity $v_{\text{end}}$. We define 1) the *initial configuration* as $(t_0, v_0)$, 2) the *arrival configuration* as $(t_{\text{end}}, v_{\text{end}})$, and 3)

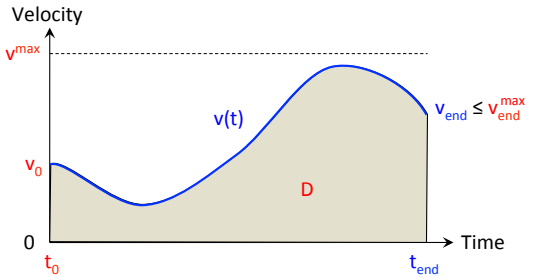



Fig. 3. The time-velocity diagram.

the *road configuration* as $(D, a_{\max}, a_{\min}, v_{\max})$, where $D$ is the road's length, $v_{\max}$ is the speed limit of the road, and $a_{\max}$ and $a_{\min}$ are the maximum and minimum acceleration, respectively. A *longitudinal motion planning problem* $\mathscr{P}_{\text{valid}}$ is a 3-tuple $\langle (t_0, v_0), (t_{\text{end}}, v_{\text{end}}), (D, a_{\max}, a_{\min}, v_{\max}) \rangle$, where $t_0, v_0, t_{\text{end}}$, and $v_{\text{end}}$ are initial time, initial velocity, arrival time, and arrival velocity, respectively, such that $t_0 = 0$, $0 \le v_0 \le v_{\max}$, $0 < t_{\text{end}}$, $0 \le v_{\text{end}} \le v_{\max}$, $0 < D$, $a_{\max} \ge 0$, $a_{\min} \ge 0$, and $0 < v_{\max}$. Our planning task is to generate a *setpoint schedule* such that if the vehicle follows the schedule exactly, it will reach the destination while satisfying all requirements and constraints. We denote a *setpoint schedule* by $\tau(\cdot)$. If $\tau(\cdot)$ is a step function, $\tau(\cdot)$ can be represented by a list of pairs $\langle (t_0, \hat{v}_0), (t_1, \hat{v}_1), \ldots (t_n, \hat{v}_n) \rangle$, such that $\tau(t) = \hat{v}_i$ for (1) $t_i \le t < t_{i+1}$ for $0 \le i < n$ and (2) $t_i \le t$ for $i = n$. In this paper, we assume all setpoint schedules are step functions.

One way to visualize what we are trying to achieve is to take a look at the time-velocity diagram in Figure 3. In this diagram, the line is a *velocity function* which denotes the velocity of the vehicle over time. A velocity function $v(\cdot)$ is *constructible* if there exists a setpoint schedule $\tau(\cdot)$ such that if the vehicle follows $\tau(\cdot)$, the velocity of the vehicle is $v(\cdot)$. A velocity function $v(\cdot)$ is *feasible* if it satisfies the following constraints:

1) $v(t_0) = v_0$;
2) $0 \le v(t) \le v_{\max}$ for $t_0 \le t \le t_{end}$ (i.e., the velocity cannot exceed the speed limit of the road or be negative at any point in time);
3) $\int_{t_0}^{t_{end}} v(t)\, dt = D$, where $t_{end}$ is the arrival time (i.e., the distance traveled must be $D$); and
4) $v(\cdot)$ is constructible.

A setpoint schedule $\tau(\cdot)$ is *feasible* if the velocity function constructed by $\tau(\cdot)$ is feasible.

The objective of a longitudinal motion planning problem $\mathscr{P}_{\text{valid}}$ is to check whether a feasible setpoint schedule $\tau(\cdot)$ exists, and generate $\tau(\cdot)$ if it exists. $\mathscr{P}_{\text{valid}}$ is also called an instance of the *validation problem*, in which we want to validate the given arrival configuration $(t_{\text{end}}, v_{\text{end}})$ by checking whether $(t_{\text{end}}, v_{\text{end}})$ is reachable by a feasible setpoint schedule.

