Cooperative Ramp Merging System: Agent-Based Modeling and Simulation Using Game Engine

Article · May 2019

DOI: 10.4271/12-02-02-0008

Ziran Wang
Toyota Motor North America, InfoTech Labs
27 PUBLICATIONS **142** CITATIONS
SEE PROFILE

Guoyuan Wu
University of California, Riverside
136 PUBLICATIONS **1,097** CITATIONS
SEE PROFILE

Kanok Boriboonsomsin
University of California, Riverside
144 PUBLICATIONS **2,667** CITATIONS
SEE PROFILE

Matthew J. Barth
University of California, Riverside
405 PUBLICATIONS **7,173** CITATIONS
SEE PROFILE

ARTICLE INFO
Article ID: 12-02-02-0008
Copyright © 2019
SAE International
doi:10.4271/12-02-02-0008

Cooperative Ramp Merging System: Agent-Based Modeling and Simulation Using Game Engine

Ziran Wang, Guoyuan Wu, Kanok Boriboonsomsin, and Matthew J. Barth, *University of California, Riverside, USA*
Kyungtae Han, BaekGyu Kim, and Prashant Tiwari, *Toyota Motor North America, USA*

Abstract

Agent-based modeling and simulation (ABMS) has been a popular approach for modeling autonomous and interacting agents in a multi-agent system. Specifically, ABMS can be applied to connected and automated vehicles (CAVs) since CAVs can operate autonomously with the help of onboard sensors, and cooperate with each other through vehicle-to-everything (V2X) communications. In order to improve energy efficiency and mobility of traffic, we have developed an online feedforward/feedback longitudinal controller for CAVs to cooperatively merge at ramps. Agent-based CAV models were built in the Unity3D environment, where vehicles are given connectivity and autonomy through C#-based scripting application programming interface (API). Agent-based infrastructure model is also built as a Unity3D simulation network based on the city of Mountain View, California. A simulation of cooperative on-ramp merging is carried out with a distributed consensus-based protocol, and then compared with the human-in-the-loop simulation where the on-ramp merging vehicle is driven by four different human drivers on a driving simulator. The benefits of introducing the proposed protocol are evaluated in terms of travel time, energy consumption, and pollutant emissions. The results show that the proposed cooperative on-ramp merging protocol can reduce average travel time, energy consumption, and pollutant emissions by 7%, 8%, and 58%, respectively, when compared to the human-in-the-loop scenario.

History

Received:	20 Jan 2019
Revised:	31 Mar 2019
Accepted:	16 Apr 2019
e-Available:	16 May 2019

Citation

Wang, Z., Wu, G., Boriboonsomsin, K., Barth, M. et al., “Cooperative Ramp Merging System: Agent-Based Modeling and Simulation Using Game Engine,” *SAE Int. J. of CAV* 2(2):2019, doi:10.4271/12-02-02-0008.

ISSN: 2574-0741
e-ISSN: 2574-075X

Introduction

CAV and Ramp Merging

Our transportation systems around the world have been developing at a very fast pace recently, driven by the desire for different countries and regions to reach economic prosperity. It is estimated that there are more than 1 billion motor vehicles worldwide now, and it is estimated that this number will be doubled within one or two decades [1]. Intensive transportation activities have led to a variety of social and economic issues, however. For instance, from the safety perspective, it was reported that there are more than 30,000 deaths from car accidents on United States (US) highways every year [2]. From the efficiency perspective, Los Angeles, for example, tops the 2017 global congestion ranking, with an average of 102 hours spent in traffic jams per commuter [3].

Connected and automated vehicle (CAV) technology is regarded as one of the transformative solutions for addressing the aforementioned issues. Connected vehicles (CVs) utilize different communication technologies to communicate with other vehicles on the road, roadside infrastructure, and the "Cloud" [4]. Automated vehicles (AVs) use a variety of methodologies (e.g., radar, LiDAR, computer vision) to perceive its surrounding, and therefore has the capability to operate without direct driver input of the steering, acceleration, and braking. CAVs take advantage of both CVs and AVs, leveraging connectivity and automation. Not only they can operate in isolation from other vehicles using onboard sensors, but they also can communicate with nearby vehicles and infrastructure to make decisions in a cooperative manner.

Researchers around the world have been developing various CAV applications to address traffic-related issues and improve efficiency and safety in specific traffic scenarios, such as highway on-ramp merging. A literature review on the coordination of CAVs merging at highway on-ramps was conducted by Rios-Torres et al., which summarized the developments and research trends in this research topic [5]. It can be noted that the optimal control approach has been adopted by many of the recent on-ramp merging works. Rios-Torres et al. presented an optimization framework and an analytical closed-form solution to allow online coordination of merging vehicles [6]. A proactive optimal merging strategy was proposed by Awal et al. to compute the optimal merging order for vehicles coming from main line and on-ramp, which introduces different benefits in terms of energy consumption, merging efficiency, and traffic flow [7]. Raravi et al. proposed an approach to optimize the time-to-conflict-zone for every vehicle once their merging sequences are defined [8]. Model Predictive Control (MPC) scheme was adopted by Cao et al. to generate a cooperative merging path for vehicles to merge smoothly into the main line [9].

In addition to the optimization-based approach, some other approaches have also been developed for on-ramp merging. Milanes et al. developed a fuzzy-logic method to allow vehicles to merge from the ramp onto main line fluidly without causing congestion on the ramp, while changing the speed of mainline vehicles to minimize the effect on the already-congested main line [10]. Marinescu et al. developed a slot-based algorithm for merging vehicles to cooperate with each other in a highly efficient manner [11]. Uno et al. used the virtual vehicle-platooning concept to map a virtual vehicle onto the main line before it actually merges [12]. Lu et al. adopted a centralized controller to interchange relevant information with merging vehicles, and each merging vehicle conducts its own control actions to achieve the assigned time and reference speed requirements [13, 14].

The approaches proposed in these previous works have one or more of the following limitations: (1) Some of the methods are not always suitable for real-time implementation due to the difficulty in finding an optimal solution; (2) a vehicle is considered in a single form, making it difficult to extend to a vehicle string; (3) benefits of energy efficiency and pollutant emissions are not typical considered. The cooperative ramp merging system proposed in this article is aimed at addressing these limitations.

Agent-Based Modeling and Simulation

Agent-based modeling and simulation (ABMS) focuses on microscale models that simulate the simultaneous operations and interactions of multiple agents [15]. There is no universal definition of the term "agent"; however, certain characteristics are often shared by agents from a modeling standpoint [16]. Those characteristics include (a) identifiable, with rules governing their decision-making capabilities; (b) interactive, with the ability to recognize and distinguish the traits of other agents; (c) goal-directed, with goals to achieve with respect to their behaviors; (d) autonomous, with the capability to function independently in their environment; (e) flexible, with the ability to learn and adapt their behaviors over time based on experience. Given the fact that CAVs can fulfill the above characteristics to a large extent, ABMS is considered an attractive approach for modeling transportation systems comprised of CAVs.

Many different tools for the modeling and simulation of CAVs are available. From the traditional four-step travel demand models to the state-of-the-art agent-based models, they all have their unique advantages and thus are suited for different purposes [17]. The emergence of the CAV technology triggers a challenging system modeling problem: Traditional modeling tools that consider only one target vehicle can no longer be adopted since we need to also consider a CAV's surrounding environment, such as other vehicles, road infrastructure, and pedestrians. Microscopic traffic simulators, such as SUMO, VISSIM, and Aimsun, provide the options for researchers to model a relatively large amount of vehicles in a traffic environment [18, 19, 20]. However, users of such simulators cannot model full dynamics of vehicles, and neither can they get involved in the control of vehicles. Conversely, game engines are able to model complex virtual reality environments and also allow users (i.e., game players) to get fully immersed in the simulation game. Therefore, in this work we adopt the Unity3D game engine to conduct ABMS of CAVs in a case study of cooperative on-ramp merging.

The Unity3D game engine used in this work integrates a custom rendering engine with the Nvidia PhysX physics engine and Mono, the open source implementation of Microsoft's NET libraries [21]. Unity3D has been widely used to build simulation platforms. Graighead et al. implemented the Search and Rescue Game Environment (SARGE) with Unity3D, where robotic vehicles in this environment are equipped with various sensors, such as 3D camera, GPS, odometer, inertial measurement unit (IMU), and planar laser ranging [22]. KTH Royal Institute of Technology in Sweden conducted several studies on the visualization of truck platooning using Unity3D [23, 24]. Toyota InfoTechnology Center in the USA also contributed a series of work to the vehicle prototyping research by Unity3D. Yamaura et al. built a virtual prototype of advanced driver-assistance systems (ADAS) with a closed-loop simulation framework that consists of four tools: Unity3D, Simulink, OpenMETA, and Dymola [25]. Kim et al. proposed several research directions and potential approaches for testing autonomous vehicle software in a virtual prototyping environment using Unity3D, from the perspective of test criteria and test case generation [26]. As an extended work of that work, Dai et al. presented a co-simulation toolchain for the automated optimization of various parameters in the virtual prototyping environment [27]. In general, Unity3D has the following advantages over other simulation tools:

