Adaptive Controllers for Vehicle Velocity Control for Microscopic Traffic Simulation Models

Yiannis Papelis, Omar Ahmad & Horatiu German
National Advanced Driving Simulator, The University of Iowa, USA

Abstract

This paper describes a hybrid follow controller designed for regulating the speed of automatically controlled vehicles in car-following situations within a virtual environment of a microscopic traffic simulation system. Unlike earlier work in follow algorithms where the modeling goal is the correlation of aggregate traffic behavior, the primary purpose of this model is to appear realistic when viewed by a human participant driving a vehicle within the virtual environment of a driving simulator. The paper briefly describes the overall simulation system along with specific problems that had to be addressed in this particular application. Examples of the performance of the model are provided for typical driving situations.

Introduction, Background and Motivation

Microscopic traffic simulation (MTS) models are often utilized within virtual driving environments to facilitate testing of transportation technologies related to vehicle, infrastructure, and human factors issues. Within an MTS system, individual vehicles are controlled by driver models whose behavior is meant to resemble the decision-making abilities and reactions of human drivers. As part of the National Advanced Driving Simulator (NADS) [1] project, a comprehensive MTS system has been developed [2]. During its development, a key problem that emerged was the implementation of realistic velocity control algorithms for car following that are applicable to a wide range of operational conditions yet appear realistic when viewed from within the virtual environment. Whereas classical controllers can be optimized by using a linearized model of the vehicle, it is
difficult to tune such controllers for the wide range of conditions encountered in everyday driving such as stop-and-go and emergency braking. For example, a controller with low settling time may be appropriate for a quick stop triggered by the change of a traffic light; however, that same controller may be inappropriate for a smooth velocity increase when a vehicle enters a highway. In addition to the difficulty of utilizing a single controller for many situations, the tactical behavioral model needs to dynamically modify the controller's behavior to simulate changing aggressiveness, loss of patience, and other similar behavioral shifts. The solution presented in this paper overviews the NADS MTS system and how it addressed this particular problem by utilizing a set of layered classical controllers, each of which is focused on a sub-set of the control tasks involved in driving. The controllers are tightly integrated with the tactical behavioral model, which dynamically switches among them and often fuses multiple controller outputs depending on the current driving situation.

The remainder of the paper is organized as follows. Following a short research background in follow controllers, a summary of the overall MTS is given to provide the context within which the follow controller operates. The controller algorithm is described next, followed by a few examples of its performance under various driving situations. A conclusion and references complete the paper.

Research Background

Research in traffic simulation began as early as in the mid-1950s. By the mid-1980s, significant developments of car following algorithms for use in microscopic traffic simulation models had taken place. One of the most widely known models is CARSIM [3], a validated model that supports stop-and-go and regular following conditions. Other models include FRESIM and NETSIM; they are compared to CARSIM and field data by Aycin and Benekohal [4]. Generally, after calibration most of these models compare favorably to field data, making microscopic traffic simulation a very effective tool for predicting traffic flows and delays in actual road networks. However, most validation efforts related to these models utilize macroscopic metrics such as average delays, average vehicle speeds, and inter-vehicle spacing. Validation efforts do not include vehicle-centered metrics, such as individual acceleration profiles or driver attention demand. The model's stability and sensitivity to step disturbances—such as when a vehicle cuts in front of another, in effect introducing a stop reduction in the following distance—can have a profound effect on individual vehicle behavior. For example, a model that oscillates following a step disturbance or yields negative following distances may provide macroscopic traffic measures that match field data, but is not acceptable from the standpoint of individual vehicle behavior. A more detailed analysis of the stability and performance of typical models is provided by Aycin and Benekohal [5]. The primary focus of the work presented here is to address such issues by designing a car-following algorithm that operates integrated into a larger autonomous driving simulation model and that provides individual vehicle behaviors that are stable and appear realistic when viewed by a human participant in the virtual environment.
Driving Simulation Challenges and Motivation