## V. INSTANCE-BASED LEARNING ALGORITHMS

As aforementioned, a performance model has two functions: $T^{\text{stable}}$ and $D^{\text{stable}}$. Our learning task is to estimate these functions by function approximators. One popular function approximator is artificial neural networks (ANNs).

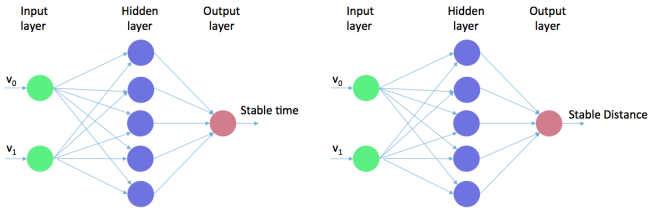



Fig. 4. A performance model represented by two artificial neural networks

A well-known algorithm for training ANNs with hidden layers is the backpropagation algorithm [12]. Fig. 4 shows the ANNs that we used as the performance model in our experiments. We used two ANNs, one for $T^{\text{stable}}$ and the other for $D^{\text{stable}}$, and they work independently although we can also build a larger, single ANN to output both $T^{\text{stable}}$ and $D^{\text{stable}}$. The input nodes are $v_0$ and $v_1$, which are the starting velocity and the setpoint, respectively. Both ANNs have a hidden layer with 5 nodes. Given the error in the output nodes, we will use the backpropagation algorithm to adjust the weights of the ANNs.

The learning process, however, can perform better if there is a way to utilize the prior knowledge about the correlation between the performance models on different roads, since a performance model is largely dominated by the vehicle's controller rather than the road conditions. Therefore, we propose an instance-based learning approach that utilizes the performance model on another road while learning the performance model. This approach involves three steps: First, compute the performance models of some *reference roads* which we consider typical among the set of roads we consider. Second, choose a reference road that is similar to the road on which the vehicle will run. Third, adjust the performance model of the chosen reference road according to the "samples" the vehicle collected by interacting with the road. Here, a sample is the stable time $t_s$ and the stable distance $d_s$ after setting a new setpoint. Given a sample $(t_s, d_s)$, we adjust the stable time function and the stable distance function, which are internally represented by two tables, according to the following update rule. Suppose the previous velocity is $v$ and the new setpoint is $v_{\text{new}}$. Let $t_s$ and $d_s$ be the stable time and the stable distance we *measured* after setting $v_{\text{new}}$ and until the velocity settles at $v_{\text{new}}$. Let $t_s'$ and $d_s'$ be the stable time and the stable distance in the performance model before the application of the update rule. Then the update rule is that for every pair $(v_1, v_2)$ of velocities in the tables.

$$T^{\text{stable}}(v_1, v_2) = \begin{cases} t_s & \text{if } v_1 = v \text{ and } v_2 = v_{\text{new}} \\ T^{\text{stable}}(v_1, v_2) + \lambda L \Delta_t & \text{otherwise} \end{cases}$$

and

$$D^{\text{stable}}(v_1, v_2) = \begin{cases} d_s & \text{if } v_1 = v \text{ and } v_2 = v_{\text{new}} \\ D^{\text{stable}}(v_1, v_2) + \lambda L \Delta_d & \text{otherwise,} \end{cases}$$

where $\lambda$ is the learning rate, $\Delta_t = t_s - t_s'$, $\Delta_d = d_s - d_s'$, and $L = ||v_2 - v_1||_2^2$. Here we assume the measurement

is noise free but our approach can still work with noisy measurements by taking the average of measurements instead of assigning a single measurement value to $T^{\mathsf{stable}}(v_1, v_2)$ and $D^{\mathsf{stable}}(v_1, v_2)$. The term $L$ gives the entries closer to the top-left corner and the bottom-right corner of the tables a larger weight. It is because these entries usually have bigger values due to the fact that there is large difference between the current velocity and the setpoint, and the vehicle will take more time and longer distance to stabilize at the setpoint.

In our experiments, the ANNs were pre-trained in the same environment that yields the performance model of the chosen reference road, so that the starting point of ANNs is equivalent to the performance model used by the instance-based learning method. This provides a fair comparison between our proposed method and ANNs in the experiments.

