A Safety Critical Automatic Emergency Braking POMDP Planner

Hasan Tafish
Stanford University
Stanford, CA
htafish@stanford.edu

***Abstract*—This project develops a Partially Observable Markov Decision Process (POMDP) planner capable of performing vehicular Automatic Emergency Braking (AEB) to avoid forward collisions with other vehicles. The proposed system handles state uncertainty and noisy sensor measurements through belief state planning and is designed with explainability and validation in mind, two crucial elements in safety critical systems. The system is evaluated on a number of vehicle-to-vehicle collision simulation scenarios to evaluate its safety and the level of discomfort passengers may experience when the system is activated.**

I. Introduction

In 2025, the National Highway Traffic Safety Administration (NHTSA) finalized a rule (49 CFR) [1] mandating that (beginning in 2029) all new vehicles sold in the United States be equipped with an automatic emergency braking (AEB) system. AEB is a system that warns the driver about a possible collision and automatically applies the brakes when it detects an imminent collision. The system escalates its response from a warning to soft brake, and then strong brake depending on the severity of a possible collision. AEB systems are not particularly new, Several passenger cars have been equipped with some AEB capability since the early 2000s with each manufacturer taking liberty in defining the specifications and performance metrics of their proprietary systems, however, the recent NHTSA ruling clearly defines the minimum requirements and performance metrics against which commercial AEB systems need to be evaluated prior to commercial release. This regulation forms an opportunity to develop a universal AEB system that has the potential to work with different vehicle platforms, and sensor suites, and varying sources of uncertainty in measurements and dynamics.

In this project we propose to solve the optimal AEB policy using an offline POMDP planner. The resulting policy can then be integrated with a vehicle that is equipped with odometry sensors to measure its own motion and a forward facing sensor to detect other vehicles that pose a collision threat. The policy can then advise optimal braking actions ”Maintain speed”, ”Soft brake” or ”Strong brake” to avoid forward collision with another vehicle. Figure 1 shows an example of the scenarios handled by this system.

Section II summarized the related literature. Section III describes the problem formulation. Section IV goes through the experimental setup, and Section V summarizes the results.

II. Related Work

A. Regulatory and scenario-based AEB evaluation

A large body of AEB research is organized around repeatable, scenario-based testing that links performance to relative kinematics such as longitudinal gap and closing speed. NHTSA defines a set of 127 test cards [1] that AEB manufactures are required to show that their systems are capable of successfully avoiding collisions in all of the 127 scenarios. Regulations like FMVSS 127 provide a standardized frameworks to the evaluation of AEB systems. These scenarios can be replicated in simulation and real world experiments as a way to develop and validate AEB systems

B. Rule-based AEB and kinematic triggers

Classical AEB algorithms commonly rely on interpretable kinematic measures—especially time-to-collision (TTC)—to determine when to issue warnings and apply staged braking. Empirical and modeling studies show that TTC thresholds and deceleration profiles strongly influence crash mitigation outcomes, and that earlier or more conservative triggers can increase safety benefits at the cost of nuisance interventions [2]. Recent work continues to refine TTC-based logic by blending professional-driver-inspired braking patterns with thresholding schemes, emphasizing explainability and calibration to human-like behavior [3]. While these methods have seen success in commercially deployed systems, they often require complex rule-based programming and substantial engineering effort to ensure the safety and comfort of these systems. Moreover, most of these systems fail to directly consider uncertainty in the system in any formal rigorous method, and hence, they are sensitive to changes in noise characteristics of the systems underneath.

C. Learning-based longitudinal control and AEB/ACC integration

Reinforcement learning has increasingly been applied to longitudinal control, often unifying adaptive cruise control (ACC) with collision-avoidance/AEB behaviors in a single policy. Early RL work demonstrated the feasibility of learning smooth longitudinal behaviors in interactive traffic settings [4] More recent deep RL studies propose safety-focused ACC variants and integrated ACC–AEB controllers that better handle complex multi-vehicle interactions and hazardous edge cases compared to purely rule-based systems [5]. These RL based methods move away from the rule based methods of the previous section, but they still neglect to consider uncertainty stemming from noisy measurements and vehicle dynamics.

