Traffic Mitigation in Mixed-Autonomy via V2V-Enhanced Intelligent Driving Models

Travis Miller
travisamm@berkeley.edu
Electrical Engineering Computer Science
University of California, Berkeley

Lawrence Rhee
llancerhee@berkeley.edu
Electrical Engineering Computer Science
University of California, Berkeley

Rayyan Saiyed
rayyan.saiyed09@berkeley.edu
Electrical Engineering Computer Science
University of California, Berkeley

Bazil Ahmad
bazilahmad@berkeley.edu
Electrical Engineering Computer Science
University of California, Berkeley

Abstract

This project investigates the use of Vehicle-to-Vehicle (V2V) communication to mitigate stop-and-go traffic in mixed-autonomy settings. While initial efforts explored reinforcement learning–based policies in simulation, practical sim-to-real limitations motivated a transition to a physics-based, V2V-enhanced controller designed for reliable execution on embedded hardware. The system is implemented as a cyber-physical platform comprising RP2040-based robots, ESP8266 WiFi communication, and Time-of-Flight sensing. Physical experiments demonstrate that V2V-assisted braking preserves throughput while reducing downstream traffic oscillations, validating the effectiveness of connectivity-enabled control for traffic smoothing on real hardware.

1 Introduction

Stop-and-go traffic waves ("phantom jams") arise from small perturbations in human driving behavior and significantly reduce traffic efficiency. Prior work has shown that a small number of Connected and Automated Vehicles (CAVs) can dampen these waves by reacting earlier and more consistently than human drivers. This project investigates how such traffic smoothing can be achieved on real embedded hardware in a mixed-autonomy setting.

The final control policy combines a "perfect" Intelligent Driving Model (IDM) with an asymmetric V2V braking mechanism that propagates deceleration signals downstream while preserving local safety constraints. Physical experiments demonstrate that V2V-assisted control maintains throughput while reducing downstream traffic oscillations, validating connectivity-enabled traffic smoothing on real hardware.

2 Hardware Implementation

2.0.1 Pololu 3pi+ 2040. We chose the provided Pololu as our vehicle for this project as we were able to use many of its onboard features. First we take advantage of the five down-facing infrared reflectance sensors to provide continuous lateral deviation data, which drives the feedback loop for line tracking (described in 4.2.1). Furthermore, to manage discrete navigation states, the robot utilizes an on-board gyroscope. By integrating the yaw rate, the system calculates relative heading changes in real-time. This inertial data is specifically employed to detect the completion of high-curvature turns, acting as a trigger to revert the control logic from a 'Cornering' state to a 'Driving Straight' state.

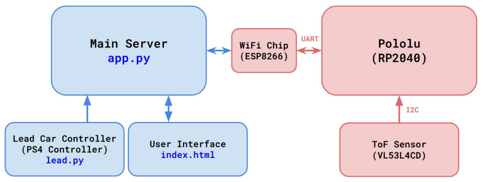

Figure 1: High-level system architecture

2.0.2 ESP8266 WiFi module. This module is used as the middle layer to wirelessly connect the Pololu to the central server with 2.4GHz bandwidth. To run the server, we used our Macbook to broadcast a local network. Upon boot of the module, it searches for this specific network. Once connected, the module uses the created client to read incoming data from the server and stores the data in a buffer. After message reception, it sends the buffer to the Pololu via UART over a 115200 baud rate (**Class Concept #1 - I/O Hardware**). This is accomplished by sending the buffer through the module's serial port.

2.0.3 VL53L4CD Time-of-Flight (ToF) sensor. This sensor is used to read the distance of the vehicle in front. As with many ToF sensors, this module emits an infrared light and tracks how long to receive the light to determine distance. An interesting bug we had was that the sensor initially was placed behind cardboard, which made the readings inaccurate due to its dark color. By taping white paper over the reflected part, we increased the amount of reflected light and drastically improved accuracy. Empirically we determined that the accuracy ranges from 7mm to 1000mm and a bias of +15mm (**Class Concept #2 - Sensors**). Furthermore, the sensor exposes an I2C (clock and data) interface, which the WiFi module reads and relays to the server (**Class Concept #1 - I/O Hardware**).