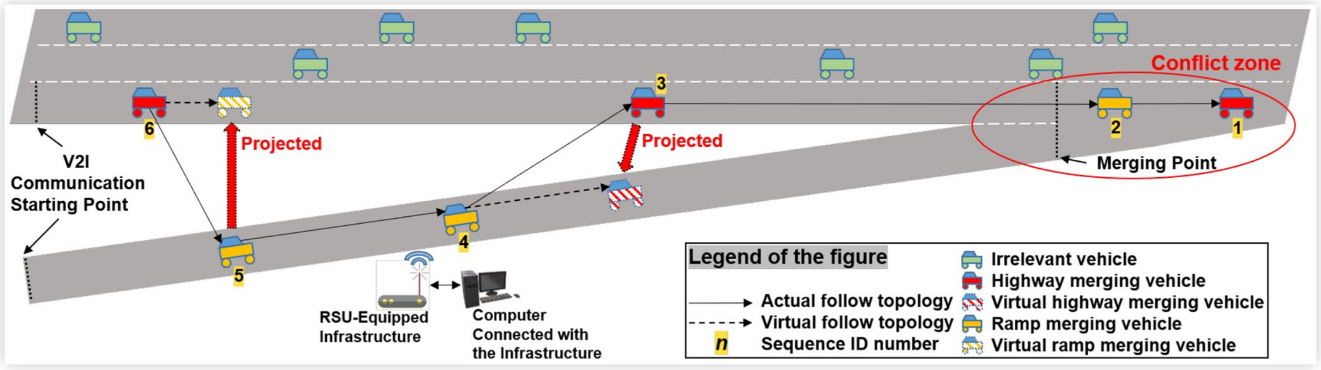
a. Graphics and visualization: Since Unity3D is designed for developers to develop 3D video games, it has an impressive capability of graphics rendering and visualization. It streamlines the demonstration of the proposed CAV technology to the audience, especially to the general public (without knowing technical details). This is the primary reason why we selected Unity3D for performing ABMS of CAVs.
b. Integration with driving simulator: Unity3D provides easy access to change the input equipment, which makes it possible to integrate it with driving simulator hardware. Since we want to compare the proposed CAV technology with the baseline, using driving simulator hardware with human-in-the-loop simulation is more realistic than simply applying some human driver models in the simulation.
c. Asset store: Unity3D has an official asset store where Unity3D developers and users can upload and download different Unity3D assets, which allows Unity3D users to develop their own game environment based on others' previous work, instead of building things from scratch.
d. Documentation and community: Unity3D provides thorough, well-organized, and easy-to-read documentation covering how to use each component in Unity3D, and an online commUnity website for all Unity3D users to ask and answer questions.

The structure of this article is as follows: The following section introduces the general architecture of the proposed cooperative ramp merging system, together with assumptions of this study. Then the proposed online feedforward/feedback longitudinal controller for CAVs is demonstrated, including an online parameters modeling algorithm and a distributed consensus longitudinal control algorithm. The agent-based modeling of CAVs and infrastructures, and the agent-based simulation of the proposed system are included next. The last section concludes the article and introduce some potential next steps of this study.

System Architecture

The proposed cooperative on-ramp merging system utilizes V2X communications of CAVs, which means vehicles can communicate with each other through V2V communications, and with the infrastructure through V2I communications. The illustrative system architecture is shown in Figure 1. When merging vehicles from the upstream of on-ramp and main line enter the V2I communication range of the infrastructure, they send the infrastructure their own information measured by onboard sensors. That information includes but is not limited to current acceleration, speed, and position of the vehicle. Then, the computer connected to the infrastructure processes all the information gathered from on-coming vehicles within a certain

FIGURE 1 Illustrative architecture of the V2X-based cooperative on-ramp merging system.



time interval, and assigns a series of sequence identification numbers to different vehicles based on the vehicle sequencing protocol. Vehicles retrieve those sequence identification numbers from the infrastructure at the next time step.

Once a vehicle gets its own sequence identification number, it can automatically match its predecessor, which is not necessarily the one in front. In the case that the ego CAV and its predecessor are not on the same lane (such as vehicles 6 and 5 in Figure 1), the predecessor is strategically projected as a virtual CAV on the ego CAV's lane, where the virtual CAV has the same longitudinal speed and position as the original CAV. Note that longitudinal position of CAVs in this on-ramp merging system are calculated by their distance to the merging point, so the relative longitudinal position between any two CAVs is the difference between their distances to the merging point. After the predecessor of the ego CAV is decided, the proposed online feedforward/feedback longitudinal controller can be applied, so the ego CAV will cooperatively merge with its predecessor before reaching the merging point. In this manner, the possibility of rear-end or side collision in the conflict zone will be largely decreased compared to human driving.

In this study, we focus on developing the online feedforward/feedback longitudinal controller for the CAV and build agent-based CAV model and infrastructure model. The aforementioned vehicle sequencing protocol is adopted from one previous study, where details of the protocol are not introduced in this study and can be referred to the literature [28]. However, the vehicle sequencing protocol is integrated in the agent-based infrastructure model in Unity3D environment, and also gets simulated in this study.

It should be noted that some reasonable specifications and assumptions are made in this work, as follows:

a. All vehicles in this system are CAVs with appropriate onboard sensors and communication equipment, and all hardware functions perfectly without any error or noise.
b. Only vehicles on the right-most main line are considered, and no cut-in maneuver is conducted by mainline vehicles which are in other lanes.
c. Only the longitudinal control is discussed in this system, while the lateral movements of vehicles do not affect their longitudinal movements.

Online Feedforward/ Feedback Longitudinal Controller

Problem Statement

First, the longitudinal dynamics of a vehicle i can be given as the following equations:

$$\dot{r}_i(t) = v_i(t) \quad \text{Eq. (1)}$$

$$\dot{v}_i(t) = a_i(t) \quad \text{Eq. (2)}$$

$$a_i(t) = \frac{1}{m}\left[F_{net_i}(t) - R_i T_{br_i}(t) - c_{vi} v_i(t)^2 - c_{pi} v_i(t) - d_{mi}(t)\right] \quad \text{Eq. (3)}$$

where $r_i(t)$, $v_i(t)$, and $a_i(t)$ denote the longitudinal position, longitudinal speed, and longitudinal acceleration of vehicle i at time t, respectively; m_i denotes the mass of vehicle i; $F_{net_i}(t)$ denotes the net engine force of vehicle i at time t, which mainly depends on the vehicle speed and the throttle angle; R_i denotes the effective gear ratio from the engine to the wheel of vehicle i; $T_{bri_i}(t)$denotes the brake torque of vehicle i at time t; c_{vi} denotes the coefficient of aerodynamic drag of vehicle i; c_{fi} denotes the coefficient of friction force of vehicle i; and $d_{mi}(t)$ denotes the mechanical drag of vehicle i at time t.

The following equations can then be derived from the principle of vehicle dynamics when the braking maneuver is deactivated, i.e., vehicle i is accelerating by the net engine force:

$$F_{net_i}(t) = \ddot{x}_i(t) m_i + c_{vi}\dot{x}_i(t)^2 + c_{pi}\dot{x}_i(t) + d_{mi}(t) \quad \text{Eq. (4)}$$

and when the braking maneuver is active, i.e., vehicle i decelerates by the brake torque:

$$T_{br_i}(t) = \frac{\ddot{x}_i(t) m_i + c_{vi}\dot{x}_i(t)^2 + c_{pi}\dot{x}_i(t) + d_{mi}(t)}{R_i} \quad \text{Eq. (5)}$$

It should be noted that the net engine force is a function of the vehicle speed and the throttle angle, which is generally based on the steady-state characteristics of engine and transmission systems, and the mathematical derivation can be referred to [29, 30].

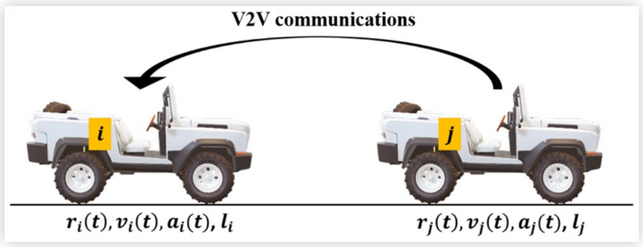
Generally, the longitudinal control command of a vehicle is based on a hierarchical strategy, where the high-level controller (Equations 1 and 2) generates a target acceleration, while the low-level controller commands the vehicle actuators to track the target acceleration (Equation 3). In this work, we focus on the high-level vehicle controller, where we propose the online feedforward/feedback longitudinal controller based on the predecessor following information flow topology of a string of vehicles, where the following vehicle only gets information from its immediate leading vehicle through V2V communications, given that the problem can be generalized as a longitudinal control problem shown as Figure 2.

In this figure, the term l_j denotes the length of vehicle j. As can be seen, the following vehicle i receives information from the leading vehicle j through V2V communications. Therefore, the problem of forming a predecessor following a string of vehicles can be formulated, given l_i and l_j, and initial states $r_i(0)$, $v_i(0)$, $a_i(0)$, $r_j(0)$, $v_j(0)$, $a_j(0)$, how to apply a longitudinal control algorithm such that

$$r_i(t) \rightarrow r_j(t) - r_{headway} \quad \text{Eq. (6)}$$

$$v_i(t) \rightarrow v_j(t) \quad \text{Eq. (7)}$$

FIGURE 2 Illustration of the predecessor following information follow topology.

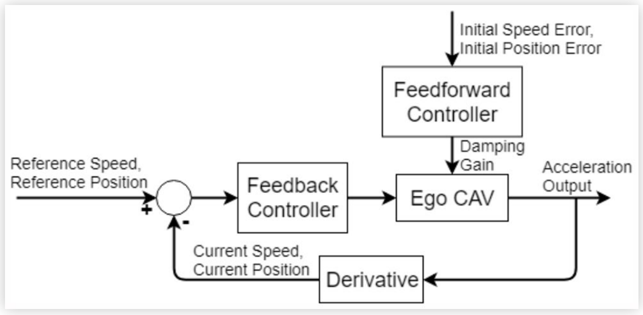


$$a_i(t) \to a_j(t) \qquad \text{Eq. (8)}$$

where "→" means the value on the left-hand side converges to the value on the right-hand side and $r_{headway}$ is the desired distance headway between two vehicles.

