Received: 3 December 2021 | Revised: 18 February 2022 | Accepted: 7 March 2022

DOI: 10.1002/adc2.101

ORIGINAL ARTICLE

WILEY

Improving autonomous vehicle in-traffic safety using learning-based action governor

Kyoungseok Han[1] | **Nan Li**[2] | **Eric Tseng**[3] | **Dimitar Filev**[3] | **Ilya Kolmanovsky**[2] | **Anouck Girard**[2]

[1]School of Mechanical Engineering, Kyungpook National University, Daegu, South Korea

[2]Department of Aerospace Engineering, University of Michigan, Ann Arbor, Michigan, USA

[3]Ford Motor Company, Dearborn, Michigan, USA

Correspondence
Kyoungseok Han, School of Mechanical Engineering, Kyungpook National University, 80 Daehak-ro, Buk-gu, Daegu 41566, South Korea.
Email: kyoungsh@knu.ac.kr

Abstract

The *Action Governor (AG)* is a supervisory scheme augmenting a nominal control system in order to enhance the system's safety and performance. It acts as an action filter, monitoring the action commands generated by the nominal control policy and adjusting the ones that might lead to undesirable system behavior. In this article, we present an approach based on learning to developing an AG for autonomous vehicle (AV) decision policies to improve their safety for operating in mixed-autonomy traffic (i.e., traffic involving both AVs and human-operated vehicles (HVs)). To achieve this, we demonstrate that it is possible to train the AG in a traffic simulator that is capable of representing in-traffic interactions among AVs and HVs. We illustrate the effectiveness of this learning-based AG approach to improving AV in-traffic safety through simulation case studies.

KEYWORDS

action governor, autonomous vehicle, learning-based control, reinforcement learning

1 | INTRODUCTION

With the rapid technological advances in sensing,[1] perception,[2] and computations,[3] autonomous driving is becoming increasingly feasible.[4-6] Planning and control for autonomous vehicles (AVs) to operate safely and effectively in a range of traffic scenarios,[7-9] including on roads shared with human-operated vehicles,[10,11] have been the subject of much research over the last decade.

Conventionally, rule-based design of control policies is often adopted in practice,[12-15] where AV behaviors in various situations are defined by (typically hand-crafted) explicit rules. However, due to the significant complexity and variety of real-world traffic scenarios, it is difficult to determine a complete set of rules that achieves desirable performance in all scenarios. Therefore, there has been in both academia and industry a rapidly growing interest in data-driven/learning-based approaches to the design of AV control policies. One strategy along these lines is to learn a policy from traffic data that imitate expert (e.g., human driver) behavior in the data.[16-18] The implementation of such an approach relies on the availability of sufficient quantity of data. Another popular strategy is based on reinforcement learning (RL), which trains an optimal AV control policy with respect to a reward function through an exploration-and-exploitation process.[19-24] However, a significant issue with these learning-based approaches that hinders their practical applications is that the control policies obtained by these approaches typically do not have safety guarantees, that is, may lead to unsafe AV behavior (such as vehicle collision) under certain circumstances.

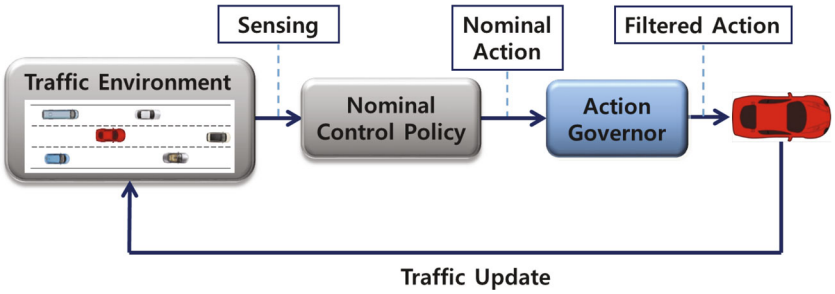

FIGURE 1 Architecture of autonomous driving system augmented with the action governor

One approach to addressing this issue is by augmenting the nominal policy (e.g., the RL policy) with a safety supervisor that monitors the action commands produced by the nominal policy and adjusts/corrects the ones that may lead to unsafe behavior. This approach has been pursued, for instance, in References 21,25-30. When an explicit model representing the inter-vehicle dynamics of a specific traffic scenario (e.g., car-following or lane-changing) is available, such a safety supervisor may be designed using a model-based approach, as done in References 27-30. However, for more complex scenarios including those involving multiple vehicle interactions, an explicit model is typically not available. Therefore, an emerging approach is to train a safety supervisor through learning, as investigated in References 26,31-33. Note that the learning objectives of those aforementioned learning-based approaches to control policy design and a learning-based approach to safety supervisor design are different: The former aims at developing a nominal control policy that achieves optimal performance with regard to (the expected value of) a reward function, while the latter focuses on the safety aspect and typically pursues minimum modification to the nominal control.

This article introduces such a learning-based approach to designing a safety supervisor for improving AV in-traffic safety based on the *Action Governor* (AG) framework. The AG is an add-on scheme augmenting a nominal control loop to enhance the system's safety and performance (see Figure 1). It monitors the nominal action commands and minimally adjusts the ones that may lead to undesirable system behavior. The AG for discrete-time linear systems (with known models) subject to nonconvex, exclusion-zone avoidance-type constraints was studied in our previous work.[34] It was extended to discrete-time piecewise-affine systems with additive set-bounded disturbances in our work.[35] Using a model-based AG to realize safe RL (i.e., ensuring constraint satisfaction during the RL exploration-and-exploitation process) was discussed in our recent work.[30] Meanwhile, developing an AG using a learning-based approach has not been addressed before.

The AG and the control barrier function (CBF)[36-38] have similar goals—they are both intended to monitor and adjust nominal control signals in order to enforce constraints. Meanwhile, the AG and the CBF have the following several distinctions: (1) Unlike CBFs which are typically premised on continuous-time system models and infinite sampling frequencies, the AG is a discrete-time scheme and easier to implement with a digital microcontroller. (2) A CBF adjusts the control signal according to a sufficient condition for safety. Specifically, the sublevel set of the barrier function used to guard against constraint violations is only a subset of all safe states. In contrast, an AG uses a sufficient and necessary condition to characterize the safety of actions. Specifically, for linear and piecewise-affine systems, it is shown that the set of states the AG allows the system to reach without constraint violations is maximal (in terms of containing all safe states).[34,35] Such an "if-and-only-if" safety characterization on the one hand allows for a greater control action flexibility, which can translate into improved control performance, and on the other hand facilitates learning, which will be discussed in detail in this article.

In this context, the contributions of this article include the following:

1. We propose a learning-based approach to designing an AG as a safety supervisor for a nominal AV control policy that monitors the action commands generated by this nominal policy and minimally adjusts the unsafe ones to improve the AV in-traffic safety. Such a learning-based approach to AG design is new.
2. The learning approach is based on identification and recording of "unviable" state and action pairs (i.e., pairs that lead to inevitable future safety violations) rather than only the ones that cause immediate safety violations. Note that a safety supervisor based on identification of only the latter ones is easier to design but cannot avoid all safety violations due to the dynamic nature of traffic situations.

3. We illustrate the proposed learning-based AG approach in a simulation-based case study representing AV highway driving. The training and testing of the AG is in a traffic simulator developed in our previous work[39] where the interactive behavior of vehicles on the highway is represented through a game-theoretic approach. We show that the number of vehicle safety constraint violations can be reduced moderately to considerably (depending on the nominal AV policies) when we augment the nominal policy with the trained AG.

The rest of this article is structured as follows. In Section 2, we first review the principal mechanism of the AG scheme and then present our learning algorithm for learning-based AG design. We describe the setup of our simulation-based case study representing AV highway driving in Section 3 and present and discuss the results in Section 4. Finally, we conclude the article in Section 5.

2 | ACTION GOVERNOR

In this section, we first introduce the principal mechanism of the AG approach, and then present a learning-based approach to AG development.