D. Partial observability, belief-state estimation, and hybrid approaches

Studies that model AEB using POMDPs are rare. In [**?**], the authors formulate long-horizon driver warning generation as a POMDP, explicitly criticizing the rule based one-shot TTC thresholding for its lack of long future-horizon modeling of driver behavior, and traffic participants. The study focuses only on forward collision warnings. Their proposed action space, therefore, does not allow any automatic braking. The work in [7] is the most direct application of POMDPs to the collision avoidance problem. The authors of that study devise a POMDP planner that can intervene with varying levels of mild longitudinal deceleration. Their planner integrates with a TTC based AEB system to improve the overall collision avoidance performance of the AEB system specifically in scenarios where the collision target is a partially occluded pedestrian. While the work in [7] is the closest we found in literature to a POMDP-based AEB system, it focused only on collision with pedestrians, there's no treatment for vehicle-to-vehicle collision scenarios, and the proposed system works adjacent to an existing AEB system which still interferes as a last countermeasure against collision instead of redesigning the AEB system itself as a POMDP. The study also does not directly measure the level of passenger discomfort that the proposed system might introduce to the ride quality.

III. Problem Formulation

A. Scenario Modeling

Since all of the FMVSS 127 test cards describe straight road scenarios, we mainly focus on modeling and evaluating our system for these scenarios, but the algorithms described are generalizable to any road shape with the proper dynamics models. Figure 1 shows an example scenario used to develop and evaluate our AEB system.

B. Action Space

Generally, AEB systems can perform one of four maneuvers, namely: {Maintain speed, Warning, Soft brake, Strong brake} "Maintain speed“ is the default action, where the car does not warn the driver or apply any deceleration. "Warning“ sends an audio visual notification to alert the driver to a possible collision. "Soft brake“ applies a deceleration of -6 m/s^2 if the driver does not respond to the warning. "Strong brake“ applies a deceleration of -9 m/s^2 if the system determines that neither "Warning“ nor "Soft brake“ are sufficient to avoid the collision.

In our system, we only consider three actions {Maintain speed, Soft brake, Strong brake}. Extending the work to include "Warning“ requires a proper driver response model which is outside of the scope of this project.

C. State Space

The chosen state space is the relative distance between the target and the ego vehicle x_t, the velocity of the target v_t, the velocity of the ego vehicle v_e, and the current acceleration of the ego vehicle a_e.

$$S = [x_t, v_t, v_e, a_e]^T \tag{1}$$

To make POMDP planning computationally tractable, we discretize the state using a uniform 4D grid. x_t is discritized into 50 bins that range from 0 to 100 m, v_t and v_e are discretized into 20 bins that range from 0 to 21 m/s, a_e ranges from -9 to 0 m/s^2 and is discretized into 10 bins. We assume that the ego vehicle doesn't accelerate in our scenarios, and therefore only decelerating maneuvers are modeled. This discretization results in a total of 220500 states.

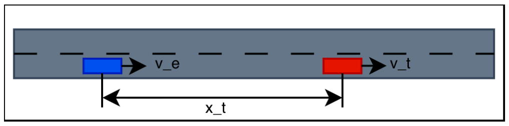

Fig. 1. A vehicle to vehicle collision scenario where the ego vehicle (shown in blue) moves at a speed v_e and acceleration a_e, and the target vehicle (shown in red) is x_t meters ahead of the ego vehicle and moves at a speed of v_t

D. Observations

We assume that the vehicle is equipped with noisy motion sensors that can provide an estimate of its velocity m_{v_e}, acceleration m_{a_e}, and a forward facing camera that provides an estimate of the target distance m_{x_t}

$$o = [m_{x_t}, m_{v_e}, m_{a_e}]^T \tag{2}$$

We assume that the observation contains an additive normally distributed noise $e \sim N(0, R)$ where R is the 3×3 measurement noise covariance.