3 Software Architecture

The system consists of three interconnected components that enable real-time control and monitoring.

3.1 Main Server

3.1.1 Multithreaded Design. (**Class Concept #3 - Multithreading)** The server uses a multi-threaded architecture where each connected robot receives a dedicated handler thread. Upon connection, the server assigns a unique bot ID, creates an entry in the global robots dictionary, and spawns a background thread that continuously receives messages from that robot. This allows independent, non-blocking communication for each robot.

The global robots dictionary stores each robot's current data and requires lock-protected access due to concurrent access from multiple threads. Each handler thread is responsible for updating the robot's speed, headway distance, and timestamp in the shared dictionary.

3.1.2 Recording and Playback System. The recording phase begins from the UI. The server launches a background thread with a 60-second timer and begins to capture the controller's inputs, writing the vehicles desired velocity alongside a timestamp.

When the system begins playback, the server reconstructs the recorded trajectory by reading the file and issuing commands to the lead car.

3.1.3 Autonomous Control System. A third background thread runs to determine desired vehicle speeds from the actual velocities. The thread first acquires a lock briefly to create a snapshot of all robots and their current velocities, then releases the lock to perform calculations without blocking other threads.

The control system calculates target speeds using either the human driver or autonomous vehicle algorithm depending on the robot's type (discussed in section 4).

3.2 Controller Server

The controller server provides a human-operated interface for commanding the lead vehicle using a PlayStation 4 game controller. The system translates physical button presses ('X' for acceleration, 'O' for deceleration) into velocity commands that are transmitted to the main server's dedicated controller port (8081).

3.3 Web User Interface

To provide granular control and real-time visualization of the cyber-physical system, we developed a Web UI hosted on the central server. Its main purpose is to initiate recording/playback and display data.

4 Methodology and Algorithms

4.1 Autonomous Vehicle Policy Design

4.1.1 Preliminary Considerations: Reinforcement Learning. Our initial AV policy formulation adapted the reinforcement learning approach of Jiang et al. [1] for robot-scale deployment. While the original work employed DDPG with continuous actions, we implemented a Deep Q-Network (DQN) with ϵ-greedy exploration over a discrete acceleration set ($\mathcal{A}$) chosen to match the physical acceleration limits of the Pololu robots. The policy observes a 7-dimensional state vector derived from local vehicle information and outputs a desired velocity (using the element chosen from $\mathcal{A}$).

The reward function followed the core structure of Jiang et al., combining headway safety, speed regulation, approach-rate penalization, and acceleration smoothness, with additional TTC and distance shaping terms for robot-scale collision avoidance. Reward weights were rebalanced for hardware dynamics, and redundant components were consolidated in the final implementation.

Although the policy achieved near zero-collision behavior in simulation, it did not transfer reliably to physical hardware. SUMO simulations do not scale cleanly to robot-scale dynamics, particularly with respect to sensing noise, actuation delay, and contact interactions. While the trained policy was deployed to Pololu robots, its aggressive behavior (safe in simulation) led to collisions during real-world testing. Limited access to a full robot fleet further constrained large-scale hardware validation. These limitations—along with the high training cost (time & compute) and inference latency of RL-based policies—motivated a transition to a deterministic, V2V-based control policy designed for robust execution on embedded hardware.

4.1.2 Final V2V-Based Control Policy. The deployed AV policy is a deterministic controller that combines a locally computed "perfect" Intelligent Driving Model (IDM) update with a vehicle-to-vehicle (V2V) braking signal from the nearest upstream AV. Each AV i receives local telemetry—its own velocity v_i and Time-of-Flight headway s_i to vehicle i–1—and computes a nominal IDM acceleration a_{idm}. This acceleration is filtered by a safety layer enforcing minimum-gap, time-to-collision, and bounded deceleration constraints to produce a safe acceleration a_{safe}. We then pass this into:

$$v_{\text{idm}} = \text{clip}(v_{\text{base}} + a_{\text{safe}}\Delta t,\ 0,\ v_{\max})$$

to produce a local candidate velocity.