2.1 | Action governor mechanism

The AG is an add-on scheme for a nominal control system to enhance the system's safety and performance. It is placed between the system's nominal control policy and plant (see Figure 1) and acts as an action filter, monitoring the control actions generated by the nominal policy and adjusting the ones that might lead to undesirable system behavior.[34] Specifically, it adjusts the actions according to the following optimization problem solved online at each sample time instant $k \in \mathbb{N}_0$:

$$u(k) \in \arg\min_{u} \ \textit{Metric}\ (u_{\text{nom}}(k), u \mid x(k)), \tag{1a}$$

$$\text{s.t.} \quad u \in \mathcal{U}_{\text{safe}}(x(k)) \subseteq \mathcal{U}, \tag{1b}$$

where $x(k)$ designates the state of the system at time k, $u_{\text{nom}}(k)$ designates the control action generated by the nominal policy, $u(k)$ designates the control action after the AG modification. The function $\textit{Metric}\,(\cdot, \cdot \mid x(k))$ in (1a) gives a distance between the nominal action $u_{\text{nom}}(k)$ and each action u of the action space $\mathcal{U}$. Therefore, the minimization in (1a) aims to reduce the difference of the modified action $u(k)$ from the nominal action $u_{\text{nom}}(k)$. In general, this *Metric* function can be state-dependent, meaning that for different states $x(k)$, which $u \in \mathcal{U}$ is considered closer to the same nominal action $u_{\text{nom}}(k)$ can be different. Most importantly, the AG uses a set, $\mathcal{U}_{\text{safe}}(x(k))$, to characterize the "safe actions" (i.e., the actions that will not lead to undesirable system behavior) when the system state is at $x(k)$ and restrict the selection of modified action $u(k)$ to this set in (1b).

A common approach to characterizing (un)desirable system behavior is through imposing safety-related requirements as pointwise-in-time constraints on the system state in the form of References 34 and 40

$$x(k) \notin \mathcal{X}_0, \qquad \forall k \in \mathbb{N}_0, \tag{2}$$

where $\mathcal{X}_0$ represents a set of undesirable system states (called the "avoid-set"). For instance, for an AV operating in traffic, $\mathcal{X}_0$ may represent the set of traffic states that correspond to collision events of the AV with another vehicle.

For systems with continuous state and action spaces, with state dynamics that can be expressed as $x(k+1) = f(x(k), u(k))$, and with safety requirements represented as (2), the following specific formulation of the AG is proposed in References 34 and 35,

$$u(k) \in \arg\min_{u \in \mathcal{U}} \ \|u_{\text{nom}}(k) - u\|_S^2, \tag{3a}$$

$$\text{s.t.} \quad f(x(k), u) \in \mathcal{X}_{\text{safe}}, \tag{3b}$$

where $\mathcal{X}_{\text{safe}}$ represents a "safe set" of states, and $\|\cdot\|_S^2 = (\cdot)^\top S(\cdot)$ with S being a positive-definite weighting matrix. Note that (3a) is a specific formulation of (1) because the minimization in (3aa) is equivalent to (1a) with *Metric* $(u_{\text{nom}}(k), u \mid x(k)) = \|u_{\text{nom}}(k) - u\|_S$ and the constraint in (3ab) can be equivalently written as

$$u \in \mathcal{U}_{\text{safe}}(x(k)) = f^{-1}(\mathcal{X}_{\text{safe}})|_{x(k)}, \tag{4}$$

where $\mathcal{U}_{\text{safe}}(x(k)) = f^{-1}(\mathcal{X}_{\text{safe}})|_{x(k)}$ is the cross-section of the preimage of $\mathcal{X}_{\text{safe}}$ under f at $x = x(k)$.

It can be seen from (1)–(4) that the key component of the AG to avoid undesirable/unsafe system behavior is the state-dependent set of safe actions, $\mathcal{U}_{\text{safe}}(x)$. For safety requirements represented as (2), this set is characterized as follows,

$$\begin{aligned}\mathcal{U}_{\text{safe}}(x) = \{u \in \mathcal{U} \,:\, &\text{If } x(0) = x, u(0) = u, \text{then } x(1) \notin \mathcal{X}_0 \\ &\text{and } \exists u(\tau) \in \mathcal{U} \text{ such that } x(\tau+1) \notin \mathcal{X}_0 \text{ for all } \tau = 1, 2, \dots\},\end{aligned} \tag{5}$$

where the first line, $x(1) \notin \mathcal{X}_0$, represents the immediate safety of the system under any action $u \in \mathcal{U}_{\text{safe}}(x)$, and the second line, $\exists u(\tau) \in \mathcal{U}$ such that $x(\tau+1) \notin \mathcal{X}_0$, represents the feasibility to achieve future safety (also called "viability").

For a known and explicit model of state dynamics f, it is possible to derive an explicit expression for $\mathcal{U}_{\text{safe}}(x)$ defined above and compute it accordingly.[34,35] When f is not explicitly known or highly complex, learning-based approaches may be exploited to estimate $\mathcal{U}_{\text{safe}}(x)$. In what follows, we introduce such a learning-based approach, which is specifically designed for training an AG to supervise AV decision policies in order to improve AV in-traffic safety.

2.2 | Learning action governor algorithm

The learning algorithm is based on the observation that the set $\mathcal{U}_{\text{safe}}(x)$ in (5) can be expressed as follows,

$$\mathcal{U}_{\text{safe}}(x) = \bigcap_{k=1}^{\infty} \mathcal{U}_{\text{safe},k}(x) = \lim_{k\to\infty} \mathcal{U}_{\text{safe},k}(x), \tag{6}$$

where $\mathcal{U}_{\text{safe},k}(x)$, $k = 1, 2, \dots$, is a nonincreasing sequence of subsets of $\mathcal{U}$ (and therefore, the set-theoretic limit exists) defined as

$$\begin{aligned}\mathcal{U}_{\text{safe},k}(x) = \{u \in \mathcal{U} \,:\, &\text{If } x(0) = x, u(0) = u, \text{then } x(1) \notin \mathcal{X}_0 \\ &\text{and } \exists u(\tau) \in \mathcal{U} \text{ such that } x(\tau+1) \notin \mathcal{X}_0 \text{ for all } \tau = 1, \dots, k\}.\end{aligned} \tag{7}$$

The set $\mathcal{U}_{\text{safe},k}(x)$ represents the set of actions that achieves immediate safety $x(1) \notin \mathcal{X}_0$ and ensures feasibility to achieve safety $x(\tau+1) \notin \mathcal{X}_0$ over future steps from $\tau = 1$ up to k. When system dynamics and safety specifications have additional properties, it is possible that there exists a finite $k^* \in \mathbb{N}_0$ such that for all $x \notin \mathcal{X}_0$,

$$\mathcal{U}_{\text{safe}}(x) = \lim_{k\to\infty} \mathcal{U}_{\text{safe},k}(x) = \mathcal{U}_{\text{safe},k^*}(x). \tag{8}$$

See, for instance, proposition 5 of Reference 34. In an AV control application, k^* may correspond to a time duration that ensures a full stop of the vehicle with maximum braking for all speeds under the speed limit, because once the vehicle comes to a full stop it can stay at its current state forever. The learning algorithm presented in this section aims to learn the complement of $\mathcal{U}_{\text{safe},k^*}(x)$, that is,

$$\begin{aligned}\mathcal{U}^c_{\text{safe},k^*}(x) = \{u \in \mathcal{U} \,:\, &\text{If } x(0) = x, u(0) = u, \text{then } x(1) \in \mathcal{X}_0 \text{ or every} \\ &u(\tau) \in \mathcal{U} \text{ leads to } x(\tau+1) \in \mathcal{X}_0 \text{ for some } \tau \in \{1, \dots, k^*\}\}.\end{aligned} \tag{9}$$

Moreover, the presented learning algorithm is designed specifically for training an AG to supervise AV decision policies that have a finite number of decisions, that is, $\mathcal{U}$ is a finite set. These decisions may represent motion primitives[41] or behavior-level maneuvers, such as making a constant acceleration over a sampling interval or making a lane change to the left/right and so forth.[39]

Algorithm 1. Learning algorithm

Input: $\mathcal{U}, \mathcal{X}_0, N, M, K, \mathcal{D}$
Output: $\mathcal{D}, \mathcal{D}_N$

$\mathcal{D}_N \leftarrow \emptyset$
for $m = 1 : M$ **do**
  Initialize state $x(0)$;
  Initialize buffers $\mathcal{B}_x \leftarrow \{x(0)\}$ and $\mathcal{B}_u \leftarrow \emptyset$;
  $k \leftarrow 0$;
  while $k<K$ **do**
    $u(k) \in \mathcal{U} \setminus \mathcal{D}(x(k))$;
    $x(k+1) \leftarrow \mathsf{Sim}(x(k), u(k))$;
    if $x(k+1) \in \mathcal{X}_0$ **then**
      $\mathcal{D} \leftarrow \mathcal{D} \cup (x(k), u(k))$;
      $n \leftarrow 0$;
      while $\mathcal{U} \setminus \mathcal{D}(x(k-n)) = \emptyset$ **do**
        $n \leftarrow n+1$;
        if $x(k-n) \in \mathcal{B}_x$ **then**
          $\mathcal{D} \leftarrow \mathcal{D} \cup (x(k-n), u(k-n))$, where $x(k-n)$ and $u(k-n)$ are read from buffers $\mathcal{B}_x$ and $\mathcal{B}_u$;
        else
          $\mathcal{D}_N \leftarrow \mathcal{D}_N \cup \{x(k-n+1)\}$;
          Go to Step 2;
        end if
      end while
      $k \leftarrow k-n$;
      $\mathcal{B}_x \leftarrow \mathcal{B}_x(\{\dots, x(k)\})$ and $\mathcal{B}_u \leftarrow \mathcal{B}_u(\{\dots, u(k-1)\})$;
    else
      $k \leftarrow k+1$;
      $\mathcal{B}_x \leftarrow \{x(k-N), \dots, x(k)\}$ and $\mathcal{B}_u \leftarrow \{u(k-N), \dots, u(k-1)\}$;
    end if
  end while
end for

We now present the learning algorithm as Algorithm 1.