E. State Estimation

To estimate the belief over the State s given sequential noisy observations o we use a Kalman Filter with the following dynamics model

$$\hat{s}[k] = As[k-1] + \epsilon \tag{3}$$

where $s[k-1]$ is the state estimate from the previous step, $\hat{s}[k]$ is the state after performing the dynamics propagation. A is the dynamics transition matrix, and $epsilon$ is the process model error which is assumed to be normally distributed $\epsilon \sim N(0, Q)$, where Q is the 4×4 process noise covariance. We assume constant acceleration dynamics for the ego vehicle and constant velocity dynamics for the target as shown in (4) (we omit the step index k and $k-1$ for brevity):

$$\begin{bmatrix} \hat{x_t} \\ \hat{v_t} \\ \hat{v_e} \\ \hat{a_e} \end{bmatrix} = \begin{bmatrix} x_t + v_t\Delta t - v_e * \Delta t - 0.5 a_e \Delta t \\ v_t \\ v_e + a_e \Delta t \\ a_e \end{bmatrix} + N(0, Q) \tag{4}$$

The measurement model is

$$z[k] = H\hat{s}[k] + e \tag{5}$$

Where e is the normally distributed measurement noise $e \sim N(0, R)$ and H is the measurement matrix:

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{6}$$

F. QMDP Planning

1) QMDP background: The QMDP method solves the problem under the assumption that the state becomes fully observable after one time step. With this assumption, we use the value iteration algorithm to solve for the optimal state-action utility function $Q(s, a)$ assuming full observability. The optimal utility function over belief state is then obtained by $Q(b, a) = \sum_s b(s) * Q(s, a)$ where $b(s)$ is the belief vector over the state space, which can be obtained from the Kalman Filter. Given a state belief b we can extract the optimal action a^* that maximizes $Q(b, a)$

2) Transition model $T(s' \mid s, a)$: The transition model encodes the probability of transitioning from state s to state s' if action a is taken. The transition model can be encoded as a number $|A|$ of $|S| \times |S|$ matrices. Where $|A|$ is the size of the action space. We use the dynamics model described in Section III.E to compute $T(s' \mid s, a)$ and store it as 3 $|S| \times |S|$ matrices, one for each action in our action space. Storing the dense $|S| \times |S|$ matrices is space inefficient, as each matrix will consume around 181GB of memory if 32-bit floating point representation is used. This amounts to a total of 3*181=543GB for the whole transition model. Instead, we utilize the fact that the probability of transitioning to a state s' from any other state s is equal to zero for the majority of the 220500 states. This allows storing the transition model as a compressed sparse row matrix which dramatically reduces the space requirements and speeds up the value iteration as the dense matrix - dense vector multiplication $\sum_{s'} T(s'|s, a) * U(s')$ in Bellman backup is replaced by a sparse matrix - dense vector multiplication.

The grid discretization (described in Section III.C) might lead to the problem of aliasing if the cell size is too large. Aliasing happens when we lump many slightly different real driving situations into the same discrete state, even though those situations would lead to different outcomes after an action. A simple discretized model that uses one representative point per state can therefore "pretend" the whole cell behaves the same way, giving misleading next-state predictions. One solution is to make sure that the cell size is small, but that would result in a massive total number of states and might render the problem intractable. We use Monte Carlo simulation to avoid this problem by sampling many points inside each cell, simulating them forward using the dynamics model, and counting where they land, so the transition becomes a probability distribution that reflects the variety of behaviors within the cell. This does not remove aliasing entirely, but it greatly reduces the bias caused by treating a whole region in the state space as a single point. Algorithm 1 describes the procedure to compute $T(s' \mid s, a)$ using Monte Carlo simulation.