Most existing relevant works adopt the same set of control parameters (i.e., damping gains) independent of certain driving scenario characteristics [31, 32, 33, 34]. Such uniform assignment of control gains may not guarantee the constraints of the proposed longitudinal control algorithms under some driving scenarios. For example, the algorithm might work well when the initial speed difference of two vehicles are relatively small, and the initial headway is relatively large. However, when the initial conditions of these parameters change dramatically but the control gains remain the same, some overshoot of the headway might appear during the convergence process. Although the dynamics of two vehicles will still converge to consensus eventually, safety constraints cannot be satisfied since rear-end collision between vehicles will happen. The convergence to stability might also take a very long time, making the algorithm inefficient. Additionally, the speed or acceleration can change dramatically during a short period without considering the riding comfort of human passengers. These are the issues we want to address in this study by developing an online feedforward/feedback longitudinal controller, where the block diagram is shown as Figure 3.

FIGURE 3 Block diagram of the online feedforward/feedback longitudinal controller.



Feedback Control: Distributed Consensus Longitudinal Control Algorithm

The information flow topology of a string of vehicles can be represented by a directed graph $\mathcal{G} = (\mathcal{H}, \mathcal{E})$, where $\mathcal{H} = \{1, 2, \ \ldots, \ n\}$ is a finite nonempty node set and $\mathcal{E} \subseteq \mathcal{H} \times \mathcal{H}$ is an edge set of ordered pairs of nodes (i.e., edges). The edge $(i, j) \in \mathcal{E}$ denotes that vehicle j can obtain information from vehicle i. It is not necessarily true vice versa, since the information flow is not always bidirectional. The neighbors of vehicle i are denoted by $\mathcal{N}_i = \{ j \in \mathcal{H} : (i,j) \in \mathcal{E} \}$. The topology of the graph is associated with an adjacency matrix $\mathcal{A} = [\alpha_{ij}] \in \mathbb{R}$, which is defined such that $\alpha_{ij} = 1$ if edge (j, i), $\in \mathcal{E}$, $\alpha_{ij} = 0$ if edge $(j, i) \notin \varepsilon$, and $\alpha_{ii} = 0$. $\mathcal{L} = [\ell_{ij}] \in \mathbb{R}$ (i.e., $\ell_{ij} = -\alpha_{ij}$, $i \neq j$, $\ell_{ii} = \sum_{j=1, j\neq i}^{n} \alpha_{ij}$) is the nonsymmetrical Laplacian matrix associated with $\mathcal{G}$. A directed spanning tree is a directed tree formed by graph edges that connects all the nodes of the graph.

A double-integrator distributed consensus longitudinal control algorithm for CAVs is proposed as

$$\dot{r}_i(t) = v_i(t) \qquad \text{Eq. (9)}$$

$$\dot{v}_i(t) = -\alpha_{ij} k_{ij} \cdot \Big[\Big(r_i(t) - r_j\big(t - \tau_{ij}(t)\big) + l_j + v_i(t) \cdot \big(t_{ij}^{g}(t) + \tau_{ij}(t) \big) \Big) + \gamma_i \cdot \Big(v_i(t) - v_j\big(t - \tau_{ij}(t)\big) \Big) \Big], \ i, j \in \mathcal{H} \qquad \text{Eq. (10)}$$

where $\tau_{ij}(t)$ denotes the time-variant communication delay between two vehicles and $t_{ij}^{g}(t)$ is the time-variant desired time gap between two vehicles, which can be adjusted by many factors like road grade, vehicle mass, braking ability, etc. The term $\big[l_j + v_i(t) \cdot \big(t_{ij}^{g}(t) + \tau_{ij}(t) \big) \big]$ is another form of the term $r_{headway}$ in Equation 6. The longitudinal position and speed converges to position consensus and speed consensus, respectively, as

$$r_i(t) \to \Big[r_j\big(t - \tau_{ij}(t)\big) - l_j - v_i(t). \ \big(t_{ij}^{g}(t) + \tau_{ij}(t) \big) \Big] \qquad \text{Eq. (11)}$$

$$v_i(t) \to v_j\big(t - \tau_{ij}(t)\big) \qquad \text{Eq. (12)}$$

If we define $\tilde{r}_i$ and $\tilde{v}_i$ as the position error and speed error of vehicle i with respect to the leading vehicle of a vehicle string (i.e., the desired values of all following vehicles in the string), then Equations 9 and 10 can be rewritten as

$$\dot{r}_i(t) = v_i(t) \qquad \text{Eq. (13)}$$

$$\dot{v}_i(t) = -\alpha_{ij} k_{ij} \cdot \Big[\Big(\tilde{r}_i(t) - \tilde{r}_j\big(t - \tau_{ij}(t)\big) \Big) + \gamma_i \cdot \ \Big(\tilde{v}_i(t) - \tilde{v}_j\big(t - \tau_{ij}(t)\big) \Big) \Big], \quad i, j \in \mathcal{H} \qquad \text{Eq. (14)}$$

If we further define the dynamics of the vehicle in a compact form as

$$\tilde{r} = \left[\tilde{r}_1^T, \tilde{r}_2^T, \ldots, \tilde{r}_i^T, \ldots, \tilde{r}_n^T\right]^T \qquad \text{Eq. (15)}$$

$$\tilde{v} = \left[\tilde{v}_1^T, \tilde{v}_2^T, \ldots, \tilde{v}_i^T, \ldots, \tilde{v}_n^T\right]^T \qquad \text{Eq. (16)}$$

then the state vector can be defined as

$$\tilde{\chi} = \left[\tilde{r}^T \tilde{v}^T\right]^T \qquad \text{Eq. (17)}$$

The double-integrator vehicle dynamics in Equations 15 and 16 can be further transformed into a compact form as

$$\dot{\tilde{\chi}}(t) = \Gamma_1 \tilde{\chi}(t) + \Gamma_k \tilde{\chi}\left(t - \tau_k(t)\right) \qquad \text{Eq. (18)}$$

$$\Gamma_1 = \begin{bmatrix} 0_{n\times n} & I_{n\times n} \\ -\tilde{A} & -\gamma \tilde{A} \end{bmatrix}, \quad \Gamma_k = \begin{bmatrix} 0_{n\times n} & 0_{n\times n} \\ \tilde{A}_k & \gamma \tilde{A}_k \end{bmatrix} \qquad \text{Eq. (19)}$$

$$\tilde{A} = \operatorname{diag}\left\{\alpha_{12}, \alpha_{23}, \ldots, \alpha_{ij}, \ldots, \alpha_{(n-1)n}\right\} \qquad \text{Eq. (20)}$$

where $\tau_k(t)$, $k = 1, 2, \ \ldots, \ m$ with $m \le n(n-1)$ is defined as an element of the time-varying communication delay $\tau_{ij}(t)$.

Given the Leibniz-Newton formula,

$$\begin{aligned} \tilde{\chi}\left(t - \tau_k(t)\right) &= \tilde{\chi}(t) - \int_{-\tau_k(t)}^{0} \dot{\tilde{\chi}}(t+s)\,ds \\ &= \tilde{\chi}(t) - \Gamma_l \int_{-\tau_k(t)}^{0} \tilde{\chi}\left(t+s-\tau_l(t+s)\right)ds \end{aligned} \qquad \text{Eq. (21)}$$

where $l = 0,1,2, \ \ldots, m$, and substitute this into Equation 18 as

$$\dot{\tilde{\chi}}(t) = B\tilde{\chi}(t) - \Gamma_k \Gamma_l \int_{-\tau_k(t)}^{0} \tilde{\chi}\left(t+s-\tau_l(t+s)\right)ds \qquad \text{Eq. (22)}$$

$$B = \Gamma_1 + \Gamma_k = \begin{bmatrix} 0_{N\times N} & I_{N\times N} \\ -\bar{A} & -\gamma \bar{A} \end{bmatrix} \qquad \text{Eq. (23)}$$

where $\bar{A} = -\tilde{A} + \tilde{A}_k$.

If there exists a directed spanning tree in the platoon information flow topology $\mathcal{G}$, and the control gain γ of Equation 10 suffices,

$$\gamma_i > \max_{\mu_i \in \eta(\bar{A})} \left\{ \frac{\left|Im\{\mu_i\}\right|}{\sqrt{\left|\operatorname{Re}\{\mu_i\}\right| \cdot \left|\mu_i\right|}} \right\} \qquad \text{Eq. (24)}$$

where μ_i is the ith eigenvalue of $\bar{A}$ and $\eta\left(\bar{A}\right)$ is the set of all eigenvalues of $\bar{A}$, then there exists a constant $\tau_0 > 0$ such that when $0 \le \tau_k \le \tau_0 (k = 1, 2, \ \ldots, \ m)$, the vehicles in the same string can achieve consensus as defined in Equations 11 and 12. Due to the limitation of space in this article, the proof of this theorem is not covered in this section, but it can be referred to the reference [28].