The inputs of Algorithm 1 include an action space $\mathcal{U}$, an avoid-set $\mathcal{X}_0$, a look-back horizon length N, a maximum number of training episodes M, a maximum length of one episode K, and a dataset $\mathcal{D}$ that represents an estimate of $\mathcal{U}^c_{\text{safe},k^*}(x)$ before learning. The avoid-set $\mathcal{X}_0$ does not need to be provided in an explicit form but can be defined through a logic that checks the condition $x \in \mathcal{X}_0$. For instance, $\mathcal{X}_0$ may represent the set of states where the AV collides with another vehicle (in a simulator) or violates safe separation distance constraints (in a simulator or in real-world traffic). In this case, a logic that detects collision or safe distance violation events can be used to define $\mathcal{X}_0$ and as the input of Algorithm 1. The look-back horizon length N represents how far back we trace to determine the first time instant and the state at which a safety violation can no longer be avoided (called an "unviable" state). If such a state is identified, then its previous state and the action that leads to this unviable state are included in the dataset $\mathcal{D}$. On the one hand, an overly small N (e.g., $N \ll k^*$) may cause the learning algorithm to fail to identify many unviable states and lead to an underestimate of $\mathcal{U}_{\text{safe},k^*}(x)$. On the other hand, a larger N leads to larger-sized buffers $\%_x$ and $\%_u$ that are used to temporarily store trajectory data for tracing back, increasing the memory footprint of the learning process. A method to automatically adjust N is presented in Algorithm 2.

The outputs of Algorithm 1 include an updated dataset $\mathcal{D}$ that represents an updated estimate of $\mathcal{U}^c_{\text{safe},k^*}(x)$ after learning and another dataset $\mathcal{D}_N$. The reason for having $\mathcal{D}$ in both the inputs and the outputs is to make Algorithm 1 incremental, that is, one can feed a previously obtained $\mathcal{D}$ to Algorithm 1 as an input to continue learning and improve $\mathcal{D}$. The dataset $\mathcal{D}$ has the following structure,

$$\mathcal{D} = \begin{bmatrix} \cdots & x_{i-1} & x_i & x_{i+1} & \cdots \\ \cdots & u_{i-1} & u_i & u_{i+1} & \cdots \end{bmatrix}, \tag{10}$$

where (x_i, u_i) in each column represents a state and action pair that leads to an unsafe or unviable next state. In Steps 7 and 12 of Algorithm 1, we also use the notation $\mathcal{D}(x)$, which extracts from $\mathcal{D}$ the data of actions for a specific state value x, that is,

$$\mathcal{D}(x) = \begin{bmatrix} \cdots & u_{j-1} & u_j & u_{j+1} & \cdots \end{bmatrix}, \tag{11}$$

with $(x, u_j) \in \mathcal{D}$. In particular, $\mathcal{D}(x)$ represents an estimate of $\mathcal{U}^c_{\text{safe},k^*}(x)$. Note that as only unviable state and action pairs are stored, the memory requirements may not be very large. In lieu of storing data, a functional or cluster-based representation of $\mathcal{U}^c_{\text{safe},k^*}(x)$ could be updated online. However, updating a functional or cluster-based classifier introduces a number of separate issues, and is therefore not pursued in this work but left to future research.

The dataset $\mathcal{D}_N$, updated in Step 17, is used to record the cases where the first state stored in the buffer $\%_x$ is already an unviable state. In such a case, the previous state and action pair that leads to this unviable state is not in the buffers $\%_x$ and $\%_u$ and thus cannot be recorded in $\mathcal{D}$. In particular, identification of such a case is through the combination of Steps 12 and 14, where the condition in Step 12, $\mathcal{U} \setminus \mathcal{D}(x(k-n)) = \emptyset$, implies $x(k-n)$ to be an unviable state and violation of the condition in Step 14, $x(k-n) \in \%_x$, implies the state $x(k-n+1)$, which has been identified as unviable in Step 12, to be the first state in $\%_x$. The occurrence of such cases implies that the current look-back horizon length N is not large enough to capture all state and action pairs that cause a safety violation unavoidable. Therefore, the dataset $\mathcal{D}_N$ can be used to inform an adjustment of N (as in Algorithm 2) and for postanalysis.

In Step 8, the Sim function represents a model of the system that can produce a next state $x(k+1)$ given a pair of current state $x(k)$ and current action $u(k)$. This model does not need to be in an explicit form, that is, it can be a black-box model or a simulation code as long as the state of this model can be read and reset.

Remark 1. The presentation from (3a) to (9) has assumed a deterministic model to simplify expressions. However, Algorithm 1 can also be used to treat stochastic systems, that is, the transition $\text{Sim}(x(k), u(k)) \to x(k+1)$ can have randomness. In this case, Algorithm 1 estimates the complement of the following robust version of $\mathcal{U}_{\text{safe},k^*}(x)$,

$$\begin{aligned} &\mathcal{U}_{\text{safe},k^*}(x) = \{u \in \mathcal{U} \,:\, \text{If } x(0) = x \text{ and } u(0) = u, \text{then with probability } 1, \\ &\text{we have } x(1) \notin \mathcal{X}_0 \text{ and } \exists u(\tau) \in \mathcal{U} \text{ such that } x(\tau+1) \notin \mathcal{X}_0 \text{ for all } \tau = 1, \ldots, k^*\}. \end{aligned} \tag{12}$$

Remark 2. Due to the pointwise sampling and estimate of $\mathcal{U}_{\text{safe},k^*}(x)$, Algorithm 1 is most suitable for problems with a discrete/quantized state space. However, with the following modification to the definition of $\mathcal{D}(x)$ in Steps 7 and 12,

$$\mathcal{D}(x) = \{u \in \mathcal{U} \,:\, (x', u) \in \mathcal{D} \text{ for some } x' \text{ satisfying } \|x' - x\| \le r\}, \tag{13}$$

Algorithm 1 can be applied to continuous state spaces. Note that (13) acts as a quantization of the continuous state space with the parameter $r > 0$ representing a quantization step size.

Algorithm 2. Learning with automatic adjustment of look-back horizon

Input: $\mathcal{U}, \mathcal{X}_0, N_0, M, K, \eta$
Output: $\mathcal{D}, \mathcal{D}_N, N_\text{f}$

1: $N \leftarrow N_0$, $\mathcal{D} \leftarrow \emptyset$;
2: $(\mathcal{D}, \mathcal{D}_N) \leftarrow$ Learning Algorithm $(\mathcal{U}, \mathcal{X}_0, N, M, K, \mathcal{D})$;
3: **while** $|\mathcal{D}_N| > \eta M$ **do**
4: $\quad N \leftarrow N + 1$;
5: $\quad (\mathcal{D}, \mathcal{D}_N) \leftarrow$ Learning Algorithm $(\mathcal{U}, \mathcal{X}_0, N, M, K, \mathcal{D})$;
6: **end while**
7: $N_\text{f} \leftarrow N$;

Algorithm 2 represents a method to automatically adjust the look-back horizon length N. The input N_0 is an initial value of N. The input η represents an acceptable ratio of observed number of cases where the buffers $\%_x$ and $\%_u$ corresponding to the current N are not long enough to capture the state and action pair leading to the first unviable state (as discussed above) to the number of episodes M. If the number of data points in $\mathcal{D}_N$, $|\mathcal{D}_N|$, is greater than ηM, the look-back horizon length N increases by one and learning is continued. Learning is completed when the condition $|\mathcal{D}_N| \leq \eta M$ is satisfied.