AVs are organized into an AV-only daisy chain based on mode flags. Lead AVs (or AVs without an upstream AV) operate purely under local "perfect"-IDM control. Trailing AVs incorporate V2V information using an **asymmetric trust rule**: if the upstream AV is accelerating or maintaining speed, the downstream AV ignores V2V input; if the upstream AV is braking, the downstream AV blends its local IDM velocity with the communicated upstream velocity:

$$v_{\text{final}} = \text{clip}\big(\beta\, v_{\text{idm}} + (1-\beta)\, v_{\text{up}},\ 0,\ v_{\max}\big),$$

where v_{up} is the upstream AV's target velocity, β is a fixed trust factor, and v_{idm} (v_{local} in the FSM) is the local IDM velocity.

4.2 Vehicle-to-Vehicle Communication Algorithm

4.2.1 Deterministic IDM Controller. Local longitudinal control for both AVs and human-driven vehicles is governed by the Intelligent Driving Model (IDM) [2]:

$$\dot{v}_\alpha = a\left[1 - \left(\frac{v_\alpha}{v_0}\right)^\delta - \left(\frac{s^*(v_\alpha, \Delta v_\alpha)}{s_\alpha}\right)^2\right],$$

- v_α is the velocity of vehicle α;
- s is the bumper-to-bumper distance to the next vehicle;
- Δv is $v_{\alpha+1} - v_\alpha$;
- v_0 is desired velocity of the vehicle;
- s_0 is the minimum gap to the next vehicle;
- T is the minimum possible time to the next vehicle;
- a is the maximum vehicle acceleration;
- b is the target deceleration rate;
- $s^*(v_\alpha, \Delta v_\alpha)$ is the dynamic desired gap given by

$$s^*(v_\alpha, \Delta v_\alpha) = s_0 + v_\alpha T + \frac{v_\alpha \Delta v_\alpha}{2\sqrt{ab}};$$

AVs use "perfect" IDM with $\delta = 4$ [2], producing smoother, more anticipatory behavior. Human drivers use $\delta = 5$ with less conservative parameters (i.e. larger a and b) and stochastic perturbations for realistic variability.

4.2.2 Event-Based FSM and Asymmetric Trust Logic. (**Class Concept #4 – FSM**) Figure 2 shows the finite state machine (FSM) governing V2V control. AVs default to a *Local* state, where velocity commands are generated solely from local IDM control. Upon detecting upstream AV deceleration, downstream AVs transition to a *Global* state, in which the local IDM velocity is blended with velocity from the nearest upstream AV. Lead AVs remain in the Local state at all times. This FSM enables early braking propagation while preserving local safety constraints and preventing traffic buildup. The demonstration used upstream target velocity as the propagated signal; acceleration-based propagation is discussed as a future improvement.

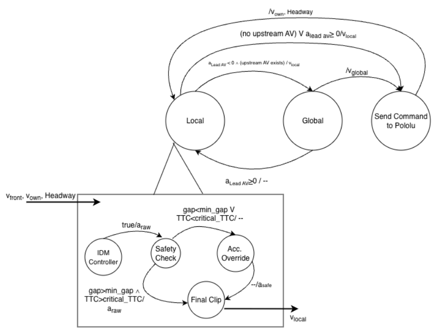

Figure 2: Event-based FSM for AV V2V control.

4.3 Human Driver Modeling

Human-driven vehicles are modeled using a stochastic "human"-IDM controller executed only for human robots. At each control step, the controller computes IDM acceleration using the measured ego speed, the measured headway to the vehicle immediately ahead, and that relative velocity to that vehicle. To introduce realistic variability, we apply an Ornstein–Uhlenbeck (OU) process [3] to a multiplicative speed factor, yielding a temporally correlated perturbation of the effective desired speed:

$$v_0^{\text{eff}} = v_0 \cdot \eta_t,$$

where v_0 is the IDM desired (free-flow) speed and η_t follows OU dynamics and is clipped to a bounded range. The IDM acceleration is then computed using v_0^{eff} with a human parameterization, producing more aggressive speed convergence and less anticipatory behavior than the AV model.

