A POMDP Approach for Safety Assessment of Autonomous Cars

Ivan Ang[0000−0003−2732−5625] and Hanna Kurniawati[0000−0001−5053−7146]

School of Computing, Australian National University
{ivan.ang, hanna.kurniawati}@anu.edu.au

Abstract. This paper proposes a mechanism to automatically assess the safety of autonomous robots, and in particular autonomous cars. Most methods to assess the safety of autonomous cars generate adversary strategies, which will then be used to test whether the car being assessed can avoid accidents with the adversary. To generate such adversarial strategies, many have proposed learning techniques that require a large amount of accident data. But, such data are difficult to obtain because accidents are rare. To alleviate this issue, we leverage the observation that when safe and colliding adversary trajectories are closer together, the vehicle is less safe because there is generally less buffer to avoid accidents. Specifically, we generate/utilise data on adversaries' safe trajectories, which is more abundant than accidents data, and compute colliding adversarial trajectories that are as close as possible to the safe trajectories. The average distance between safe and colliding adversarial trajectories provides an indicator of the vehicles' safety. To compute colliding adversarial trajectories, we take into account that the driving strategy of the vehicle being assessed is not fully known, and therefore propose a multi-objecive POMDP framing of the problem and an on-line planning method, called Constraint-Aware Tree (CAT), to compute approximate solution to the multi-objective POMDP. Evaluations of four learning-based autonomous driving software on pedestrian crossing and lane merging scenarios, derived from the National Highway Traffic Safety Administration (NHTSA), indicate the viability of the proposed testing mechanism in assessing a variety of autonomy software. Moreover, evaluations of CAT on the nuScenes dataset indicate that CAT generates more colliding adversarial trajectories in less time compared to state-of-the-art learning-based method, STRIVE.

Keywords: Autonomous Cars · Motion Planning · Partially Observable Markov Decision Process

1 Introduction

As autonomous robots start to become consumer products, their safety assessment becomes increasingly important. This paper focuses on a type of safety assessment which aims to rank the safety of different autonomous robots. Specifically, given a set of multiple autonomous robots with similar hardware capabilities but different autonomy software, our goal is to rank the safety of these autonomous robots under a given testing scenario. The algorithms that pilot these robots may not be known a priori, but we assume that the safety assessment mechanism is provided with a simulator that can be used to probe the robots' behaviors, albeit with some inaccuracies and errors. The mechanism and method we propose can be applied to assess the safety of various types of robots, but in this paper, we focus on the safety assessment of autonomous cars.

The car industry has a well-established safety assessment, namely the New Car Assessment Program (NCAP) rating system [28]. This rating system has originally focused on the physical structure of the vehicle, such as the famous crash test. NCAP has started to expand their testing to accommodate autonomous technologies such as autonomous emergency braking, lane support systems, automatic emergency steering and speed assistance systems [31]. These NCAP testing mechanisms do not require the inner working of the cars or its autonomous technologies to be disclosed, which is suitable to assess the safety of autonomous systems with potentially many black-box components. However, their testing rely on hand-crafted adversary strategies that are not adaptable to the behaviors of the cars being assessed [30]. For instance, in testing pedestrian avoidance capability, a mock-up pedestrian acts as an adversary to the vehicle being tested, but its behavior in crossing the street is generated without consideration of the behavior of the vehicle being assessed. As a result, they are often insufficient to test many autonomous car technologies with Level 3 Autonomy and beyond. Methods that generate adaptable adversary strategies for testing the safety of autonomous cars have been proposed. Most rely on machine learning techniques that require a large amount of data (e.g., [34, 48]). However, large datasets for such training are generally difficult to obtain because accidents, and more importantly catastrophic accidents, are rare, which is fortunate for the users, but lead to limited accident data for safety assessment [21]. Moreover, the long training time tend to be prohibitive in performing frequent assessment, such as every time software update is performed.

To alleviate the above difficulty, in this paper, we propose an adaptive safety assessment mechanism based on the observation that autonomous cars with closer safe and colliding adversaries' trajectories tend to be less safe, because there is less buffer to avoid catastrophic accidents. The mechanism starts by generating/utilizing data on adversaries' safe trajectories, which is much more abundant than accident data. For each such trajectory, the mechanism computes colliding adversarial trajectories that are as close as possible to the safe trajectory. The average Fréchet distance between the safe trajectories and their associated colliding trajectories indicates the relative safety of the assessed autonomous car, where smaller average distance indicates less safe vehicles.

Core to the above safety assessment mechanism is an efficient method to compute colliding adversarial trajectories that are as close as possible to a given safe trajectory. Such a method must take into account that the driving strategy of the vehicle being assessed is not exactly known. Therefore, computing colliding adversary trajectories is essentially a Partially Observable Markov Decision Process (POMDP) problem. To compute colliding trajectories that are also as close as possible to a given safe trajectory, we explicitly frame the problem as a multi-objective POMDP problem and develop a novel online planning method, called Constraint-Aware Tree (CAT), to approximately solve such a POMDP.