2.3 | Learning-based AG for RL policies

In this section, we introduce a few customizations of the AG formulation (1) and the learning algorithm, Algorithm 1, for applications where the nominal policy is obtained through value-based reinforcement learning (RL). Note that typical RL policies do not provide strict constraint enforcement guarantees by themselves, and in this case the AG can be a very useful add-on scheme for enhancing the system's safety.[30]

In value-based RL, each action u is assigned a value at each state x to represent how good this action is at this state.[42] In this article, we represent the action values using the following *Score* function,

$$Score\ (u \mid x), \tag{14}$$

defined for all u and x.

The above *Score* function plays a similar role as the Q-function in Q-learning,[42] that is, it measures how good different actions are at a given state and over a certain evaluation horizon. In particular, better actions correspond to higher *Score* values. Here, we choose to use the term "*Score* function" instead of "Q-function" to admit a more general definition, that is, a more general way for evaluating actions. On the one hand, if the nominal policy is obtained through Q-learning, the above *Score* function can be set to be the Q-function. On the other hand, the *Score* of an action can be defined in different ways, for example, not necessarily be defined as the expected cumulative reward but can be according to worst-case reward, and not necessarily be evaluated over an infinite horizon but can be over a finite horizon as in the approaches of References 43 and 44.

An optimal RL policy always selects an action that corresponds to the highest value at the current state, that is,

$$u_{\text{nom}}(k) \in \arg\max_{u \in \mathcal{U}} Score\ (u \mid x(k)). \tag{15}$$

As discussed above, actions selected by an optimal RL policy may not strictly enforce safety-related requirements, either due to RL convergence to local optima or because safety violations under these actions are rare events. In this case, one can consider using an AG (1) with the following *Metric* function,

$$Metric\ (u_{\text{nom}}(k), u \mid x(k)) = |Score\ (u_{\text{nom}}(k) \mid x(k)) - Score\ (u \mid x(k))|\,. \tag{16}$$

Such an AG formulation is equivalent to an AG that always selects an action that belongs to the set of safe actions, $\mathcal{U}_{\text{safe}}(x(k))$, and has the highest score among all actions in $\mathcal{U}_{\text{safe}}(x(k))$, that is,

$$u(k) \in \arg\max_{u} Score\ (u \mid x(k)), \tag{17a}$$

$$\text{s.t.} \quad u \in \mathcal{U}_{\text{safe}}(x(k)) \subseteq \mathcal{U}. \tag{17b}$$

Meanwhile, because the nominal policy and the AG both prefer selecting actions with higher scores over actions with lower scores, one can prioritize testing the safety/viability of actions with higher scores during the AG training process to improve sample efficiency and learning speed. This can be done by modifying Step 7 of Algorithm 1, which takes actions randomly, to

$$u(k) \in \arg\max_{u \in\ \mathcal{U} \setminus \mathcal{D}(x(k))} Score\ (u \mid x(k)), \tag{18}$$

so an action with the highest score among all actions in $\mathcal{U} \setminus \mathcal{D}(x(k))$ is tested first.

3 | CASE STUDY: IMPROVING AV HIGHWAY SAFETY USING A LEARNING-BASED AG

We consider a case study that represents AV highway driving to illustrate the effectiveness of our proposed learning-based approach to AG design. This section introduces the simulation setup.

3.1 | Vehicle kinematics

We represent the kinematics of a vehicle on a highway using the following discrete-time model,

$$p_x(k+1) = p_x(k) + v_x(k)\Delta T, \tag{19a}$$

$$v_x(k+1) = v_x(k) + a_x(k)\Delta T, \tag{19b}$$

$$p_y(k+1) = p_y(k) + v_y(k)\Delta T, \tag{19c}$$

where p_x and p_y designate, respectively, the vehicle's longitudinal and lateral positions on the highway, v_x and v_y designate its longitudinal and lateral velocities, a_x designates its longitudinal acceleration, and ΔT is the sampling period. Such a model is frequently used to represent vehicle motion on highways in the literature.[21,25,26,39] In this model, (p_x, v_x, p_y) represents the single vehicle's state, and $u = (a_x, v_y)$ is its control action. In this case study, we assume the vehicle's longitudinal velocity v_x is bounded to the range $[v_{\min}, v_{\max}] = [62, 98]$ km/h to represent common speeds for highway driving. Such speed lower and upper-bound values are set following References 39 and 45. If the velocity value calculated according to (19ab) gets outside this range, it is saturated to this range.

Following References 21,25,26,39, we assume the vehicle takes action u from a finite set of action options, called the "action space," as follows

$$u(k) \in \mathcal{U} = \left\{(-a_{\max}, 0), (-a_{\text{nom}}, 0), (0, 0), (a_{\text{nom}}, 0), (a_{\max}, 0), (0, -v_y), (0, v_y)\right\}, \tag{20}$$

where $a_{\max}$ and a_{nom} represent the vehicle's maximum and nominal acceleration/deceleration values, respectively. The seven actions in $\mathcal{U}$ correspond to the maneuvers "hard decelerate", "nominal decelerate", "maintain speed", "nominal accelerate", "hard accelerate", "move to right", and "move to left", respectively.[39]

3.2 | In-traffic state representation

Note that when a vehicle is operating in traffic, its state not only depends on its own position and velocity but also on the positions and velocities of its neighboring vehicles. In this case study of AV highway driving, we adopt the affordance indicator method[46] in the form used in Reference 12 to represent the vehicle's state in traffic.

In particular, the ego vehicle observes the ranges (longitudinal distance $\Delta p_x(k) = p_x^{\text{other}}(k) - p_x^{\text{ego}}(k)$) and range rates (longitudinal relative velocity $\Delta v_x(k) = v_x^{\text{other}}(k) - v_x^{\text{ego}}(k)$) to the vehicles in its front, front right, front left, rear, rear right, and rear left (see Figure 2). We classify ranges $\Delta p_x(k)$ into three categorical values $\mathcal{P} = \{\text{close}, \text{nominal}, \text{far}\}$ according to

$$\Delta p_x(k) \in \begin{cases} \text{close} & \text{if} \quad |\Delta p_x(k)| \le p^{\text{lb}}, \\ \text{nominal} & \text{if} \quad p^{\text{lb}} < |\Delta p_x(k)| \le p^{\text{ub}}, \\ \text{far} & \text{if} \quad |\Delta p_x(k)| > p^{\text{ub}}, \end{cases} \tag{21}$$

where p^{lb} and p^{ub} are specified thresholds. Similarly, we classify range rates $\Delta v_x(k)$ into three categorical values $\mathcal{V} = \{\text{approaching}, \text{stable}, \text{moving away}\}$ according to

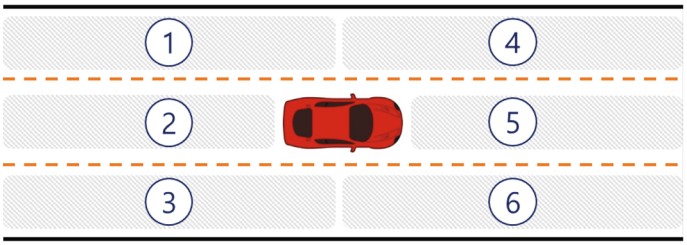

FIGURE 2 Autonomous vehicle in-traffic state representation

$$\Delta v_x(k) \in \begin{cases} \text{approaching} & \text{if} \quad \Delta p_x(k) \geq 0 \ \text{ and } \ \Delta v_x(k) < 0 \\ & \text{or} \quad \Delta p_x(k) < 0 \ \text{ and } \ \Delta v_x(k) > 0, \\ \text{stable} & \text{if} \quad \Delta v_x(k) = 0, \\ \text{moving away} & \text{if} \quad \Delta p_x(k) \geq 0 \ \text{ and } \ \Delta v_x(k) > 0 \\ & \text{or} \quad \Delta p_x(k) < 0 \ \text{ and } \ \Delta v_x(k) < 0. \end{cases}$$

The above classifications (21) and (22) describe the relative motion of each neighboring vehicle with respect to the ego vehicle. For instance, a vehicle in the front left of the ego vehicle (area 4 in Figure 2) can be "close and approaching," and so forth. These classifications also represent a discretization/quantization of the state space. Note that such a discretization/quantization is not always necessary for the application of our learning AG algorithm (see Remark 2).