String stability is a desirable characteristic for vehicle strings to attenuate either distance error, velocity, or acceleration along upstream direction, and therefore guarantee the safety of the longitudinal control algorithm. If we consider vehicle i as a following vehicle (=i − 1), then we can write Equation 10 in the Laplace domain with time-variant communication delay set to a constant value τ as

$$\begin{aligned} A_i(s) = -\alpha_{i(i-1)} k_{i(i-1)} \cdot \Bigg[\Bigg(& R_i(s) - R_{(i-1)}(s) e^{-\tau s} + \frac{l_{i-1}}{s} + \\ & V_i(s) e^{-\tau s} \frac{\left(t_{i(i-1)}^g + \tau\right)}{s} \Bigg) + \gamma_i \left(V_i(s) - V_{i-1}(s) e^{-\tau s}\right) \Bigg], \quad i \in \mathcal{H} \end{aligned} \qquad \text{Eq. (25)}$$

After some algebraic manipulations when assuming low frequency condition, the following equation can be derived

$$\frac{A_i(s)}{A_{i-1}(s)} = \frac{\alpha_{i(i-1)} k_{i(i-1)} \cdot \left[e^{-\tau s} + s e^{-\tau s}\left(t_{ij}^g + \tau\right) b_i + s\gamma_i e^{-\tau s}\right]}{s^2 + \gamma_i s + 1} \qquad \text{Eq. (26)}$$

where the control gains k and γ in Equation 10 can be chosen to guarantee $\dfrac{A_i(s)}{A_{i-1}(s)} \le 1$ and hence satisfy the string stability for all frequencies of interest [28].

Feedforward Control: Online Parameters Modeling Algorithm

As can be seen in the proposed consensus algorithm (10), there are two control gains k and γ. Although most existing literature proved convergence and string stability of their proposed consensus algorithms, whether they can satisfy real-world constraints during the implementation is not known. Since most works only adopt one initial condition of vehicle in the simulation study, one single set of well-defined control gains worked well under that condition. However, given different initial states of CAVs, the consensus algorithm tends to behave differently in terms of overshoot, convergence rate, and maximum changing rate. A set of control gains working well under one initial condition does not necessarily mean working well under all other initial conditions. Finding the ideal value of control gains in real time when the initial conditions of vehicles are dynamically changing remains an unsolved problem.

In this section, we propose the feedforward control-based parameter modeling algorithms, aiming to find the ideal values of control gains in terms of different initial states of the leading vehicle and the following vehicle by building a lookup table. Given the second-order dynamics of vehicles as Equations 1 and 2, the initial condition of the following vehicle i and the leading vehicle j are $(r_i(t_0), v_i(t_0), a_i(t_0))$ and

$(r_j(t_0 - \tau_{ij}(t_0)), v_j(t_0 - \tau_{ij}(t_0)), a_j(t_0 - \tau_{ij}(t_0)))$, where t_0 denotes the initial time step when the consensus algorithm is applied. Since the proposed double-integrator consensus algorithm (10) does not consider the acceleration of the leading vehicle, and the positions of vehicles are only calculated as their difference, the initial condition of the proposed consensus algorithm can be simplified to $(\Delta r_{ij}(t_0), v_i(t_0), v_j(t_0 - \tau_{ij}(t_0)))$, where $\Delta r_{ij}(t_0) = r_j(t_0 - \tau_{ij}(t_0)) - r_i(t_0)$. The reason we cannot simplify the initial speed of two vehicles into one term is that the term $v_i(t_0)$ is also calculated in the position consensus term (11), so the value of each vehicle's speed matters to the consensus algorithm.

Every time the consensus algorithm (10) starts to run on the vehicle i, the value of control gains k_{ij} and γ can be set in real time with the initial condition of vehicles $(\Delta r_{ij}(t_0), v_i(t_0), v_j(t_0 - \tau_{ij}(t_0)))$. The method is to build a three-dimensional lookup table ahead of time covering certain possible values of the initial conditions within certain sets, and the ideal values of control gains can be picked from certain sets of candidates. The three major constraints we consider when choosing the ideal control gains are safety, efficiency, and comfort.

Constraint 1: Safety Constraint The overshoot of the algorithm influences the safety of the longitudinal motion controller. Since the consensus algorithm is proposed to control the longitudinal motion of vehicles, overshoot of the longitudinal position might cause rear-end collision between two vehicles. Therefore, the following constraint should be satisfied to guarantee the safety of the longitudinal motion controller

$$r_j\left(t-\tau_{ij}(t)\right)-r_i(t)>l_j,\ t\in\left[t_0,\ t_{consensus}\right] \quad \text{Eq. (27)}$$

where $t_{consensus}$ denotes the time step when consensus is reached. If the headway between the leading vehicle and the following vehicle is no greater than the length of the leading vehicle, a rear-end collision happens. Control gains should be set to guarantee no overshoot of the headway.

Constraint 2: Efficiency Constraint The convergence rate of the consensus algorithm influences the efficiency of the longitudinal motion controller. If the convergence process takes a relatively long time, the traffic mobility and roadway capacity are highly affected during this process. Specifically, if the consensus algorithm is applied to control the longitudinal motion of ramp merging vehicles, slow convergence rate also introduces safety issue since consensus must be reached before the two vehicles merge with each other. Control gains should be set with the least time to consensus $\min t_{consensus}$ (they need to firstly satisfy constraint (27)). Consensus is reached when the following constraints are satisfied

$$\left|r_j\left(t_{consensus}-\tau_{ij}\left(t_{consensus}\right)\right)-r_i\left(t_{consensus}\right)\right|\le\eta_r\cdot \left[l_j+v_i\left(t_{consensus}\right)\cdot\left(t_{ij}^g\left(t_{consensus}\right)+\tau_{ij}\left(t_{consensus}\right)\right)\right] \quad \text{Eq. (28)}$$

$$\left|v_j\left(t_{consensus}-\tau_{ij}\left(t_{consensus}\right)\right)-v_i\left(t_{consensus}\right)\right| \le\eta_v\cdot v_j\left(t_{consensus}-\tau_{ij}\left(t_{consensus}\right)\right) \quad \text{Eq. (29)}$$

$$\left|a_i\left(t_{consensus}\right)\right|\le\delta_a \quad \text{Eq. (30)}$$

$$\left|jerk_i\left(t_{consensus}\right)\right|\le\delta_{jerk} \quad \text{Eq. (31)}$$

where $jerk_i$ is the derivative of vehicle i's acceleration/deceleration; η_r and η_v are proportional thresholds of the headway consensus and speed consensus, respectively; δ_a and δ_{jerk} are differential thresholds of acceleration and jerk consensus, respectively.

Constraint 3: Comfort Constraint The maximum changing rate of the consensus algorithm is defined as the maximum absolute value of acceleration/deceleration and jerk. This factor influences the ride comfort of the proposed longitudinal motion controller. A high maximum changing rate does not necessarily mean a high convergence rate, since algorithm (10) can either converge to consensus within a relatively short time but in a smooth manner or converge to consensus within a relatively long time but change extremely fast at first. In this constraint, the maximum absolute value of acceleration/deceleration and jerk matter, since passengers on the vehicle would expect a comfort ride with acceleration/deceleration and jerk limited to certain intervals. The maximum changing rate of the consensus algorithm is evaluated by defining a parameter Ω as

$$\Omega_i=\omega_1\cdot\max_{t\in\left[t_0,t_{consensus}\right]}\left(\left|a_i^{\max}(t)\right|,\left|d_i^{\max}(t)\right|\right) +\omega_2\cdot\max_{\tau\in\left[t_0,t_{consensus}\right]}\left(\left|jerk_i^{\max}(t)\right|,\left|jerk_i^{\min}(t)\right|\right), \quad t\in\left[t_0,t_{consensus}\right] \quad \text{Eq. (32)}$$

where $a_i^{\max}$, $d_i^{\max}$, $jerk_i^{\max}$, and $jerk_i^{\min}$ denote the maximum acceleration, maximum deceleration, maximum jerk, and minimum jerk of vehicle i, respectively, and ω_1 and ω_2 are weighting parameters. Control gains should be set with the minimum value of Ω in this constraint.

Online Parameter Modeling Protocol After above three constraints are developed, we propose *Algorithm 1* to build the three-dimensional lookup table, aiming to choose the control gains. The set of Δr_{ij} contains $\zeta1$ elements, the set of v_i contains $\zeta2$ elements, and the set of v_j contains $\zeta3$ elements; therefore, the size of this lookup table is $\zeta1 \times \zeta2 \times \zeta3$. These three sets $\Pi_{\Delta r_{ij}}$, Π_{v_i}, and Π_{v_j} are sorted set in ascending order. Each combination of these three parameters maps to an ideal value of k and an ideal value of γ, out of their sets Π_γ and Π_k. Note that some specific initial condition cannot satisfy *Constraint 1*, as shown on line 04-05 of *Algorithm 1*. In that case, no value of control gains is generated, considering algorithm (10) being not functional under that particular condition.