## VI. MIN-DISTANCE EXPLORATION STRATEGY

The two important factors in learning the performance model of a vehicle are completeness and convergence speed. For completeness, we can easily show that in our instance-based learning approach, the performance model will converge to the true model when there are enough measurements of stable times and the stable distances of with different initial and target setpoints.

On the other hand, a fast rate of convergence to the true performance model is very important because the vehicle may not have enough time to acquire all the measurements before it uses the performance model for motion planning. Obviously, the rate of convergence can be greatly improved if the vehicle can obtain a fairly accurate performance model early on by finding a good reference road. The rate of convergence also depends on the sequence of samples that the vehicle collects. In general, since the performance model $(T^{\mathsf{stable}}, D^{\mathsf{stable}})$ will be used repeatedly for many different motion planning episodes, each of them uses different parts of the model, we want the estimated model to be as *complete* as possible, meaning that we should spread out the samples so that no part of the model will be ignored forever while avoid collecting the same sample multiple times. To facilitate the sampling process, the vehicle employs an *exploration strategy*, which determines the sequence of samples so as to minimize the time it takes to learn the performance model.

Here we consider exploration strategies that have the following three characteristics: First, the vehicle will sample at the discrete velocity values only. Let $\mathbb{V} = \{n \times d\}_{n=\{0..m\}}$ be the set of velocities where $d$ is the discretization step and $d \times m$ is the maximum velocity, which is either the speed limit of the road or the maximum velocity of the vehicle. Second, after a vehicle deliberately changes its setpoint to measure the stable time and the stable distance, it will *immediately* start another measurement right after the vehicle is stabilized at the new setpoint. The result is that the ending velocity of a sampling step is always the starting velocity of the next sampling step. This way the vehicle can avoid the idle time between two samples. Third, the vehicle should avoid collecting the same sample again, because the same measurement provides less information than the new ones.

Based on these characteristics, an exploration strategy can be considered as a sequence of setpoints $\langle v_0, v_1, v_2, \ldots, v_n \rangle$, where $v_0$ is the initial velocity of the vehicle, and $v_i$ is the next setpoint after the vehicle stabilizes at $v_{i-1}$, for $i \geq 1$. The vehicle, starting with the initial velocity $v_0$, will first set its setpoint at $v_1$ and then measure the stable time and stable distance when its velocity stabilizes at $v_1$. After this measurement, it immediately sets its setpoint at $v_2$ for the second measurement. Then the process continues until the last setpoint $v_n$. To ensure completeness, this sequence of setpoints should be chosen to exhibit the property that the set of all possible consecutive pairs of setpoints (i.e., $\{(v_i, v_{i+1})\}_{i=0..(n-1)}$) is exactly the set of all possible pairs of *different* velocities in $\mathbb{V}$ (i.e., $\{(v, v') : v, v' \in \mathbb{V} \text{ and } v \neq v'\}$). Thus, under this exploration strategy, the performance model will converge to the true one.

In this paper, we propose an exploration strategy as follows: when choosing the next setpoint for the next measurement after a vehicle stabilizes at $v$, always choose the one that has the minimum stable distance according to the performance model (i.e., choose $\arg\min_{v'}\{D^{\mathsf{stable}}(v, v') : D^{\mathsf{stable}}(v, v')$ has not been measured yet.$\}$). If there is no such setpoint, choose the next setpoint randomly. We call this strategy the *min-distance* exploration strategy. We hypothesize that if an exploration strategy takes the measurements that require a shorter distance of travel first, the rate of convergence to the true performance model will be faster. We will evaluate this greedy approach in our experiments.

## VII. EXPERIMENTS

We conducted two experiments to evaluate the learning methods and the exploration strategy. In the first experiment, we ran the learning algorithms to generate vehicle's performance models and compare them to the true performance model in order to assess the errors of the generated tables. In the second experiment, we used the tables in the first experiment to generate setpoint schedules and executed them to evaluate the errors in planning. All experiments are based on a simulation platform we developed using PyGame[1], a Python library that supports 2-D simulation. In a simulation, a vehicle is running along a straight road with a slope. Fig. 5 shows a screenshot with a very short slope. In our experiments, the slope is much longer—long enough to finish a learning episode. Before each learning episode, we have to set various parameters including the angle of the slope, the air resistance, the friction force coefficients, etc. For each parameter setting, our goal is to learn a performance model when the vehicle runs on the slope.