The ego vehicle also observes its lane l. For instance, $l \in \{1, 2, 3\}$ when the vehicle is operating on a three-lane highway. Therefore, the in-traffic state of the ego vehicle is represented by the following vector,

$$\begin{aligned} x(k) = \ & (\Delta p_x^{\mathrm{f}}(k), \Delta v_x^{\mathrm{f}}(k), \Delta p_x^{\mathrm{fl}}(k), \Delta v_x^{\mathrm{fl}}(k), \Delta p_x^{\mathrm{fr}}(k), \Delta v_x^{\mathrm{fr}}(k), \\ & \Delta p_x^{\mathrm{r}}(k), \Delta v_x^{\mathrm{r}}(k), \Delta p_x^{\mathrm{rl}}(k), \Delta v_x^{\mathrm{rl}}(k), \Delta p_x^{\mathrm{rr}}(k), \Delta v_x^{\mathrm{rr}}(k), l), \end{aligned} \tag{22}$$

which takes values in a discretized state space of size 3^{13}.

3.3 | Simulated traffic environment

We aim to train an AG using our proposed learning-based approach to minimize safety constraint violations between the autonomous ego vehicle and other vehicles, where a safety constraint violation is defined as an occurrence of the distance between two vehicles violating a safe threshold.[39] For this, we need a training environment that can simulate the transitions of the ego vehicle's in-traffic state $x(k)$ as results of its action $u(k)$ and other vehicles' responses to its action (see Step 8 of Algorithm 1). In this case study, we use the game-theoretic traffic simulator developed in Reference 39 as both the training and testing environment.

The traffic simulator of Reference 39 uses level-k game theory[47] to model the interactions among AVs and human-operated vehicles in traffic. In particular, it uses different levels, $k = 0, 1, 2$, to model different driver types. According to the definition of a level-0 driver,[39] a level-1 driver drives aggressively and a level-2 driver drives cautiously. See References 39 and 45 for more detailed descriptions on this level-k game theory-based traffic simulator. This simulator has been validated using simulated and real-world traffic data in References 48 and 49, and has been used by several researchers as the training and/or testing environments for developing AV control policies, for example, in References 21, 22, and 26.

3.4 | Nominal control policies

The AG monitors and corrects the action commands generated by a nominal control policy. In this case study, we consider the following three nominal policies:

TABLE 1 Safety violation rates for different look-back horizons

N	0	1	2	3
Rate (%)	64.3	51.5	11.5	0

1. RL policy,
2. Stackelberg policy,
3. decision tree policy.

The RL policy is a policy trained using an RL algorithm in the simulated traffic environment. Because in this case study we use the level-k game theory-based traffic simulator[39] as the training environment, we also refer to this policy as a "level-k policy." The Stackelberg policy and the decision tree policy were originally proposed in References 43 and 44, respectively. Our implementations are modified versions of these policies so that they are more compatible with our traffic simulator. For detailed descriptions of our modified versions of the Stackelberg and the decision tree policies, see Reference 39. Note that our focus of this article is not on the design of these or other nominal AV policies, but on the use of learning-based AG to improve the safety of such existing policies.

The RL, Stackelberg, and decision tree policies are all value-based policies taking the form of (15), that is, they assign values to the actions at each state to represent how good they are at this state and select the one with the highest value. For these nominal policies, we let the AG correct action commands according to the expression (17a). Note that although these policies use carefully designed reward functions including penalty terms for constraint violations to promote safety, they cannot eliminate all safety constraint violations due to the following reasons: For the RL policy, because it aims to maximize the expected value (i.e., probability-weighted average) of the reward, it may select actions that correspond to high performance in most cases although causing occasional constraint violations. Safety constraint violations may also be attributed to suboptimal behavior of the policy related to RL convergence to local extrema.[50] For the Stackelberg policy and the decision tree policy, because they use certain models to predict neighboring vehicles' behaviors over their planning horizons (the Stackelberg policy uses a Stackelberg game formulation for such predictions and the decision tree policy predicts each of other vehicles' motion according to a constant velocity model), safety constraint violations may occur due to mismatches between their predictions and the neighboring vehicles' actual behaviors. In such cases, our learning-based AG approach can be a viable option for improving their safety.

4 | RESULTS

We present and discuss the simulation results of our case study in this section.

4.1 | Look-back horizon

Firstly, to clearly see the importance of a sufficient horizon length N for constraint violation avoidance, we randomly generate a set of traffic situations[1] and train AGs using our learning algorithm, Algorithm 1, with varying N on this same set of traffic situations (i.e., we initialize the traffic simulator to this given set of traffic situations and run forward simulation and our learning algorithm). For this experiment, we consider a three-lane highway with dense traffic (30 vehicles on a 1000-m cyclic road) composed of purely level-1 drivers. Note that the level-1 driver model corresponds to the most aggressive driving style.[39] Therefore, this dense and aggressive traffic environment is challenging to AV policies in terms of causing vehicle safety constraint violations easily. Also, we consider the Stackelberg policy as the nominal AV control policy in both learning and testing.

After learning, we test the obtained AGs by measuring and comparing their resulting *safety constraint violation rates* on the training set of traffic situations, which is defined as the percentage of traffic situations where a safety constraint violation between the ego AV (controlled by nominal policy + AG) and some other vehicle occurs during forward simulation (i.e., percentage of situations where the trained AG fails to avoid safety constraint violations). The results are reported in Table 1, where $N = 0$ corresponds to the case without an AG.

[1] A traffic situation is a certain assignment of position and velocity values of all vehicles in traffic.

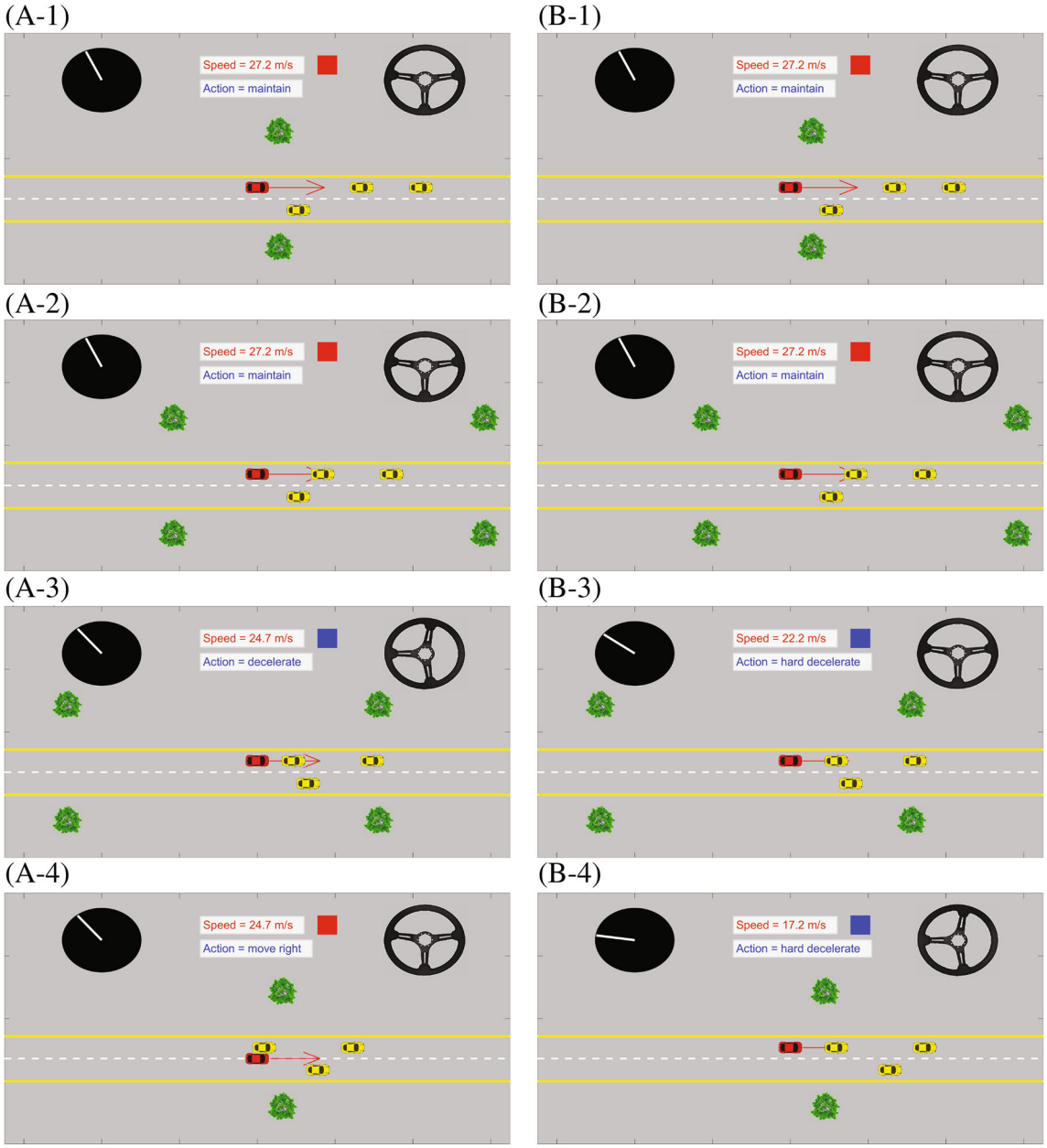

FIGURE 3 Simulation snapshots of a two-lane highway scenario. (A-1)–(A-4) Consecutive steps where the ego AV is controlled by the Stackelberg policy without an AG. (B-1)–(B-4) Consecutive steps where the ego AV is controlled by the Stackelberg policy augmented with the trained AG