Once the lookup table has been generated, the initial condition of vehicles does not always match certain values while implementing algorithm (10)

$$\left(\Delta r_{ij}\left(t_0\right),v_i\left(t_0\right),v_j\left(t_0-\tau_{ij}\left(t_0\right)\right)\right)\ne\left(\Delta r_{ij_{\xi1}},\ v_{i_{\xi2}},\ v_{j_{\xi3}}\right) \quad \text{Eq. (33)}$$

ALGORITHM 1 Build feedforward lookup table.

```
Input: $\Pi_{\Delta r_{ij}} = \left\{\Delta r_{ij_1}, \Delta r_{ij_2}, \dots, \Delta r_{ij_{\zeta 1}}\right\}$ , $\Pi_{v_i} = \{v_{i_1}, v_{i_2}, \dots, v_{i_{\zeta 2}}\}$ ,
$\Pi_{v_j} = \{v_{j_1}, v_{j_2}, \dots, v_{j_{\zeta 3}}\}$ , $\Pi_\gamma = \{v_{\gamma_1}, v_{\gamma_2}, \dots, v_{\gamma_{\zeta 4}}\}$ , $\Pi_k =$
$\{v_{k_1}, v_{k_2}, \dots, v_{k_{\zeta 5}}\}$, $\zeta 1 = \left|\Pi_{\Delta r_{ij}}\right|$, $\zeta 2 = \left|\Pi_{v_i}\right|$, $\zeta 3 = \left|\Pi_{v_j}\right|$
Output: 3-dimension table with size $\zeta 1 \times \zeta 2 \times \zeta 3$
01:for $\xi 1 \in [1, \zeta 1], \xi 2 \in [1, \zeta 2], \xi 3 \in [1, \zeta 3], \Delta r_{ij_{\xi 1}} \in \Pi_{\Delta r_{ij}},$
       $v_{i_{\xi 2}} \in \Pi_{v_i}, v_{j_{\xi 3}} \in \Pi_{v_j}$ do
02:    run algorithm (10) with $\Delta r_{ij_{\xi 1}}$, $v_{i_{\xi 2}}$ and $v_{j_{\xi 3}}$
03:    find $\Lambda_\gamma \subseteq \Pi_\gamma, \Lambda_k \subseteq \Pi_k$ satisfy Constraint 1
04:    if $\Lambda_\gamma = \emptyset \,||\, \Lambda_k = \emptyset$ then
05:        $\gamma_{(\xi 1, \xi 2, \xi 3)} = NaN,\ k_{(\xi 1, \xi 2, \xi 3)} = NaN$
06:    else
07:        find $\Psi_\gamma \subseteq \Lambda_\gamma, \Psi_k \subseteq \Lambda_k$ satisfy Constraint 2
08:        if $\left|\Psi_\gamma\right| == \left|\Psi_k\right| == 1$ then
09:            $\gamma_{(\xi 1, \xi 2, \xi 3)} \in \Psi_\gamma, k_{(\xi 1, \xi 2, \xi 3)} \in \Psi_k$
10:        else
11:            find $\Phi_\gamma \subseteq \Psi_\gamma, \Phi_k \subseteq \Phi_k$ satisfy Constraint 3
12:            if $\left|\Phi_\gamma\right| == \left|\Phi_k\right| == 1$ then
13:                $\gamma_{(\xi 1, \xi 2, \xi 3)} \in \Phi_\gamma, k_{(\xi 1, \xi 2, \xi 3)} \in \Phi_k$
14:            else
15:                $\gamma_{(\xi 1, \xi 2, \xi 3)} = \min_{\gamma \in \Phi_\gamma} \Phi_\gamma, k_{(\xi 1, \xi 2, \xi 3)} = \min_{k \in \Phi_k} \Phi_k$
16:            end if
17:        end if
18:    end if
19: end for
```

Therefore, in order to find the ideal values of control gains given different initial conditions, Algorithm 2 is proposed. If the initial states of vehicles fall out of the ranges, invalid values are returned as shown on line 02-03, meaning no values of γ and k could be selected by the online parameter modeling protocol. In such rare cases, algorithm (10) cannot guarantee all three constraints and will not be applied to control the vehicle. The default car-following algorithm and lane-changing algorithm equipped on the vehicle (whatever they are) will be applied to control the longitudinal and lateral vehicle motion, respectively.

ALGORITHM 2 Search lookup table in real time.

```
Input: $\left(\Delta r_{ij}(t_0), v_i(t_0), v_j\left(t_0 - \tau_{ij}(t_0)\right)\right)$ , $(\zeta 1 \times \zeta 2 \times \zeta 3)$ size
of lookup table
Output: values of control gains $k$ and $\gamma$
01:for $\Delta r_{ij}(t_0), v_i(t_0), v_j\left(t_0 - \tau_{ij}(t_0)\right)$ do
02:    if $\Delta r_{ij}(t_0) < \Delta r_{ij_1} || \Delta r_{ij}(t_0) > \Delta r_{ij_{\zeta 1}} || v_i(t_0) < v_{i_1} || v_i(t_0) >$
        $v_{i_{\zeta 2}} || v_j\left(t_0 - \tau_{ij}(t_0)\right) < v_{j_1} || v_j\left(t_0 - \tau_{ij}(t_0)\right) > v_{j_{\zeta 3}}$ then
03:        return $\gamma = NaN,\ k = NaN$
04:    else
05:        find $\left(\Delta r_{ij_{\xi 1}} \middle| \min \left|\Delta r_{ij}(t_0) - \Delta r_{ij_{\xi 1}}\right|\right)$,
             $\left(v_{i_{\xi 2}} \middle| \min \left|v_i(t_0) - v_{i_{\xi 2}}\right|\right)$,
             $\left(v_{j_{\xi 3}} \middle| \min \left|v_j\left(t_0 - \tau_{ij}(t_0)\right) - v_{j_{\xi 3}}\right|\right)$
06:        return $\gamma = \gamma(\Delta r_{ij_{\xi 1}}, v_{i_{\xi 2}}, v_{j_{\xi 3}}), k = k\left(\Delta r_{ij_{\xi 1}}, v_{i_{\xi 2}}, v_{j_{\xi 3}}\right)$
07:    end if
08: end for
```

ABMS Using Unity3D Game Engine

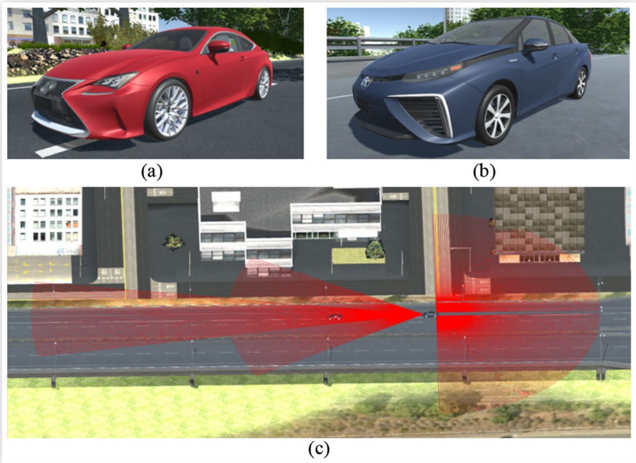
Agent-Based Modeling of CAVs

A vehicle is modeled as a game object in Unity3D, which always includes a rigid body and its associated colliders. By applying forces or torque to the vehicle's rigid body, the vehicle will start to move. Forces such as gravity, friction, drag, and angular drag also have effects on the movement of a vehicle. Compared to real-world environment, time in Unity3D is discrete (default simulation time step is 0.02 s), so the accumulative force acts on the vehicle's rigid body at the start of each simulation time step and resets to zero before the start of the next simulation time step.

Colliders are components defined by Unity3D to simulate physical collisions between two rigid bodies. Since colliders are the major physical parts of game objects, they also define the shapes and sizes of game objects in the simulation. Wheel colliders, specifically, are colliders of game objects that interact with the simulation environment in Unity3D. In our cooperative on-ramp merging case study, we adopt two different vehicle models with realistic dimensions in the real world, which are shown in Figure 4(a) and (b).

In order to enable vehicles with CAV technology in Unity3D, scripts with cooperation protocol are attached to enable their connectivity, and sensors are integrated to enable their autonomy. Specifically, a script written in Unity3D's C#-based Mono Scripting application programming interface (API) is attached to all merging vehicles, which allows those CAVs to retrieve information from the infrastructure and other vehicles through V2X communications. This script also controls CAV's longitudinal movements by the proposed online feedforward/feedback longitudinal control algorithm. Since we mainly focus on the longitudinal control of CAVs in this

FIGURE 4 Vehicle models (a, b) and radar illustration (c) built in Unity3D.



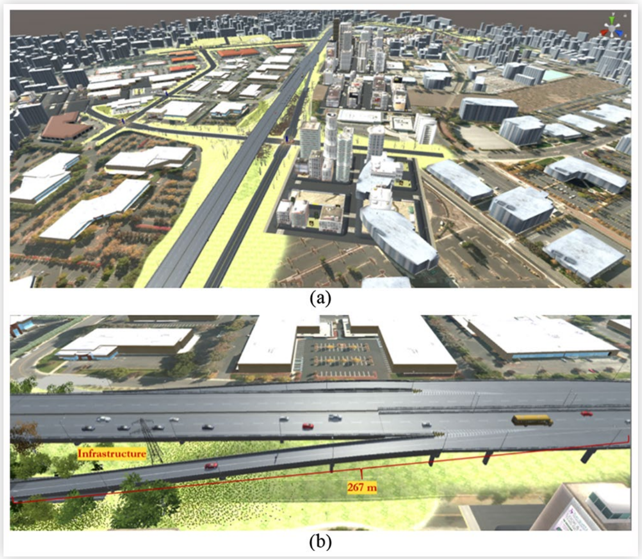
case study, the Simple Waypoint System is adopted, where a vehicle can track the preset trajectory with a user-defined longitudinal speed [35]. Additionally, four radar sensors, including long-/short-range front radars and left/right blind spot radars, are equipped on each CAV. As shown in Figure 4(c), the long-range front radar has a relative narrow angle and long detection distance, so it is appropriate to be considered the primary front-sensing approach. In order to prevent any rear-end collisions, the long-range front radar and the short-range front radar (redundant sensor) continuously check the distance to the physical preceding vehicle on the same lane.