We implemented our ANNs using PyBrain [13], a machine learning library for Python. The ANNs in our experiments are two feedforward neural networks that use the sigmoid activation function as shown in Fig. 4. Both ANNs consist of two input nodes, which correspond to the starting velocity

[1] `http://www.pygame.org`

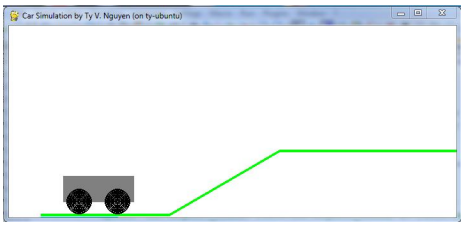



Fig. 5. A screenshot of the simulator with a short slope.

and the target velocity. We examined the effects of the number of nodes at the hidden layer and the number of hidden layers, by allowing the number of nodes at the hidden layer to vary from 2 to 10 and the number of hidden layers to vary from 1 to 10. After a thorough evaluation, we settled with one hidden layer and five nodes. As discussed in Section V, the ANNs were pre-trained in the same environment as the one generated the performance model of the chosen reference road. The ANNs were updated by using the backpropagation algorithm in PyBrain. After collecting a sample, we trained the ANNs until it cannot improve the convergence rate or the number of episodes exceeds 50.

The implementation of the instance-based learning was based on the update rule as described in Section V. The learning process started with a performance model of the reference road, and updated the performance model according to the update rule. We found that a larger learning rate can result in a faster but more uncertain convergence. In this study, the learning rate $\alpha$ is 0.5.

In our experiments, we randomly chose 30 sets of parameters in the simulation that constitute the set of reference roads. A set of parameters is a 3-tuple $\langle \theta, C_r, \rho \rangle$, where $\theta$ is the slope of the road ($0 \leq \theta \leq 60$), $C_r$ is the coefficient of rolling friction ($0.001 < C_r \leq 0.303$), and $\rho$ is the density of air ($1.146 \leq \rho 1.423$). For each set of parameters, we learnt an accurate performance model of the reference roads by exhaustively testing all possible pairs of starting velocities and setpoints. Then we used it as the performance model in our instance-based learning approach. We also ran the backpropagation algorithm for ANNs on the reference road to initialize the weights in the ANNs.

In the training stage, we considered two exploration strategies, one is the min-distance exploration strategy as discussed in Section VI and the other is a random strategy that randomly chooses the next setpoint in the learning process. We applied both strategies to both ANNs and the instance-based learning. The vehicle in the simulation used the exploration strategies to try out different sequences of setpoints such that it collects different sequences of stable times and stable distances, based on which it updates the performance models. Eventually, all possible pairs of starting velocities and setpoints would be tried and the exploration process stops. The training stage ends after the exploration process stops. Eventually, we obtained a final performance model which was solely based on the measurement of every pairs of starting velocities and setpoints, and therefore it is the true performance model.

In the first experiment, we computed the model errors against true models with respect to the training time which is the cumulative time the vehicle took to collect samples. This allows us to see how quick the learning approaches can reduce the model error. An alternative to the training time in the evaluation of the model errors is the total number of samples, but we preferred the training time for the sake of optimizing the operation time which is costly. This model errors were quantified by the root mean square errors (RMSE) of the difference of two matrices, one was the learned model and the other was the true model of the corresponding environmental conditions. Given an intermediate performance model that was acquired during learning, we subtracted the final performance model from an intermediate performance model to get an error table. For the entries in the intermediate performance model that were based on actual measurement, we set the corresponding entry in the error table to zero. Then we computed the root-mean-square of all entries in the error table.