From Table 1 we can see that the safety constraint violation rate on the training set of traffic situations can only be slightly reduced (from 64.3% to 51.5%) using an AG with a short look-back horizon of $N = 1$, which implies $N = 1$ is not sufficient for constraint violation avoidance. For instance, when two vehicles approaching each other are already very close, no feasible action exists to avoid a safety constraint violation at the next time step. When N is increased from one to two, most safety constraint violations can be avoided, demonstrated by the significant decrease in the rate (from 51.5% to 11.5%). However, there still exist some inevitable constraint violations. When N is increased to three, the trained AG ensures no safety constraint violation occurrence to the ego AV on the training dataset (rate = 0%). Therefore, we recommend $N = 3$ to be a suitable look-back horizon length and we use $N = 3$ in the simulations of the next subsection.

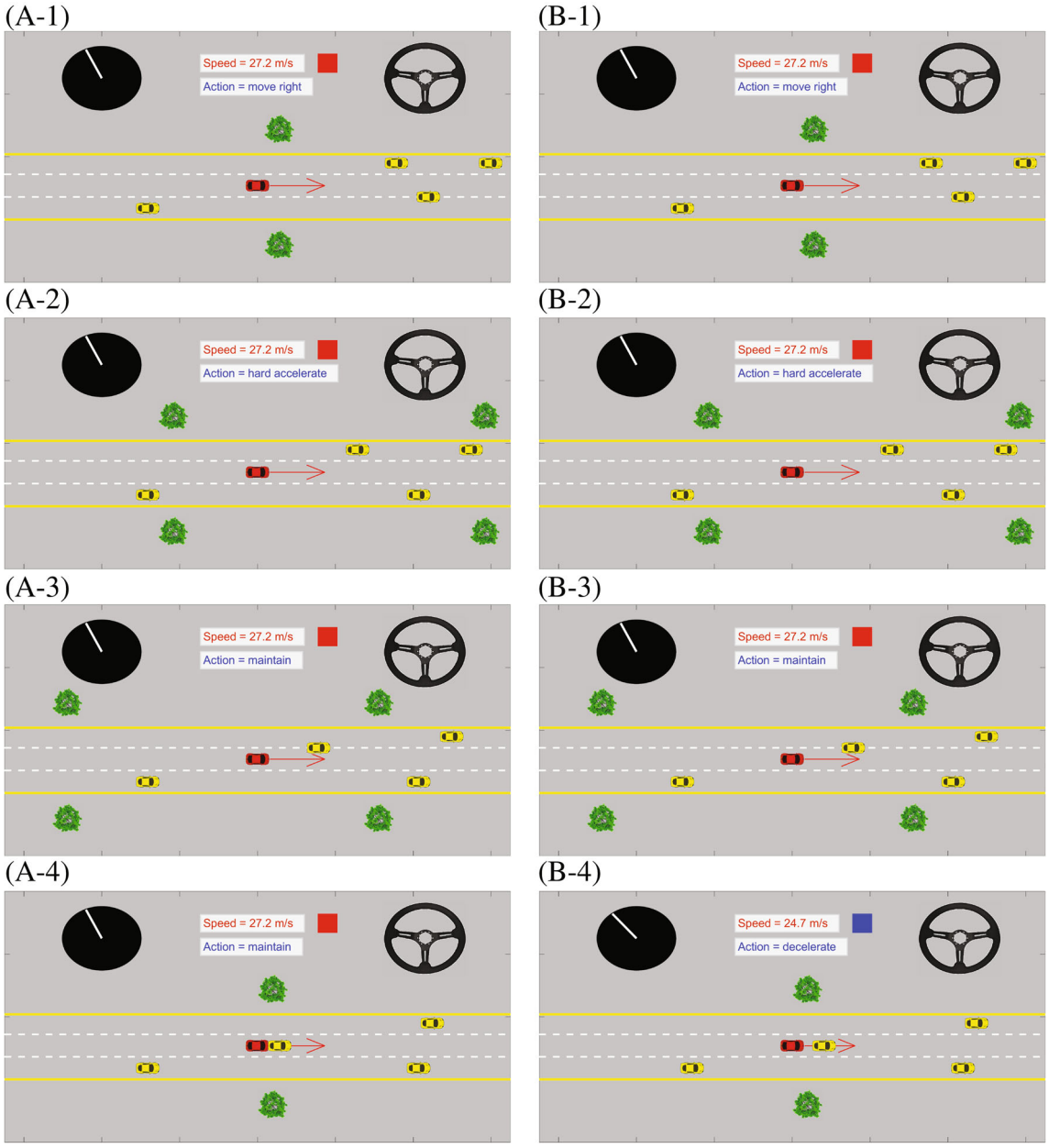

FIGURE 4 Simulation snapshots of a three-lane highway scenario. (A-1)–(A-4) Consecutive steps where the ego AV is controlled by the Stackelberg policy without an AG. (B-1)–(B-4) Consecutive steps where the ego AV is controlled by the Stackelberg policy augmented with the trained AG

4.2 | Safety improvements for different nominal policies

We now test our learning-based AG approach for improving the in-traffic safety of AVs with different nominal control policies. In this set of experiments, we consider both a highway of two lanes and a highway of three lanes with varying traffic density. Following References 39 and 45, we consider a traffic environment composed of a mixture of level-k drivers (10% level-0 drivers, 60% level-1 drivers, and 30% level-2 drivers) to represent the behavior heterogeneity of real-world drivers. Due to the dominance of level-1 drivers, such a traffic environment tends to exhibit more aggressive driving behaviors than real-world traffic. On the one hand, this may result in more vehicle safety constraint violations than observed in real-world traffic. On the other hand, this may facilitate the uncovering of challenging/dangerous scenarios and thus can accelerate the AG training and testing.

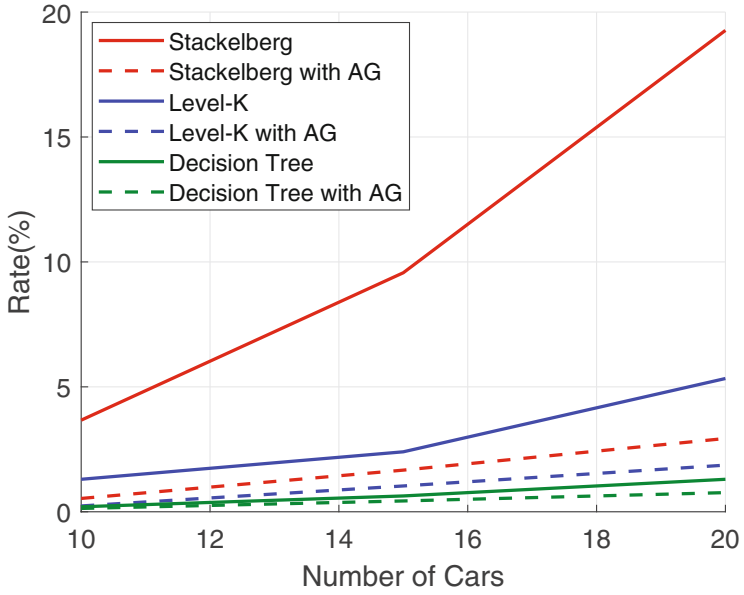

FIGURE 5 Safety constraint violation rates for two-lane highway

For learning, we randomly generated traffic situations to train the AG. Figure 3 illustrates a traffic situation on a highway of two lanes where the red car in the middle is the ego AV and the other yellow cars are environmental vehicles. Panels (A-1)–(A-4) in the left column show snapshots at four consecutive steps of the simulation where the ego AV is controlled by the Stackelberg policy without an AG. When a yellow car in front decelerates and approaches, the ego AV first decelerates at Step 3 and then tries to perform a lane change to the right at Step 4. However, due to the fast approaching of the yellow car, the ego AV fails to evade the yellow car by the right lane change, resulting in a safety constraint violation between the two cars. Panels (B-1)–(B-4) in the right column show the same traffic situation, where the ego AV is now controlled by the Stackelberg policy augmented with the trained AG. The AG corrects the actions at Steps 3 and 4 from "decelerate" and "move to right" to "hard decelerate," which avoids the safety constraint violation between the two cars. Figure 4 illustrates a traffic situation on a three-lane highway. The trained AG corrects the action of the ego AV at Step 4 from "maintain" to "decelerate" to avoid a safety constraint violation between the ego AV and a yellow car changing lanes from the left.