Algorithm 1 Transition model computation using Monte Carlo

Inputs: Discrete state space grid D, continuous dynamics function $f(s, a)$, Number of samples per cell N, Actions set A

Outputs: Transition matrices $T(s' \mid s, a)$

Procedure:

```
For each action $a \in A$:
  For each state cell $s$ in the grid D:
    Sample N states from the cell corresponding to $s$
    For each sample:
      propagate one step using $s' = f(s, a)$
      Map $s'$ to a discrete cell in $D$
      Increment the count $C_a(s, s_i^{'}) = C_a(s, s_i^{'}) + 1$
    Normalize counts to obtain transition probability:
    $T(s_i' \mid s, a) = \frac{C_a(s, s_i')}{\sum_i C_a(s, s_i')}$
```

3) Rewards: The simulation is terminated in three cases: 1) the ego vehicle applies the brakes and reaches a complete stop before a collision, 2) the ego vehicle collides with the target vehicle, and 3) the simulation time ends. A collision is the worst outcome; therefore, we severely penalize a collision state by a value proportional to the relative speed at collision:

$$collision\ penalty = -2000 - 50(v_t - v_e) \tag{7}$$

To encourage the planner to be conservative in applying the brakes we penalize "Soft brake" and "Strong brake" actions proportionally to the time to collision at the state they are applied to. This has the effect of encouraging the planner to bias braking decisions towards states that correspond to smaller TTC, and therefore delay applying the brakes until the threat is more severe.

$$Soft\ brake\ penalty = -10 * min(ttc, 10) \tag{8}$$

$$Strong\ brake\ penalty = -20 * min(ttc, 10) \tag{9}$$

We saturate the TTC at 10 seconds in (8) and (9) to avoid overly large penalties. Finally, we add a small penalty that is proportional to the level of discomfort caused by the action a taken at any state s

$$discomfort\ penalty = -compute\_discomfort(s, a) \tag{10}$$

The discomfort computation is detailed in section IV.

IV. Experimental Setup

A. Simulation Scenarios

We use OpenAI's gym framework to create simulation scenarios based on a subset of the FMVSS 127 test cases. We create 34 scenarios of vehicle to vehicle collision. The scenarios vary by the initial velocities of the ego and target vehicles, whether the target vehicle is decelerating (initial a_t), and the initial distance between ego and target. Table I summarizes the initial simulation parameters for the different scenarios. The scenarios where the target is decelerating are meant to evaluate the capability of the system to avoid a collision when a high speed lead vehicle suddenly applies the brakes. To simulate noisy sensors, we add a normally distributed noise to the observation vector (eq 2) at each step. Table II lists the selected noise parameters. We assume that the noise in m_{x_t}, m_{v_e}, and m_{a_e} is not correlated (i.e. the observation noise covariance R is diagonal).

TABLE I
SCENARIO PARAMETERS

Target Mode	**Scenario Parameters Range**			
	initial v_e	***initial*** v_t	a_t	***initial*** x_t
Stationary	10 - 70 [km/h]	0 [km/h]	0 $[m/s^2]$	$(v_e - v_t) * 10$ [m]
Slow	40 - 70 [km/h]	20 [km/h]	0 $[m/s^2]$	$(v_e - v_t) * 10$ [m]
Decelerating	50 - 80 [km/h]	$v_t = v_e$	-0.5 - -0.3 $[m/s^2]$	12, 20, 30, 40 [m]

TABLE II
OBSERVATION NOISE PARAMETERS

Measurement	**Noise standard deviation**
Relative target distance m_{x_t}	0.707 [m]
Ego velocity m_{v_e}	0.44 [m/s]
Ego acceleration m_{a_e}	0.01 [m/s^2]

B. *Evaluation Metrics*

1) Safety Metrics: We measure the safety of the system by counting the number of times it failed to prevent a collision. We run each of the 34 scenarios 10 times, the randomized observation noise ensures diversity of outcome when running the same scenario several times. We then compute the probability of collision as the ratio between the number of simulation runs that ended in collision to the total number of simulations (i.e. 340).