We plotted the RMSEs as the performance models evolved over time, and the result is shown in Fig. 6 and 7. Notice that each data point in the figures is an average of 30 RMSEs of 30 trials, and the error bars represent the 95% confident intervals. As can be seen, the instance-based approach outperformed the ANN approach in terms of the learning speed in both the stable time and the stable distance. Moreover, the min-distance exploration strategy did have some positive influence on the learning speed. The only exception is that when the ANN approach was used with the min-distance exploration strategy, the learning speed was comparable to the best strategy when the training time was large. But overall the instance-based learning approach, together with the min-distance exploration strategy, had the best performance.

In the second experiment, we used the intermediate models generated during the training stage to compute setpoint schedules, and then evaluated these setpoint schedules using the simulation platform. The purpose of this experiment is to figure out which learning method is better for our setpoint scheduling problem. In order to evaluate the intermediate performance models with a given problem configuration, we implemented the bisection method as described in [1] to generate a setpoint schedule and executed it in the simulation platform under the corresponding set of parameters. Each experiment includes the following steps: 1) randomly chose a road with a set of parameters as in the first experiment and generated the reference performance models; 2) learnt performance models for this road with different learning methods and exploration strategies to generate intermediate models; 3) randomly generated 30 different setpoint scheduling problems and use the bisection method to generate 30 corresponding motion plans for each intermediate model; and 4) executed those plans and observed the errors in arrival time, velocity, and distance.

We repeated the above procedure 50 times and report the result. Due to the specific property of the bisection method which put arrival time and arrival velocity at a high priority, the errors in arrival time and arrival velocity are

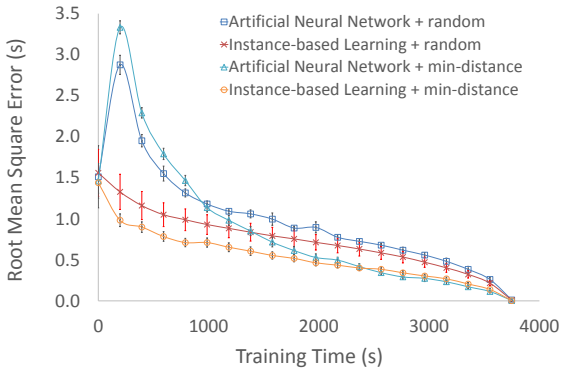



Fig. 6. RMSEs of stable time versus training time

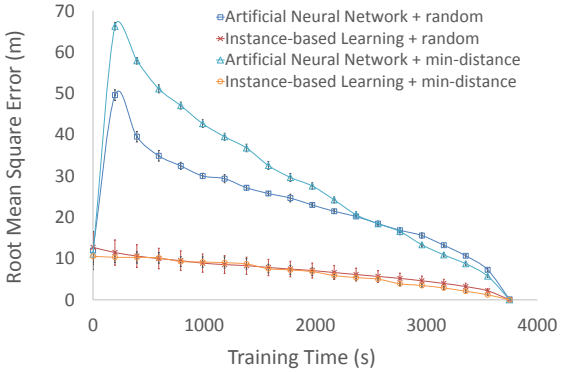



Fig. 7. RMSEs of stable distance versus training time

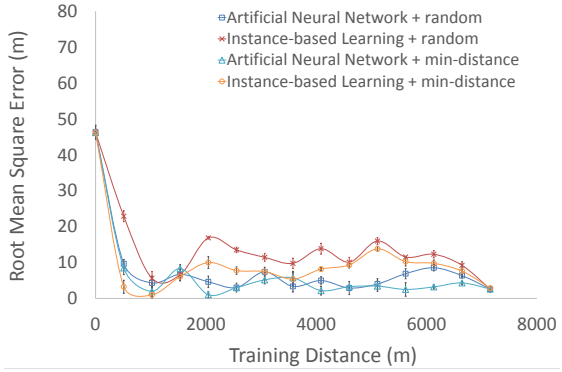



Fig. 8. RMSEs of execution distance versus training distance.

quite small. Therefore, we only report the errors in distance which is shown in Fig. 8. From this figure, we can see that the ANN approach produced smaller distance errors than the instance-based learning approach did while the min-distance exploration strategy yielded a better accuracy than the random exploration strategy when being combined with each learning approach.