We now evaluate the effectiveness of the trained AG for safety improvement by measuring and comparing the *safety constraint violation rates* of the policies before and after augmentation with the AG. Following References 39 and 45, the safety constraint violation rate is now defined to be the percentage of simulation episodes starting with random initial conditions and lasting 200 s where at least one safety constraint violation occurs to the ego AV over the episode. Note that due to the randomness of the initial conditions, the traffic situations encountered by the ego AV during testing are not always the same as the traffic situations encountered during training. The rates versus varying traffic densities (represented by the number of vehicles on a 1000-m cyclic road) for a two-lane highway are reported in Figure 5 and for a three-lane highway in Figure 6.

From Figures 5 and 6 we can make the following observations: (1) For the Stackelberg policy, which has high safety constraint violation rates before augmentation with the AG, the use of the trained AG can significantly reduce the rates. For the RL policy (also referred to as the "level-k policy" in the figures), the safety constraint violation rates are reduced by half after augmentation with the trained AG. For the decision tree policy, which has high performance before augmentation (in terms of having low safety constraint violation rates), augmentation with the trained AG can slightly reduce the rates. In summary, our learning-based AG approach can improve the safety of every policy, and the amount of improvement is policy-dependent.

Note that our implementation of the Stackelberg policy has the lowest computational complexity, and our implementation of the decision tree policy has the highest computational complexity.[39] This fact contributes to the difference between their nominal performance (i.e., safety performance before augmentation with AG). Furthermore, the less significant safety improvement by the trained AG for the decision tree policy is related to the long-tail phenomenon of

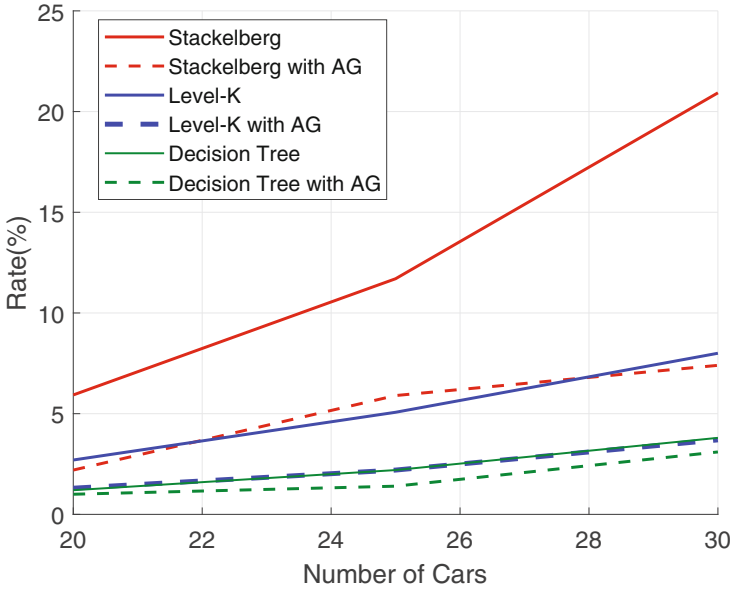

FIGURE 6 Safety constraint violation rates for three-lane highway

traffic events,[51] which causes further improvement through learning from certain traffic events to become more and more difficult as the performance improves.

A possible approach to improving the learning efficiency (i.e., achieving more significant safety improvement within the same or a shorter amount of training time) is through guided sampling or experimental design instead of the random generation of traffic situations during learning. For instance, a dataset of traffic collision incidents (including frames before collision events) might be used to inform the generation/design of traffic situations for AG training. This is left as a topic to investigate in future research.

5 | CONCLUSION AND DISCUSSION

In this article, we introduced a novel approach based on learning to AG design for improving AV safety. Specifically, we presented a learning algorithm to train an AG as a safety supervisor for monitoring and adjusting the action commands generated by nominal AV control policies to avoid undesirable AV behavior. To improve AV safety for operating in mixed-autonomy traffic, we proposed to train the AG in a traffic simulator capable of representing in-traffic interactions among vehicles. We then illustrated the effectiveness of our proposed learning-based AG approach to improving AV in-traffic safety through a simulation case study representing AV highway driving with different nominal policies. Simulation results showed that AV safety could be improved moderately to considerably by augmentation of the nominal policies with the trained AG.

The proposed learning-based AG uses a collection of unviable state and action data points identified during learning to approximate the set of viable/unviable actions at each state. For high-dimensional state spaces, such a strategy may lead to high memory requirements. Cluster-based or neural network-based approximations of the dataset may be employed to alleviate this issue, where the AG uses a relatively small number of clusters or a neural network, instead of searching over a dataset, for online prediction of whether an action is viable/unviable at the current state, as pursued in References 31 and 52. Such strategies will be investigated in future work.

ACKNOWLEDGMENT

This research has been funded by the Ford Motor Company.

DATA AVAILABILITY STATEMENT

The data that support the findings of this study are available from the corresponding author upon reasonable request.

ORCID

Kyoungseok Han https://orcid.org/0000-0002-4986-2053
Nan Li https://orcid.org/0000-0001-7928-8796
Ilya Kolmanovsky https://orcid.org/0000-0002-7225-4160