Agent-Based Modeling of Infrastructure

In order to conduct ABMS for CAVs, we build a simulation environment in Unity3D partially based on the city of Mountain View, California as shown in Figure 5(a). California State Route 237 (SR 237) is the major corridor in this environment, with several on-ramps and off-ramps connecting it with urban arterial roads. In this work, we conduct the case study based on the on-ramp which connects E Middlefield Rd and SR 237 westbound. As shown in Figure 5(b), this on-ramp has a length of 267 meter, with a roadside unit-equipped infrastructure positioned between the on-ramp and main line. There is also an elevation difference between the on-ramp and main line, which means the vision of the on-ramp vehicle's driver is obstructed for a long period before merging.

In this simulation environment, a game object associated with a sphere collider is used to simulate the V2I-enabled infrastructure. Essentially, the center of the sphere collider is positioned at this game object (which is represented by a power tower in Figure 5b), and the sphere collider's radius can be set as the V2I communication range. The "isTrigger" function of

FIGURE 5 Game environment of Mountain View, California (a) and infrastructure (b) built in Unity3D.



the sphere collider is enabled to prevent any collisions with incoming vehicles, otherwise vehicles cannot enter the volume of this collider. After setting this sphere collider to "isTrigger," when a vehicle enters and exits its volume, it sends "OnTriggerEnter" and "OnTriggerExit" messages. Then we can attach a configuration script to this game object to call these functions and integrate the aforementioned vehicle sequencing protocol into the script. Namely, when a vehicle enters the radius of this sphere collider (i.e., enters the V2I communication range of the infrastructure), the "OnTriggerEnter" function is called, and the infrastructure retrieve information from the vehicle through "GetComponent" function. The infrastructure then processes this information along with other information retrieved from all other entering vehicles during a certain time window and sends the sequence identification number back to each vehicle at the next time step of sorting process. Once a vehicle exits the radius of this sphere collider, the "OnTriggerExit" function is called, and the infrastructure clear all stored information of this vehicle.

The information sent from a vehicle to the infrastructure includes its longitudinal speed, acceleration, and the global position. Since the distance to the merging point of a vehicle is needed to calculate its estimated arrival time, we also come up with a map-matching system to convert the global position of a vehicle into its distance to the merging point. Since the road segment is neither straight nor flat, we cannot simply calculate the distance between the vehicle's position and the merging point's position. Instead, we build paths with multiple waypoints along the lanes of both main line and on-ramp. Whenever the infrastructure gets the global position of a vehicle, it firstly compares the position with all waypoints' positions on that path to figure out which path segment this vehicle is currently on and what the next waypoint is. Once finished, the distance to the merging point of this vehicle is the sum of its distance to the next waypoint and the path length from the next waypoint to the merging point.

It should be noted that building such a CAV simulation environment that conforms to various test criteria requires huge efforts from developers. Test criteria are oftentimes decided by different OEMs, CAV features to be tested and the level of confidence obtained from previous tests. In order to construct a similar realistic test environment in the virtual simulation in an automatic manner, one needs to come up with some test case generation protocol, which is discussed in the reference [26].

Agent-Based Simulation and Results

Cooperative Merging Simulation with Proposed Protocols

In this work, we study the case where one on-ramp vehicle tries to merge with a six-vehicle string traveling in the main line. The proposed cooperative on-ramp merging protocol is applied to all these seven vehicles. There are other vehicles on the on-ramp and in the main line, but they are all conventional vehicles and are running with a cruise speed of 20 m/s in their own lanes, so they will not affect the cooperative merging process. Hence, they are not considered when we analyze simulation results. The on-ramp vehicle is discharged from the starting point of the on-ramp at an initial speed of 5 m/s. The mainline vehicles are discharged from the upstream of the main line (which is outside the V2I communication range of the infrastructure) with random initial speeds and longitudinal positions. Before they reach the V2I communication starting point, they are driven in a vehicle string at a desired speed of 20 m/s and with a desired time gap of 0.5 s. Upon the mainline vehicles arriving at the V2I communication starting point, the vehicle string has already been formed with a stable state.

Figure 6 illustrates the cooperative on-ramp merging process in this simulation. Figure 6a shows the state when the mainline vehicles have already formed the vehicle string, but have not entered the V2I communication range yet. Figure 6b shows that the on-ramp vehicle and all mainline vehicles are already inside of the V2I communication range, and a sequence identification number of “2” is assigned to the on-ramp vehicle. Therefore, the mainline vehicle with a sequence identification number of “3” is decelerating to create a gap for the on-ramp vehicle to merge. When the on-ramp vehicle is about to merge, as shown in Figure 6c, the gap has already been created and no further longitudinal speed adjustments are needed.

Human-in-the-Loop Simulation with Driver Simulator Platform

After running the simulation with all the vehicles controlled by the proposed online feedforward/feedback longitudinal controller, we also conduct human-in-the-loop simulations where the merging vehicle is controlled by a human driver on a driving simulator as shown in Figure 7. In this scenario, the on-ramp vehicle is a conventional vehicle with no connectivity and autonomy, while all six mainline vehicles are still CAVs. A mainline vehicle can sense the on-ramp vehicle by its long-range front radar once the on-ramp vehicle cuts in front.

FIGURE 6 Cooperative on-ramp merging process.

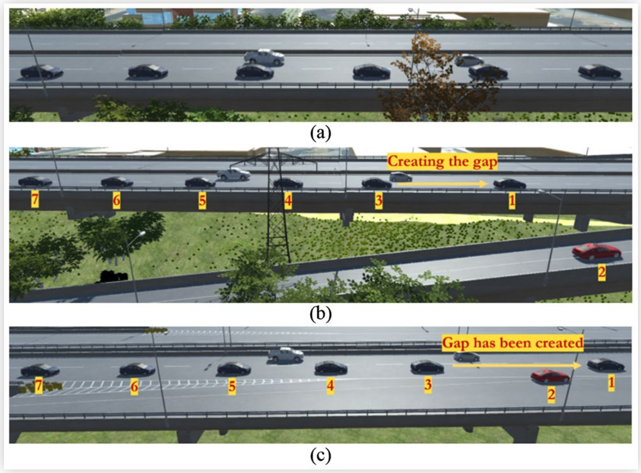
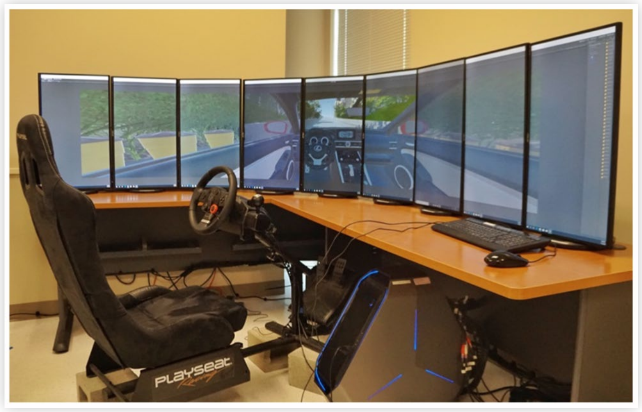


FIGURE 7 Driving simulator platform.



Given the fact that Unity3D allows users to change user input in its graphical user interface (GUI) very easily, the Logitech driving simulator (or potentially any other plug-in-and-play driving simulators) can be connected to Unity3D by simply plugging in the USB cable. A car user control script and a car controller script work together to allow a human driver to control the longitudinal and lateral movements of the vehicle on the driving simulator platform. The car user control script receives horizontal input from the driving simulator’s steering wheel and vertical input from the driving simulator’s throttle and brake pedals, then it calls the move function of the car controller script with the horizontal and vertical inputs. Wheel colliders of the vehicle will then be controlled based on these inputs, and the vehicle can be set in motion. In order to reduce any system biases on the results of human-in-the-loop simulation, we recruit four different drivers to drive the on-ramp vehicle, each for five times. The drivers drive the vehicle on the driving simulator based on their own preferences, namely, there is no requirement for them to drive aggressively or cautiously. We categorize their driving behaviors only after the speed trajectories are generated by their simulation runs. It should be also noted that a scene view of the game is displayed on the left-hand side of the driver’s view, which is slightly different with the view shown in Figure 7. The scene view works as the rearview mirrors to allow the driver of the vehicle to observe its surrounding traffic.

Simulation Results of Cooperative Merging Simulation and Human-in-the-Loop Simulation

The vehicle speed profiles and distances to the merging point from the cooperative merging scenario and the human-in-the-loop scenario are compared in Figures 8 and 9, respectively. Both cooperative merging simulation and human-in-the-loop simulation are run with a frequency of 50 FPS. It should be noted that, we only show five of the twenty simulation runs from the human-in-the-loop scenario and categorize them into “cautious driver” and “aggressive driver”

FIGURE 8 Speed profiles of vehicles driven in different scenarios.