$$P(collision) = \frac{Number\ collisions}{Number\ of\ simulations} \tag{11}$$

Another safety metric is the average collision velocity; this is important to track, since not all collisions are equal. The higher the relative velocity at the collision, the more severe the collision is.

$$\begin{gathered} average\ collision\ velocity = \\ \frac{\sum_i collision\ velocity\ in\ scenario\ [i]}{Number\ of\ collisions} \end{gathered} \tag{12}$$

C. *Discomfort Metrics*

It is important to quantify the level of discomfort the system may introduce to the rider experience. Repeated switching between high and low levels of deceleration might lead to a low quality ride. Instead, we want to design a system that makes the best effort to avoid collisions while maintaining a smooth trajectory. The discomfort experience in a single simulation step k is measured as a weighted sum of the ego deceleration at step k and the rate of deceleration change (i.e. jerk) between two consecutive steps.

$$\begin{aligned} discomfort[k] = w_0 * max(-a_e[k], 0.0)^2 + \\ w_1 * \frac{|a_e[k] - a_e[k-1]|}{\Delta t} \end{aligned} \tag{13}$$

$w0$ and $w1$ control how much we weigh the deceleration discomfort and the jerk discomfort respectively. The discomfort level over a single simulation run is the average step discomfort over the whole run.

D. *Baseline*

We compare our QMDP policy against a baseline algorithm that uses TTC thresholding to trigger "Soft brake" and "Strong brake" actions.

The baseline algorithm assumes that the state is directly observable (i.e. doesn't consider state uncertainty), and will execute a "Strong brake" if the time to collision is lower than 2 seconds, a "Soft brake" if it's higher than 2 seconds but lower than 4. Otherwise, the algorithms maintains the current speed. The TTC thresholds are chosen such that a collision can be prevented in the most aggressive scenario and assuming perfect knowledge of the state.

V. RESULTS AND DISCUSSION

A. *QMDP vs. Baseline*

Figure 2 shows the probability of collision for both our QMDP planner and the Baseline. The baseline algorithm fails to prevent a collision in 119/340 scenarios while QMDP planner fails to prevent a collision in 66/340 scenarios; a significant improvement in the safety of the system. This, however, comes at the cost of a slight increase in the discomfort level when using QMDP policy compared to the baseline, as shown in Figure 3. Both QMDP and baseline struggle in scenarios where the target is decelerating. The main reason of failure in these scenarios is the positive bias in estimating the relative velocity $v_t - v_e$ due to violating the constant target velocity assumptions made in our Kalman filter dynamics model. Figure 4 shows the bias in estimating the relative velocity and the corresponding deceleration profiles produced by Baseline, and QMDP. QMDP policy performs considerably better than baseline in these scenarios due to the fact that it takes the uncertainty of the state into consideration by picking the action that maximizes the utility of the belief state. This integration of the state uncertainty through belief state planning allows QMDP to account to the possibility that the probability of collision is larger than what the mean value of the kalman filter state alone is suggesting, and as a result it performs more aggressive braking than baseline to minimize the possibility of ending in the high cost collision state.

B. QMDP vs. Q-Learning

This project focused on developing a policy that is explainable and easy to validate in a safety critical system. Offline QMDP with a strong transition dynamics fits that requirement. However, we wanted to compare how our QMDP planner compares to the more recent Deep Q-learning networks, and explore the ability of these networks to learn hidden structure by simply interacting with the environment and observing the rewards. For that purpose, we trained a simple deep Q-network (DQN). The network architecture is shown in Table III.
The network input is a 8×1 vector that concatenates the state mean and the diagonal of the covariance matrix obtained after updating the Kalman filter with a new observation. We train the network by minimizing the mean squared error between the current estimate of $Q(s, a)$ and a target estimate: $r + \gamma max_{a'} Q(s', a') - Q(s, a)$ as described in [8] We use ϵ-greedy exploration strategy during training and then evaluate the resulting network on the same 34 scenarios. Surprisingly, the DQN performed worse than both QMDP and the baseline algorithm. Table IV shows that DQN's probability of collision is considerably higher than QMDP and the baseline. The discomfort level is low, but it becomes irrelevant when the sytem fails to prevent collisions most of the time. While more investigation is needed to diagnose this under performance, we hypothesize that including the belief state in Q-learning input leads to an exponential increase in the volume of the space that the network needs to explore (i.e. curse of dimensionality). Whereas a model that exploits the known structure of the problem (i.e. transition model) is capable of finding an optimal policy without having to explore an 8 dimensional input space.