REFERENCES

1. Yeong DJ, Velasco-Hernandez G, Barry J, Walsh J. Sensor and sensor fusion technology in autonomous vehicles: a review. Sensors. 2021;21(6):2140.
2. Van Brummelen J, O'Brien M, Gruyer D, Najjaran H. Autonomous vehicle perception: the technology of today and tomorrow. Transp Res C Emerg Technol. 2018;89:384-406.
3. Liu L, Lu S, Zhong R, et al. Computing systems for autonomous driving: state of the art and challenges. IEEE Internet Things J. 2020;8(8):6469-6486.
4. Campbell M, Egerstedt M, How JP, Murray RM. Autonomous driving in urban environments: approaches, lessons and challenges. Philos Trans R Soc A Math Phys Eng Sci. 1928;2010(368):4649-4672.
5. Ersal T, Kolmanovsky I, Masoud N, et al. Connected and automated road vehicles: state of the art and future challenges. Veh Syst Dyn. 2020;58(5):672-704.
6. Badue C, Guidolini R, Carneiro RV, et al. Self-driving cars: a survey. Expert Syst Appl. 2021;165:113816.
7. Schwarting W, Alonso-Mora J, Rus D. Planning and decision-making for autonomous vehicles. Annu Rev Control Robot Auton Syst. 2018;1:187-210.
8. Claussmann L, Revilloud M, Gruyer D, Glaser S. A review of motion planning for highway autonomous driving. IEEE Trans Intell Transp Syst. 2019;21(5):1826-1848.
9. Khayatian M, Mehrabian M, Andert E, et al. A survey on intersection management of connected autonomous vehicles. ACM Trans Cyber-Phys Syst. 2020;4(4):1-27.
10. Sadigh D, Sastry S, Seshia SA, Dragan AD. Planning for autonomous cars that leverage effects on human actions. Robot Sci Syst. 2016;2:1-9.
11. Sankar GS, Han K. Adaptive robust game-theoretic decision making strategy for autonomous vehicles in highway. IEEE Trans Veh Technol. 2020.
12. Zhang M, Li N, Girard A, Kolmanovsky I. A finite state machine based automated driving controller and its stochastic optimization. Proceedings of the 2017 Dynamic Systems and Control Conference; Vol. 58288, 2017:V002T07A002.
13. Censi A, Slutsky K, Wongpiromsarn T, et al Liability, ethics, and culture-aware behavior specification using rulebooks. Proceedings of the 2019 International Conference on Robotics and Automation (ICRA); 2019:8536-8542.
14. Ding J, Li L, Peng H, Zhang Y. A rule-based cooperative merging strategy for connected and automated vehicles. IEEE Trans Intell Transp Syst. 2019;21(8):3436-3446.
15. Xiao W, Mehdipour N, Collin A,et al. Rule-based optimal control for autonomous driving. Proceedings of the 12th ACM/IEEE International Conference on Cyber-Physical Systems; 2021:143-154.
16. Zhang J, Cho K. Query-efficient imitation learning for end-to-end simulated driving. Proceedings of the 31st AAAI Conference on Artificial Intelligence; 2017:2891-2897.
17. Codevilla F, Müller M, López A, Koltun V, Dosovitskiy A. End-to-end driving via conditional imitation learning. Proceedings of the 2018 IEEE International Conference on Robotics and Automation (ICRA); 2018:4693-4700.
18. Chen J, Yuan B, Tomizuka M. Deep imitation learning for autonomous driving in generic urban scenarios with enhanced safety. Proceedings of the 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS); 2019:2884-2890.
19. Sallab AE, Abdou M, Perot E, Yogamani S. Deep reinforcement learning framework for autonomous driving. Electron Imaging. 2017;2017(19):70-76.
20. You C, Lu J, Filev D, Tsiotras P. Advanced planning for autonomous vehicles using reinforcement learning and deep inverse reinforcement learning. Robot Auton Syst. 2019;114:1-18.
21. Nageshrao S, Tseng HE, Filev D. Autonomous highway driving using deep reinforcement learning. Proceedings of the 2019 IEEE International Conference on Systems, Man and Cybernetics (SMC); 2019:2326-2331.
22. Li H, Li N, Kolmanovsky I, Girard A. Energy-efficient autonomous vehicle control using reinforcement learning and interactive traffic simulations. Proceedings of the 2020 American Control Conference (ACC); 2020:3029-3034.
23. Duan J, Li SE, Guan Y, Sun Q, Cheng B. Hierarchical reinforcement learning for self-driving decision-making without reliance on labelled driving data. IET Intell Transp Syst. 2020;14(5):297-305.
24. Chen J, Li SE, Tomizuka M. Interpretable end-to-end urban autonomous driving with latent deep reinforcement learning. IEEE Trans Intell Transp Syst. 2021.
25. Chen D, Jiang L, Wang Y, Li Z. . Autonomous driving using safe reinforcement learning by incorporating a regret-based human lane-changing decision model. Proceedings of the 2020 American Control Conference (ACC); 2020:4355-4361.
26. Baheri A, Nageshrao S, Tseng HE, Kolmanovsky I, Girard A, Filev D. Deep reinforcement learning with enhanced safety for autonomous highway driving. Proceedings of the 2020 IEEE Intelligent Vehicles Symposium (IV); 2020:1550-1555.
27. Li Z, Kalabić U, Chu T. Safe reinforcement learning: learning with supervision using a constraint-admissible set. Proceedings of the 2018 American Control Conference (ACC); 2018:6390-6395.

28. Cheng R, Orosz G, Murray RM, Burdick JW. End-to-end safe reinforcement learning through barrier functions for safety-critical continuous control tasks. Proceedings of the AAAI Conference on Artificial Intelligence; Vol. 33, 2019:3387-3395.
29. Chen Y, Hereid A, Peng H, Grizzle J. Enhancing the performance of a safe controller via supervised learning for truck lateral control. J Dyn Syst Meas Control. 2019;141(10):06 2019.
30. Li Y, Li N, Tseng HE, Girard A, Filev D, Kolmanovsky I. Safe reinforcement learning using robust action governor. Proceedings of the Learning for Dynamics and Control (L4DC) Conference; 2021:1093-1104.
31. Han T, Nageshrao S, Filev DP, Özgüner Ü. An online evolving framework for advancing reinforcement-learning based automated vehicle control. IFAC-PapersOnLine. 2020;53(2):8118-8123.
32. Liu K, Li N, Rizzo D, Garone E, Kolmanovsky I, Girard A. Model-free learning to avoid constraint violations: an explicit reference governor approach. Proceedings of the 2019 American Control Conference (ACC); 2019:934-940.
33. Liu K, Li N, Kolmanovsky I, Rizzo D, Girard A. Model-free learning for safety-critical control systems: a reference governor approach. Proceedings of the 2020 American Control Conference (ACC); 2020:943-949.
34. Li N, Han K, Girard A, Tseng HE, Filev D, Kolmanovsky I. Action governor for discrete-time linear systems with non-convex constraints. IEEE Control Syst Lett. 2020;5(1):121-126.
35. Li Y, Li N, Tseng HE, Girard A, Filev D, Kolmanovsky I. Robust action governor for discrete-time piecewise affine systems with additive disturbances. IEEE Control Syst Lett. 2021.
36. Romdlony MZ, Jayawardhana B. Stabilization with guaranteed safety using control Lyapunov–Barrier function. Automatica. 2016;66:39-47.
37. Ames AD, Xu X, Grizzle JW, Tabuada P. Control barrier function based quadratic programs for safety critical systems. IEEE Trans Automat Contr. 2016;62(8):3861-3876.
38. Ames AD, Coogan S, Egerstedt M, Notomista G, Sreenath K, Tabuada P. Control barrier functions: theory and applications. Proceedings of the 18th European Control Conference (ECC); 2019:3420-3431.
39. Li N, Oyler DW, Zhang M, Yildiz Y, Kolmanovsky I, Girard AR. Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems. IEEE Trans Control Syst Technol. 2017;26(5):1782-1797.
40. Garone E, Di Cairano S, Kolmanovsky I. Reference and command governors for systems with constraints: a survey on theory and applications. Automatica. 2017;75:306-328.
41. Gray A, Gao Y, Lin T, Hedrick JK, Tseng HE, Borrelli F. Predictive control for agile semi-autonomous ground vehicles using motion primitives. Proceedings of the 2012 American Control Conference (ACC); 2012:4239-4244.
42. Sutton RS, Barto AG. Reinforcement Learning: An Introduction. MIT Press; 2018.
43. Yoo JH, Langari R. Stackelberg game based model of highway driving. Proceedings of the 2012 Dynamic Systems and Control Conference; Vol. 45295, 2012:499-508.
44. Claussmann L, Carvalho A, Schildbach G. A path planner for autonomous driving on highways using a human mimicry approach with binary decision diagrams. Proceedings of the 2015 European Control Conference (ECC); 2015:2976-2982.
45. Li N, Zhang M, Yildiz Y, Kolmanovsky I, Girard A. Game theory-based traffic modeling for calibration of automated driving algorithms. Control Strategies for Advanced Driver Assistance Systems and Autonomous Driving Functions. Springer; 2019:89-106.
46. Chen C, Seff A, Kornhauser A, Xiao J. Deepdriving: learning affordance for direct perception in autonomous driving. Proceedings of the IEEE International Conference on Computer Vision; 2015:2722-2730.
47. Stahl DO, Wilson PW. On players' models of other players: theory and experimental evidence. Games Econ Behav. 1995;10(1):218-254.
48. Su G, Li N, Yildiz Y, Girard A, Kolmanovsky I. A traffic simulation model with interactive drivers and high-fidelity car dynamics. IFAC-PapersOnLine. 2019;51(34):384-389.
49. Albaba M, Yildiz Y, Li N, Kolmanovsky I, Girard A. Stochastic driver modeling and validation with traffic data. Proceedings of the 2019 American Control Conference (ACC); 2019:4198-4203.
50. Jaakkola T, Singh SP, Jordan MI. Reinforcement learning algorithm for partially observable Markov decision problems. Adv Neural Inf Process Syst. 1995;345-352.
51. Fraade-Blanar L, Blumenthal MS, Anderson JM, Kalra N. Measuring automated vehicle safety: forging a framework; 2018.
52. Liu K, Li N, Kolmanovsky I, Rizzo D, Girard A. Tanker truck rollover avoidance using learning reference governor. SAE Int J Adv Current Pract Mob. 2021;3:1385-1394.

How to cite this article: Han K, Li N, Tseng E, Filev D, Kolmanovsky I, Girard A. Improving autonomous vehicle in-traffic safety using learning-based action governor. *Adv Control Appl.* 2022;4(2):e101. doi: 10.1002/adc2.101