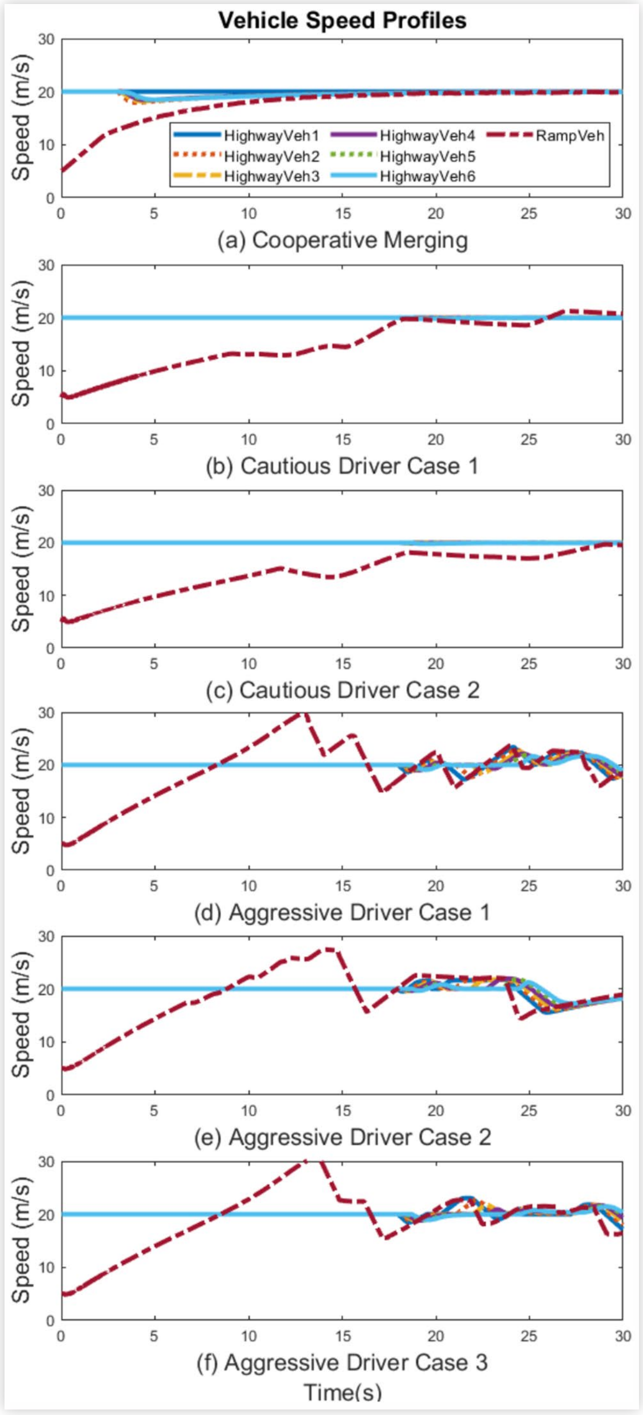


FIGURE 9 Distance profiles of vehicles driven in different scenarios.

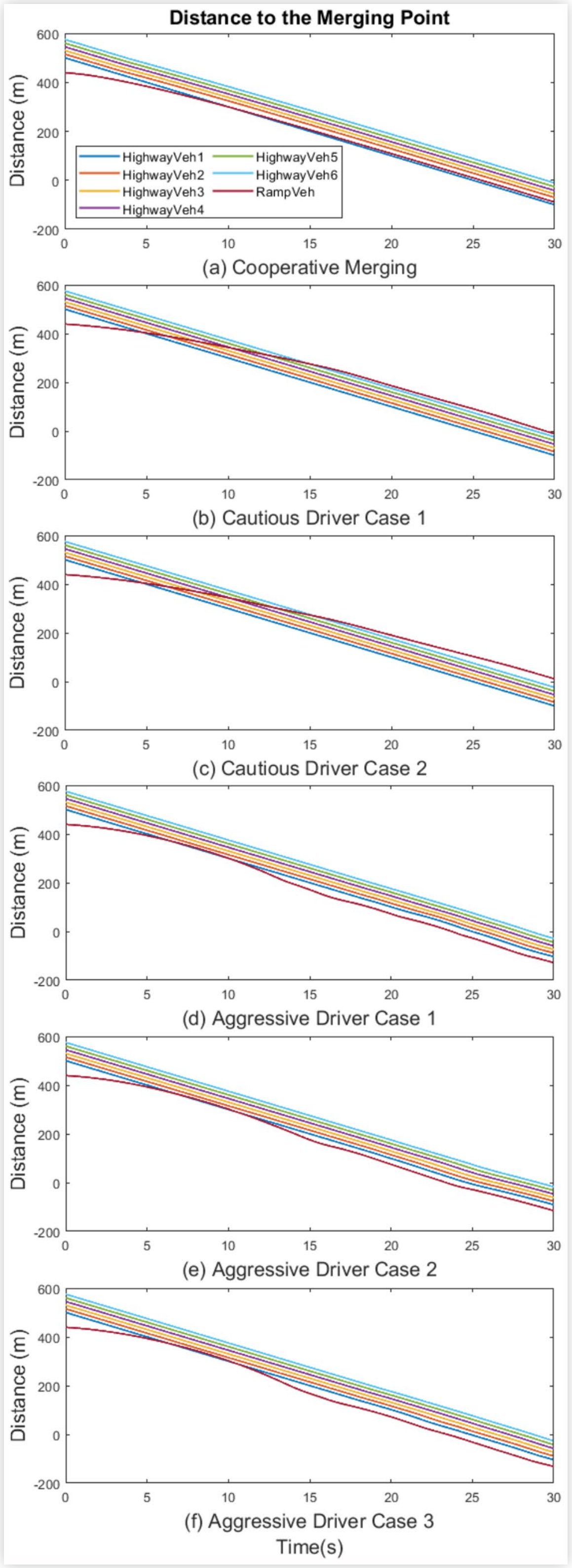


TABLE 1 Comparison results between cooperative merging and baseline.

	Travel time (s)	Energy (KJ)	HC (g)	CO (g)	CO_2 (g)	NOx (g)
Cooperative merging	218.14	9154.0	0.0094	1.1737	651.29	0.0440
Human-in-the-loop	233.58	9930.6	0.0200	2.8192	706.54	0.0759
Reduction percentage	6.6%	7.8%	53.0%	58.4%	7.8%	42.0%

cases based on the average changing rate of vehicle speed, but all twenty simulation runs are included when calculating the results shown in Table 1.

Also note that in the cooperative merging simulation, ramp vehicle merges to the second place of the vehicle string. However, in the human-in-the-loop simulation, there is no requirement for the human driver to follow. All he/she needs to do while conducting the human-in-the-loop simulation is to drive the ramp vehicle from on-ramp to the main line, but the ramp vehicle can either merge into the six-vehicle string, or attach to the front/end of the vehicle string.

Simulation results are shown in Figures 8 and 9, where Figure 8 shows the longitudinal speed profiles of CAVs, while Figure 9 shows the longitudinal position of CAVs, respectively, for different simulation cases. Note that the longitudinal positions of CAVs on different lanes (main line and on-ramp) are calculated by their distances to the merging point, which is also the way we apply our online feedforward/feedback longitudinal controller.

Figure 8a shows the speed profiles generated from the proposed cooperative merging scenario, where mainline Vehicle 2 reaches the V2I communication starting point at around 3 s, and is assigned a sequence identification number of 3, which means it needs to follow the movement of the on-ramp vehicle. When Vehicle 2 starts to decelerate to adjust its speed and longitudinal position with respect to the on-ramp vehicle, its followers (mainline Vehicles 3, 4, 5, and 6) also decelerate accordingly since they are still in a vehicle string. Meanwhile, the on-ramp vehicle gradually adjusts its speed and longitudinal position with respect to mainline Vehicle 1. Therefore, when the merging happens at around 18 s, there is no speed change of any vehicle. As shown in Figure 9a, different vehicles' distances to the merging point further explain aforementioned descriptions.

For the cautious driver cases shown in Figure 8b and c, the on-ramp vehicle speeds up with a relatively small acceleration while it is on the on-ramp. Upon approaching the merging point, the on-ramp vehicle is already behind all mainline vehicles. So, it attaches to the end of the mainline vehicle string, and there is no speed change of any mainline vehicle. The distance to the merging point plots shown in Figure 9b and c are also straightforward.

For the aggressive driver cases shown in Figure 8d, e, and f, the on-ramp vehicle speeds up with a relatively large acceleration at first since the driver's line of sight is obstructed and the driver does not know the traffic condition in the main line at that time. Once the driver observes the traffic condition at around 14 s, the vehicle begins to slow down to avoid rear-end collision with the downstream traffic (which all travel at a speed of 20 m/s). Although the aggressive drivers already decide to overpass the whole six-vehicle string, they also need to adjust their speed to merge into the gap between the string leader and its downstream vehicle (but the time gap is larger than 0.5 s so it is possible to merge in). Upon attaching to the front of the mainline vehicle string, all the six followers in that string change speed accordingly to prevent any rear-end collisions, and there are also some speed fluctuations afterwards due to the relatively poor speed control of the aggressive drivers.

We also compare the system performance of the cooperative merging scenario with that of the human-in-the-loop scenario, in terms of travel time, energy consumption, and pollutant emissions. The number shown in each cell of Table 1 is the sum of all seven vehicles' results over the same traveling distance. The results for the human-in-the-loop scenario are the average of all twenty simulation runs. Specifically, travel time is used to represent the mobility benefit of the proposed cooperative on-ramp merging protocol, and it is calculated as the average time spent by all seven vehicles to travel the same distance. It is shown in Table 1 that a reduction in travel time of 6.6% can be achieved by applying the protocol. Energy consumption and pollutant emissions are calculated by the US Environmental Protection Agency's MOtor Vehicle Emission Simulator (MOVES) [36]. Compared to the human-in-the-loop scenario, the cooperative merging protocol provides 7.8% savings on energy consumption and up to 58.4% reduction on pollutant emissions, respectively.