TABLE III
THE DEEP Q-NETWORK ARCHITEVTURE

Input(8,64)
↓
Linear(64, 64)
↓
Relu
↓
Linear (64, 64)
↓
Relu
↓
Output(64, 3)

TABLE IV
DQN PERFORMANCE

P(collision)	Average Discomfort
0.48	4.12

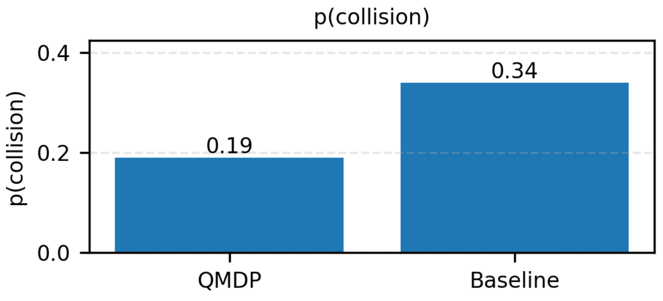

Fig. 2. The probability of a simulation scenario ending in a collision for QMDP and baseline policies

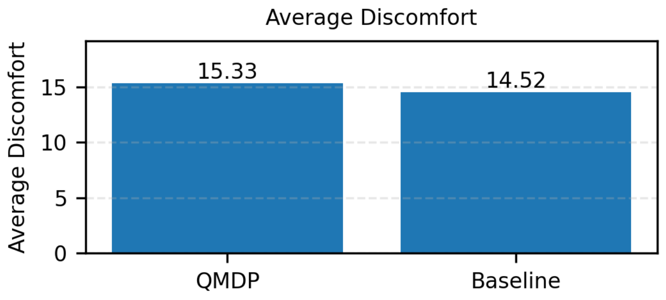

Fig. 3. The average discomfort level caused by QMDP and Baseline. Since the scenario duration might differ between QMDP and baseline, the average discomfort level is normalized by the scenario length

VI. NOTE TO REVIEWERS

The work done to meet the extra 30 hours requirement is summarized below:

- The Monte Carlo transition model algorithm explained in Section III.F, we started with a simple transition model that only used the center of each grid cell to compute the transition probabilities, when that did not perform optimally, we tried dramatically reducing the size of the cell, but that led to an explosion in the problem size. Finally, we decided to use Monte Carlo simulation to overcome aliasing while maintaining the size of the problem small. This took a considerable amount of debugging and diagnosing the model.
- Proposing a discomfort metric in Section IV.C and using it in the rewards to fine tune the QMDP planner to minimize discomfort while avoiding collision.
- The deep Q-network training and evaluation outlined in Section V.B.

ACKNOWLEDGMENT

We have utilized Open AI's ChatGPT 5.1 to aid in generating simulation scenarios, visualization, figures, and brainstorming.