4.4 Line Tracking Feedback Control

(**Class Concept #5 - Feedback**) To facilitate our physical demonstration using Pololu robots, we developed a line-tracking algorithm to navigate an oval track. We leveraged the on-board down-facing IR sensors (described in section 2.1.1), designing a track with line width matching the central three sensors s1, s2, and s3 (sensors are indexed s0 through s4, left to right). Using these sensors, we defined 5 position states: left, slightly left, centered, slightly right, and right (see figure 3). Each state is assigned an error value (centered = 0, left states positive, right states negative), which drives a PD (Proportional-Derivative) controller:

$$u = K_p \cdot e + K_d \cdot (e - \text{last\_error})$$

where u is our control value, K_p controls sensitivity, and K_d provides damping to prevent oscillations. The control value is processed into a correction term: correction $= 1 - |u|$. This correction is multiplied to our base speed (received from the main server over WiFi) and applied to the appropriate wheel based on the error sign, effectively slowing one side to execute the turn. This maintains the desired speed while staying on track. To handle both straight portions and sharp turns, we employ separate K_p and K_d values for two operating modes: corner and straight. After calibration (necessary for the IR sensors), we default to straight mode. Mode switching to corner is triggered by "brake markers" before each turn that activate both outer sensors (s0 and s4) simultaneously. The return to straight mode uses the internal gyroscope, transitioning after an 85-degree rotation (matching our oval track's uniform turn angle).

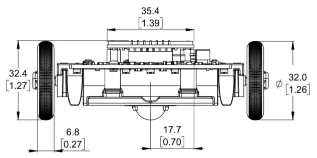

Figure 3: IR sensor layout on Pololu 3pi+ robot

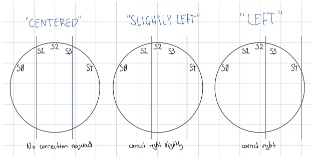

Figure 4: Line detection states based on sensor readings

5 Evaluation

We evaluated the system's capabilities by deploying our V2V policy on the Pololu robots on a physical track, comparing a control run (a leader followed by several "human" drivers) to a test run (a leader followed by a mix of "AV" and "human" drivers).

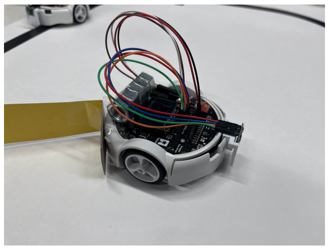

Figure 5: Image of Pololu with ToF sensor in front [right], WiFi mod in middle, and a paper trail at back [left]

5.1 Metric

5.1.1 Jerk. Our primary metric to evaluate traffic smoothness was jerk. We compute jerk (defined as the average squared acceleration over time) by first calculating acceleration: $a_i = \frac{v_{i+1} - v_i}{\Delta t}$, where Δt is the sampling interval. The jerk metric is then:

$$J = \frac{1}{T} \sum_{i=1}^{T} a_i^2$$

where T is the total number of samples. Lower values indicate smooth speeds and reduced acceleration/braking, while higher values would reflect the opposite, implying an increase of stop-and-go waves.

5.1.2 Average Velocity. The second metric for evaluation was taking the overall average of the entire system. This ensures that the system does not "cheat" by simply slowing down all vehicles to a constant velocity to minimize jerk.

5.2 Results

Across follower vehicles (Robots #1–#3), the playback phase maintained a comparable average speed to the manual baseline (0.264 m/s vs. 0.282 m/s), indicating that the V2V policy did not reduce jerk by trivially slowing the fleet. Robot #0 (the manually controlled lead vehicle) is excluded from analysis, as its behavior is identical across phases and does not reflect V2V effects. Jerk reductions were vehicle-dependent: the downstream AV (Robot #3) exhibited a substantial decrease in jerk during playback.