Conclusions and Future Work

In this work, an online feedforward/feedback longitudinal control algorithm was developed for CAVs. The game engine Unity3D was used to create ABMS of CAVs due to its visualization capability and many other advantages. Agent-based models of CAVs and infrastructures were built in Unity3D with colliders and C#-based scripting API for a roadway environment in Mountain View, California. The models were used to simulate a situation where an on-ramp vehicle merges into a string of CAVs in the main line. A comparison between the cooperative merging scenario and the human-in-the-loop scenario was made, analyzing the benefits of the proposed cooperative merging protocol in terms of travel time, energy consumption, and pollutant emissions. It is shown that the proposed cooperative merging protocol can result in 7% travel time reduction, 8% energy savings, and up to 58% pollutant emission reduction when compared to human driving.

Future work includes developing multi-fidelity CAV models with higher flexibility and quality, where a mixed traffic environment (penetration rate of CAVs is not 100%) can be evaluated. Additional case studies of CAV technology can be conducted within this ABMS platform, such as cooperative adaptive cruise control (CACC) and eco-driving at signalized intersections. Integrating Unity3D with other software platforms via User Datagram Protocol (UDP) socket communication is also another possible research direction.

Contact Information

Ziran Wang
Ph.D. Candidate
Mechanical Engineering
University of California, Riverside
zwang050@ucr.edu
www.engr.ucr.edu/~zwang

Definitions/Abbreviations

ABMS	- agent-based modeling and simulation
ADAS	- advanced driver-assistance systems
API	- application programming interface
AV	- automated vehicle
CACC	- cooperative adaptive cruise control
CAV	- connected and automated vehicle
CV	- connected vehicle
GUI	- graphical user interface
IMU	- inertial measurement unit
MOVES	- MOtor Vehicle Emissions Simulator
MPC	- model predictive control
UDP	- User Datagram Protocol
V2I	- vehicle-to-infrastructure
V2V	- vehicle-to-vehicle
V2X	- vehicle-to-everything

References

1. Jia, D., Lu, K., Wang, J., Zhang, X., and Shen, X., "A Survey on Platoon-Based Vehicular Cyber-Physical System," *IEEE Communications Surveys and Tutorials* 18(1):263-284, 2015.
2. United States Department of Transportation, "Traffic Safety Facts-Research Note," https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812318, accessed May 6, 2019.
3. INRIX, "Los Angeles Tops INRIX Global Congestion Ranking," http://inrix.com/press-releases/scorecard-2017/, accessed May 6, 2019.
4. Connected and Automated Vehicles, http://autocaat.org/Technologies/Automated_and_Connected_Vehicles/, accessed May 6, 2019.
5. Rios-Torres, J. and Malikopoulos, A., "A Survey on the Coordination of Connected and Automated Vehicles at Intersections and Merging at Highway On-Ramps," *IEEE Transactions on Intelligent Transportation Systems* 18(5):1066-1077, 2017.
6. Rios-Torres, J. and Malikopoulos, A., "Automated and Cooperative Vehicle Merging at Highway On-Ramps," *IEEE Transactions on Intelligent Transportation Systems* 18(4):780-789, 2017.
7. Awal, T., Kulik, L., and Ramamohanrao, K., "Optimal Traffic Merging Strategy for Communication- and Sensor-Enabled Vehicles," in *Proceedings of 16th International IEEE Annual Conference on Intelligent Transportation Systems*, Hague, Netherlands, 2013, 1468-1474.
8. Raravi, G., Shingde, V., Ramamritham, K., and Bharadia, J., "Merge Algorithms for Intelligent Vehicles," *Next Generation Design Verification Methodology Distribution Embedded Control Systems*, (Springer, 2007), 51-65.
9. Cao, W., Mukai, M., Kawabe, T., Nishira, H., and Fujiki, N., "Cooperative Vehicle Path Generation During Merging Using Model Predictive Control with Real-Time Optimization," *Control Engineering Practice* 34:98-105, 2015.
10. Milanés, V., Godoy, J., Villagrá, J., and Pérez, J., "Automated On-Ramp Merging System for Congested Traffic Situations," *Transactions on Intelligent Transportation Systems* 12(2):500-508, 2011.
11. Marinescu, D., Čurn, J., Bouroche, M., and Cahill, V., "On-Ramp Traffic Merging Using Cooperative Intelligent Vehicles: A Slot-Based Approach," in *Proceedings of 15th International IEEE Annual Conference on Intelligent Transportation Systems*, Anchorage, Alaska, 2012.
12. Uno, A., Sakaguchi, T., and Tsugawa, S., "A Merging Control Algorithm Based on Inter-Vehicle Communication," in *Proceedings of International IEEE Annual Conference on Intelligent Transportation Systems*, Tokyo, Japan, 1999, 783-787.
13. Lu, X.-Y. and Hedrick, J.K., "Longitudinal Control Algorithm for Automated Vehicle Merging," in *Proceedings of IEEE 39thConference on Decision Control*, Sydney, Australia, 2000, 450-455.
14. Lu, X.-Y., Tan, H.-S., Shladover, S., and Hedrick, J., "Automated Vehicle Merging Maneuver Implementation for AHS," *Vehicle System Dynamics* 41(2):85-107, 2004.
15. Gustafsson, L. and Sternad, M., "Consistent Micro, Macro, and State-Based Population Modelling," *Mathematical Bioscience* 225(2):94-107, 2010.
16. Macal, C.M. and North, M.J., "Tutorial on Agent-Based Modeling and Simulation," in *Proceedings of IEEE Winter Simulation Conference*, Orlando, FL, 2005.
17. Boesch, P.M. and Ciari, F., "Agent-Based Simulation of Autonomous Cars," in *Proceedings of American Control Conference*, Chicago, IL, 2015, 2588-2592.
18. Sumo - Simulation of Urban Mobility, https://sumo.dlr.de/index.html, accessed May 6, 2019.

19. PTV Vissim, http://vision-traffic.ptvgroup.com/en-us/products/ptv-vissim/, accessed May 6, 2019.
20. Aimsun, https://www.aimsun.com/, accessed May 6, 2019.
21. Unity3D, https://Unity3d.com/, accessed May 6, 2019.
22. Craighead, J., Burke, J., and Murphy, R., “Using the Unity3D Game Engine to Develop SARGE: A Case Study,” in *Proceedings of the 2008 Simulation Workshop at the International Conference on Intelligent Robots and Systems (IROS 2008)*, Nice, France, 2008.
23. Estreen, T. and Nord, S., “Visualization of Platooning in Unity3D,” Degree Projects in Technology, 2018.
24. Khosravi, M. and Yassiry, A., “Virtual Truck Platooning Implementation in Unity3D,” Degree Projects in Technology, 2018.
25. Yamaura, M., Arechiga, N., Shiraishi, S., Eisele, S. et al., “ADAS Virtual Prototyping Using Modelica and Unity3D Co-Simulation via OpenMETA,” in *Proceedings of Japanese Modelica Conference*, Tokyo, Japan, 2016, No. 124, 43-49.
26. Kim, B., Kashiba, Y., Dai, S., and Shiraishi, S., “Testing Autonomous Vehicle Software in the Virtual Prototyping Environment,” *IEEE Embedded Systems Letters* 9(1):5-8, 2017.
27. Dai, S., Hite, J., Masuda, T., Kashiba, Y., Arechiga, N., Shiraishi, S., Eisele, S., Scott, J., and Bapty, T., “Control Parameter Optimization for Autonomous Vehicle Software Using Virtual Prototyping,” in *Proceedings of IEEE 28th International Symposium on Software Reliability Engineering*, Toulouse, France, 2017.
28. Wang, Z., Wu, G., and Barth, M., “Distributed Consensus-Based Cooperative Highway On-Ramp Merging Using V2X Communications,” SAE Technical Paper 2018-01-1177, 2018, doi:10.4271/2018-01-1177.
29. Zhang, Y., Kosmatopoulos, E.B., Inannou, P.A., and Chien, C.C., “Autonomous Intelligent Cruise Control Using Front and Back Information for Tight Vehicle Following Maneuvers,” *IEEE Transactions on Vehicular Technology* 48(1):319-328, 1999.
30. Xiao, L. and Gao, F., “Practical String Stability of Platoon of Adaptive Cruise Control Vehicles,” *IEEE Transactions on Intelligent Transportation Systems* 12(4):1184-1194, 2011.
31. van Arem, B., van Driel, C., and Visser, R., “The Impact of Cooperative Adaptive Cruise Control on Traffic-Flow Characteristics,” *IEEE Transactions on Intelligent Transportation Systems* 7(4):429-436, 2006.
32. Wang, Z., Wu, G., and Barth, M., “Developing a Distributed Consensus-Based Cooperative Adaptive Cruise Control System for Heterogeneous Vehicles with Predecessor Following Topology,” *Journal of Advanced Transportation*, 2017, 2017.
33. Jia, D. and Ngoduy, D., “Platoon Based Cooperative Driver Model with Consideration of Realistic Inter-Vehicle Communication,” *Transportation Research Part C* 68:245-264, 2016.
34. Bernardo, M., Salvi, A., and Santini, S., “Distributed Consensus Strategy for Platooning of Vehicles in the Presence of Time-Varying Heterogeneous Communication Delays,” *IEEE Transactions on Intelligent Transportation Systems* 16(1):102-112, 2015.
35. Unity3D Asset Store, “Simple Waypoint System,” https://assetstore.Unity3D.com/packages/tools/animation/simple-waypoint-system-2506, accessed May 6, 2019.
36. “MOtor Vehicle Emission Simulator (MOVES),” https://www.epa.gov/moves, accessed May 6, 2019.