One faces several challenges when developing an MTS for use in driving simulation applications. In a driving simulator, a human participant is driving a vehicle on virtual roads populated by multiple computer-controlled or robot drivers. Unlike models used for traffic congestion studies, the model controlling the robot drivers must realistically interact with the driver. In a high-fidelity simulator such as the NADS, the participant is driving an actual vehicle cab, is surrounded by 360 degrees of highly realistic visual imagery, hears audio cues that closely match the cues of real driving and receives very realistic motion cues caused by the correlated movement of the motion subsystem. If the robot drivers within the virtual environment appear unrealistic, (i.e., oscillate, collide with other vehicles, reach excessive accelerations or decelerations during normal driving) the overall experience of immersion is significantly diminished, making research problematic. Figure 1 illustrates the simulator's setup on a scene containing numerous computer-controlled traffic participants.

Whereas most literature concentrates on a specific aspect of driving such as following or lane tracking, an MTS used in a driving simulator must include all relevant models within an integrated environment. In addition, behaviors that have not been widely studied, such as lane changes and passing, must also be incorporated into the model [6,7]. Finally, to provide adequate variability in behaviors among different robot drivers (follow distance, stopping distance, etc.), the model should contain modifiable parameters that allow perturbations of the basic model without sacrificing stability.

Figure 1: Driving simulator virtual environment.
The Microscopic Simulation Model

The challenges involved in effectively fusing all behaviors are partially addressed through the use of Hierarchical Concurrent State Machines (HCSMs) [8]. Due to space limitations, a comprehensive description of the HCSM model cannot be included here; however, Figure 2 provides a conceptual overview of the overall traffic simulation model used in this work.

In this diagram, the following controller is distributed among the Object Follow Logic, Lane Change Logic, and Control Input Fusion Logic blocks. Within the Object Follow Logic block, the following algorithm is responsible for producing an acceleration value that when fed to the vehicle dynamics will yield a following distance consistent with published field data, or with user-specified distributions. Within the Lane Change Logic block, the following algorithm is responsible for selecting which target to follow during the implementation of a lane change.

Figure 2: Block diagram of autonomous driver model.
Follow Controller Description

A block diagram of the controller algorithm is shown in Figure 3. The inputs to the algorithm are updated at each simulation step by the external framework. The Input State Vector contains position velocity acceleration triples for the current vehicle (the controlled one), along with the lead vehicle.

In addition, the state vector includes the following distance (FD), desired following distance, and an identifier that uniquely identifies the lead vehicle. Inputs are updated at each simulation time step whereas parameters remain static during execution, although they can vary for each instance of an autonomous robot driver. During each simulation frame, the controller utilizes the inputs and associated parameter to produce two outputs: the desired acceleration and a binary signal indicating algorithm engagement. The algorithm engagement flag becomes false in cases where following is not possible, such as when the lead vehicle is so far ahead that there is no meaningful follow interaction. When the engage flag is true, the acceleration value is valid.

Figure 3: Block diagram of follow algorithm.
The Approach Velocity Computation and Low Speed Boost blocks ensure that the controller does not exhibit excessive delays in reaching its goal when stopping or approaching a slow-moving or stopped vehicle. The Approach Velocity Calculation block uses the current distance to the lead vehicle along with the velocity differential and determines the deceleration required to reach zero velocity error when the following distance will reach the following distance. The formula used is \( \text{MaxVel} = \sqrt{L V V - 2 \times \text{MaxDecRate} \times FD} \). The difference between the current and maximum velocity is used to clip and scale the normal output of the controller. This ensures that the follower will not begin braking until it has gotten near enough to the lead vehicle.

Similarly, the Low Speed Boost ensures that the velocity does not become infinitesimally small when approaching a stopped vehicle, something that can happen when low proportional gains are used. In such situations, a small acceleration is added to ensure that the goal is reached within a reasonably bounded time period.

The emergency controller consists of two classical PD controllers, one tracking the lead vehicle acceleration and one tracking the desired following distance. Similar to the normal controller, the output of the two PD controllers is blended based on the following distance, with acceleration tracking taking precedence when the following distance is large.

The Normal and Emergency controller outputs are combined according to the time to collision (TTC). Two thresholds are established, TTC1 and TTC2. When TTC<TTC1, the emergency controller component is 100%, whereas when TTC>TTC2, the normal controller component is 100%. When TTC1<=TTC<TTC2, the controller outputs are proportionally combined.