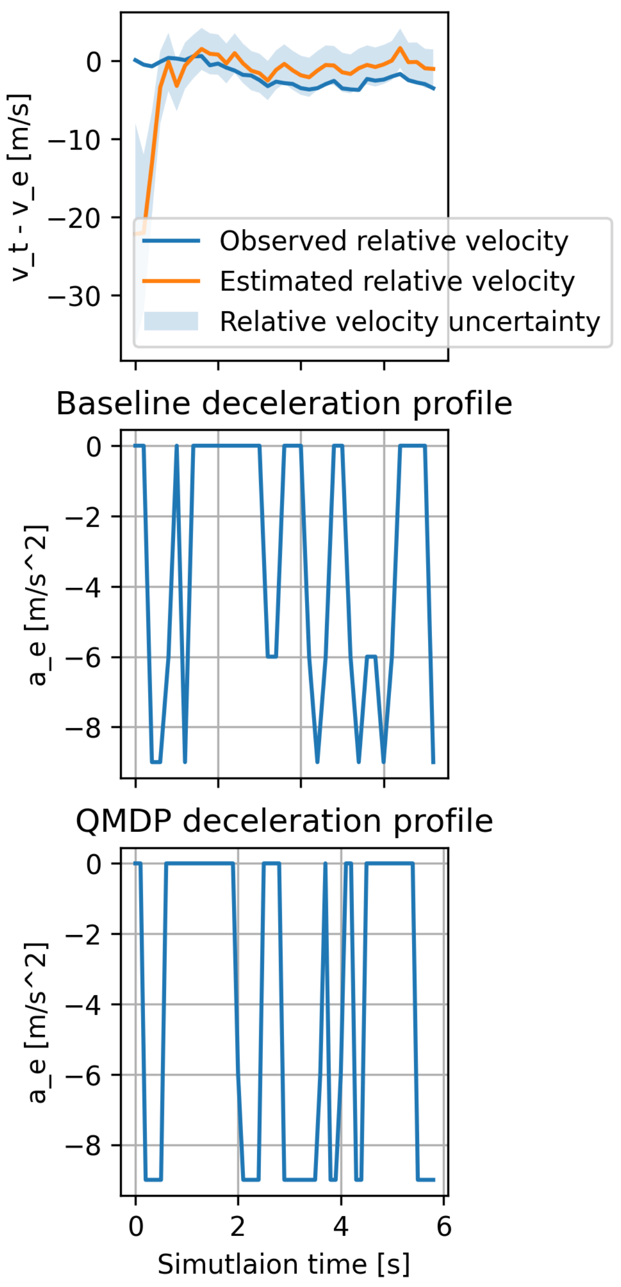

Fig. 4. The QMDP policy performs more aggressive braking than baseline to account for the error in estimating relative target velocity

References

[1] NHTSA 49 CFR. Federal motor vehicle safety standards; automatic emer- gency braking systems for light vehicles 49 cfr parts 571, 595, and 596. Technical report. URL: https://www.nhtsa.gov/sites/nhtsa.gov/files/2024-04/final-rule-automatic-emergency-braking-systems-light-vehicles_web-version.pdf .

[2] M. Abdel-Aty, Q. Cai, T. Wu, O. "Zheng Evaluation of automated emergency braking system's avoidance of pedestrian crashes at intersections under occluded conditions within a virtual simulator," in Accident Analysis and Prevention, Elsevier Science Direct, Volume 176, Oct 2022.

[3] F. Lai, J. Liu, Y. Hu, "An Automatic Emergency Braking Control Method for Improving Ride Comfort," World Electr. Veh. J. 2024, 15(6), 259.

[4] L. Ng, C. M. Clark, J. P. Huissoon, "Reinforcement Learning of Adaptive Longitudinal Vehicle Control for Dynamic Collaborative Driving, " 2008 IEEE Intelligent Vehicles Symposium

[5] R. Zhao, et al. "Adaptive Cruise Control Based on Safe Deep Reinforcement Learning," Sensors (Basel). 2024 Apr 22;24(8):2657. doi: 10.3390/s24082657

[6] C. Li, A. Xu, E. Sachdeva, et al. "Optimal Driver Warning Generation in Dynamic Driving Environment, "

[7] M. Schratter, M. Bouton, M. J. Kochenderfer, D. Watzenig "Pedestrian Collision Avoidance System for Scenarios with Occlusions," 2019 IEEE Intelligent Vehicles Symposium (IV), DOI:10.1109/IVS.2019.8814076

[8] Mykel J. Kochenderfer, Tim A. Wheeler, and Kyle H. Wray. "Algorithms for Decision Making," MIT Press, 2022