## VIII. CONCLUSIONS

For precise vehicle control, motion planning algorithms often rely on a performance model of controllers that accurately describes how the vehicle interacts with the road. In this paper, we focused on learning a behavior-based performance model of a vehicle with non-linear control. We proposed and evaluated an instance-based learning approach based on the fact that the behavior of a vehicle on different roads has a high correlation. This approach adopts the performance model for a different road that is similar to the current road, and update it according to an update rule. We compared this approach to artificial neural networks that are pre-trained on the same reference road. An exploration strategy based on the principle of least effort was proposed to speed up the learning process. Our experiments showed that the instance-based learning approach has a higher rate of convergence to the true performance model than ANNs. However, the ANN approach, when paired with the min-distance exploration strategy, performed best in our planning problems that aim to satisfy the arrival time and arrival velocity requirements. In the future, we will examine other methods such as Gaussian processes to facilitate the learning process.

## ACKNOWLEDGMENTS

This work has taken place in the ART Lab at Ulsan National Institute of Science & Technology (UNIST). ART research is supported by NRF (2.180186.01 and 2.170511.01) and the 2017 Research Fund (1.170061.01) of UNIST.

## REFERENCES

[1] T.-C. Au, M. Quinlan, and P. Stone, "Setpoint scheduling for autonomous vehicle controllers," in *IEEE International Conference on Robotics and Automation (ICRA)*, 2012, pp. 2055–2060.

[2] K. Dresner and P. Stone, "A multiagent approach to autonomous intersection management," *Journal of Artificial Intelligence Research (JAIR)*, March 2008.

[3] P. MacAlpine, E. Price, and P. Stone, "Scram: Scalable collision-avoiding role assignment with minimal-makespan for formational positioning," in *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, 2015.

[4] D. Nguyen-Tuong and J. Peters, "Model learning for robot control: a survey," *Cognitive Processing*, vol. 12, no. 4, pp. 319–340, 2011.

[5] D. R. Drew, "Traffic flow theory and control," Tech. Rep., 1968. [Online]. Available: http://trid.trb.org/view.aspx?id=115219

[6] G. H. Bham and R. F. Benekohal, "Development, evaluation, and comparison of acceleration models," in *81st Annual Meeting of the Transportation Research Board, Washington, DC*, 2002.

[7] H. Rakha, I. Lucic, S. H. Demarchi, J. R. Setti, and M. V. Aerde, "Vehicle dynamics model for predicting maximum truck acceleration levels," *Journal of transportation engineering*, vol. 127, no. 5, pp. 418–425, 2001. [Online]. Available: http://ascelibrary.org/doi/abs/10.1061/(ASCE)0733-947X(2001)127:5(418)

[8] J. Searle, "Equations for Speed, Time and Distance for Vehicles Under Maximum Acceleration," SAE Technical Paper, Tech. Rep., 1999. [Online]. Available: http://papers.sae.org/1999-01-0078/

[9] Y. U. Yim and S.-Y. Oh, "Modeling of vehicle dynamics from real vehicle measurements using a neural network with two-stage hybrid learning for accurate long-term prediction," *Vehicular Technology, IEEE Transactions on*, vol. 53, no. 4, pp. 1076–1084, 2004. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=1317211

[10] J. Park, D. Li, Y. L. Murphey, J. Kristinsson, R. McGee, M. Kuang, and T. Phillips, "Real time vehicle speed prediction using a neural network traffic model," in *Neural Networks (IJCNN), The 2011 International Joint Conference on*. IEEE, 2011, pp. 2991–2996. [Online]. Available: http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6033614

[11] T.-C. Au and P. Stone, "Motion planning algorithms for autonomous intersection management," in *AAAI 2010 Workshop on Bridging The Gap Between Task And Motion Planning (BTAMP)*, 2010.

[12] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *Nature*, vol. 323, pp. 533–536, 1986.

[13] T. Schaul, J. Bayer, D. Wierstra, Y. Sun, M. Felder, F. Sehnke, T. Rückstieß, and J. Schmidhuber, "Pybrain," *Journal of Machine Learning Research*, vol. 11, pp. 743–746, 2010.