The Following Object Change Control block acts as an intelligent filter by smoothing step changes to the following distance caused by a change in the lead vehicle. Such changes can happen when another vehicle cuts in front of the follower, or when the follower performs a lane change. When the resultant TTC is larger than a minimum threshold, the Effective FD smoothly transitions between the value it had before the object switch and the actual value. No filtering takes place when the TTC is below the threshold, as that indicates a potential collision threat.

**Interaction with other behaviors**

As seen in Figure 1, the tactical level receives acceleration commands from multiple driving behaviors in addition to the follow (Control Inputs Fusion Logic). It uses a series of rules to decide which acceleration value to use as input to the vehicle dynamic's power-train. A more detailed description of the rules is provided by Papelis and Ahmad [9]. For the examples presented here, it is enough to consider the simpler rule that is often used, which is using the most conservative (lowest value) acceleration. For example, if the intersection navigation logic provides a lower deceleration than follow (maybe because a red light necessitates that the vehicle come to a stop), then the follow algorithm output is ignored.
The follow algorithm only decelerates the following vehicle. Catching up to a lead vehicle only occurs if the natural speed selection of the follower is higher than that of the lead vehicle. Once the following distance or approach rate become such that the follow algorithm engages, it then provides a "suggested" acceleration that is utilized only when it is lower than the free driving acceleration.

Performance Examples

Examples are provided to illustrate the operation of the controller under various driving situations. In each example, performance is illustrated through time series plots of lead vehicle velocity, follower velocity, desired following distance, and actual following distance. Vehicles in the examples can brake at a maximum deceleration rate of 0.68G (typical vehicle capabilities) and utilize a reaction delay of 0.8 seconds (relatively alert driver).

Baseline Performance

Figure 4 shows the response of the follow controller to a smooth decrease followed by a smooth increase in the lead vehicle's velocity. The desired FV is set to 65 mph (104.6 km/hr), while the initial LVV is set to 50 mph (80.5 km/hr). This ensures that for the first part of the maneuver the velocity of the follower is controlled entirely by the follow algorithm. The lead vehicle uses a rate of 0.5 m/sec/sec to decelerate to 43 mph (69.2 km/hr). After a short duration at that speed, the LVV increases to 67 mph (107.8 km/hr) at a rate of 0.3 m/sec/sec. The acceleration rates are typical of drivers who do not feel pressed to make a speed change [3]. Note that at approximately the 1500th simulation step, the lead vehicle exceeds 65 mph, in which case the follow algorithm would be disengaged.

Figure 4: Baseline performance.
Emergency Brake

Figure 5 illustrates the behavior of the controller when confronted with a lead vehicle performing maximum braking.

Despite the reaction delay, which affects distance tracking, the follow vehicle comes to a complete stop at the required stop distance. The minimum desired following distance is clipped to 22 feet to compensate for the length of the vehicles along with some minimum spacing. Figure 6 illustrates the same maneuver but at a 0.4 sec. following distance. In that case, the reaction delay prevents the controller from stopping the vehicle on time causing a collision.

Stop-And-Go

Figure 7 illustrates the behavior of the controller when following a lead vehicle traveling at 20 mph (32.2 km/hr), which then comes to a complete stop, and then accelerates back up to 20 mph (32.2 km/hr). A typical [3] rate of 0.3 G is used for both deceleration and acceleration. Note that at approximately the 300th simulation step, the rate of descent of the follow velocity is interrupted and remains at about 2.4 mi/hr until the 360th simulation step. This is due to the effect of the Low Speed Boost component of the controller.

Figure 5: Emergency braking at 2.0 sec following distance.
Figure 6: Emergency braking at 0.4 sec following distance.

Figure 7: Stop and go maneuver.

**Conclusion**

A car-following algorithm was developed for use in microscopic traffic simulation models utilized within a driving simulator's virtual environment. Due to its intended use, the algorithm must accommodate several practical situations that are not handled acceptably by typical car-following models. Examples of the performance of the algorithm were provided for typical driving situations.
References