Because Robots #1 and #3 operated in AV mode, the pronounced jerk reduction at Robot #3 demonstrates effective downstream shockwave attenuation. Once braking disturbances are dampened by an AV, vehicles behind that AV experience smoother trajectories even if earlier vehicles do not.

Table 1: Jerk comparison for AV vehicles only

AV Robot	Jerk (Rec.) - m/s^3	Jerk (Playback) - m/s^3
Robot #1	3739.1	4169.2
Robot #3	4163.0	2390.2

6 Future Implications

The work completed here serves as a foundational proof-of-concept for low-cost, decentralized Cooperative Adaptive Cruise Control (CACC). By demonstrating that even a simple V2V-augmented control law can dampen shockwaves in a mixed-autonomy setting, we open several avenues for future research in Intelligent Transportation Systems (ITS).

6.1 Scalability to CACC Platoons

Currently, our system utilizes a "Leader-Follower" communication topology where the AV anticipates the actions of the lead vehicle. A significant future extension would be the implementation of a fully connected mesh network, allowing for true CACC. In this scenario, every vehicle in the chain—not just the designated AV—would share acceleration data. This would enable the transition from simple "shock absorption" (dampening a wave) to "wave prevention" (simultaneous acceleration/deceleration of the entire fleet).

6.2 Complex Geometric Topologies

While our PID line tracker and IDM successfully manage a single-lane oval track, real-world deployment requires handling complex geometries such as highway merging ramps and signalized intersections. Future work could integrate the V2V messaging stack with a high-level state machine to handle lateral negotiation. For instance, an AV approaching a merge point could broadcast a "yield request," allowing upstream vehicles to cooperatively open a gap, thereby solving the "zipper merge" inefficiency that plagues human traffic today.

6.3 Acceleration-Based V2V Propagation

In the physical deployment, downstream AVs incorporated V2V information using the upstream AV's target velocity. This introduces an edge case when the upstream AV is braking but still traveling faster than the downstream AV. In this scenario, blending with the upstream velocity can yield a v_{final} that exceeds the downstream AV's locally safe IDM velocity, potentially causing unsafe acceleration.

Using acceleration-based V2V propagation avoids this issue. If the upstream AV is braking, $a_{\text{up}} < 0$, and the propagated velocity update

$$v_{\text{final}} = v_{\text{local}} + a_{\text{up}}\Delta t$$

is guaranteed to satisfy $v_{\text{v2v}} \leq v_{\text{idm}}$. This enforces monotonic braking and prevents velocity overshoot while preserving early braking propagation. For this reason, acceleration-based V2V signaling is a safer and more principled formulation and is identified as a key improvement for future implementations.

7 Conclusion

We successfully designed and implemented a physical traffic mitigation system using a Vehicle-to-Vehicle (V2V) enhanced control strategy. By extending the standard Intelligent Driver Model (IDM) with an asymmetric trust mechanism and integrating a robust embedded sensing stack, we demonstrated that a single Autonomous Vehicle can effectively dampen stop-and-go waves on real hardware.

8 Acknowledgments

We extend our sincere gratitude to the other student groups who generously lent us their Pololu robots. Without their support, the multi-agent validation of our control policies would not have been possible.

References

[1] L. Jiang, Y. Xie, X. Wen, D. Chen, T. Li and N. G. Evans, "Dampen the Stop-and-Go Traffic with Connected and Automated Vehicles – A Deep Reinforcement Learning Approach," in *2021 7th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, IEEE, 2021, pp. 1–7.
[2] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical Review E*, vol. 62, no. 2, pp. 1805–1824, 2000.
[3] G. E. Uhlenbeck and L. S. Ornstein, "On the Theory of the Brownian Motion," *Physical Review*, vol. 36, no. 5, pp. 823–841, 1930.