To evaluate CAT's performance, we compare CAT with STRIVE [34], a state-of-the-art learning-based methods to generate collision trajectories on complex vehicle-vehicle traffic scenarios, on the nuScenes dataset [5]. The results indicate that our proposed POMDP-based method is able to generate significantly more collision trajectories, while abiding the traffic rules and uses much less computational time. We also benchmarked top-rated autonomous driving software submitted to the CARLA Driving Leaderboard [12] to validate our proposed safety assessment mechanism. Evaluation on two scenarios: pedestrian crossing and lane merging, which are part of the National Highway Traffic Safety Administration (NHTSA) testable cases and scenarios [42], demonstrates the assessment mechanism's ability to evaluate a range of car strategies with varying safety levels based on the Fréchet distance of the collision trajectories generated.

2 Background and Related Work

2.1 Autonomous Car Verification and Testing

Previous work such as [2, 38, 46] explored the use of formal methods for autonomous car system verification. These methods require complete formal specifications to be accurate, and is often difficult to construct completely due to the complexity of autonomous car systems. Scalability issues often occur due to the huge possibilities of different scenarios that an autonomous car might encounter. Other work such as [6, 29, 41, 45] attempt to generate test scenarios leading to accidents within their assessment mechanism. One of the difficulties encountered were the rare nature of accidents, which would require a large compute time to obtain edge-case scenarios. Therefore, [18] and [37] aimed to estimate and predict future trajectories to form scenarios using particle filtering and probabilistic methods. Due to the scarcity of data, [25] proposed a framework using conformal prediction to guarantee a low false negative rate in a driver alert system. However, this method still relies on the need for unsafe data, with the error rate decreasing even more with the increase in data provided.

Obviously, a mechanism to test the safety of a car is not new. The well-accepted NCAP testing protocol was introduced in 1979. However, testing scenarios developed by NCAP are mostly static with no consideration of the behavior of the car being assessed, which is not suitable for autonomous cars. To generate safe and collision trajectories, our proposed testing mechanism will benefit from having a predictive model of the car's behavior, albeit imperfect. For this purpose, many work can be adopted, such as [4, 13, 17, 36]. Adversarial trajectory generation can also benefit from work on pursuit evasion, such as [10] and [23], though the latter is focused on deforming observations rather than an adversary's behavior.

2.2 POMDP Formulation

The standard POMDP can be formally defined by the 8-tuple $\langle S, A, O, T, Z, R, b_0, \gamma \rangle$, where S is the set of states s; A is the set of actions a; O is the set of observations o; $T(s, a, s')$ is the transition function denoting the probability $P(s'|s, a)$ of the agent landing in state $s' \in S$ after performing action $a \in A$ from previous state $s \in S$; $Z(s', a, o)$ is the observation function denoting the probability $P(o|s', a)$ the agent makes observation o after executing action $a' \in A$ in state $s' \in S$; $R(s, a) \in \mathbb{R}$ is the reward function; b_0 is the agent's initial belief; $\gamma \in (0, 1)$ is the discount factor. The agent's goal is to find a policy π which maximizes the value function given by $V(b, \pi) = E\left[\sum_{t=0}^{\infty} \gamma^t R(s_t, a_t)|b, \pi\right]$. The optimal policy is then defined as $\pi^* = argmax_{\pi \in \prod} V(b, \pi)$, where $\prod$ is the set of policies defined as mappings from beliefs to actions.

In certain situations, the problem is further complicated by the presence of multiple objectives. For example, a battery powered robot have to complete tasks while minimizing power consumption. Such problems can be modeled and solved as Multi-objective POMDPs [40]. A naive way to solve MOPOMDPs is to convert the vector-valued reward function to a scalar function via weight scalarization, and solve using single-objective POMDPs. Another way is to set some of the objectives as constraints, forming a Constraint-POMDP (CPOMDP) problem. In this case, the POMDP has to be solved while requiring that the constraints are respected. CPOMDPs are generally harder to solved compared to POMDPs. The CPOMDP is defined as $\langle P, C \rangle$, with P being the POMDP and $C(s) \forall s \in S$ denoting the constraint set. Its solution is a policy for P that satisfies all the constraints defined in C. Our proposed method, CAT, adopts this second approach.

2.3 Related MOPOMDP and CPOMDP Solvers

The state of the art of solving MOPOMDPs is less mature compared to POMDPs. The idea proposed in [35] is to compute a bounded approximation of the optimal solution set for all possible weightings of the objectives. It reuses policies and value functions from previous iterations while solving a series of scalarized single-objective POMDPs, yet this is still quite expensive to compute.

Solving CPOMDPs includes either sub-optimal or approximate methods based on extending existing POMDP solving methods such as PBVI [22], dynamic programming [19], approximate linear programming [32] or online search [43]. Approximate dynamic methods such as [22] shows that optimal policies in CPOMDPs can be randomized and solved it using PBVI with admissible costs. In [19], an exact dynamic update for CPOMDPs using a piecewise linear representation of the value function is proposed. However, the proposed technique is only useful for finding solutions to modestly sized sequential decision problems that include uncertainty and constraints and will struggle to scale to large-scale problems. The work in [43] shows that constraints cannot be modeled simply as a reward function as its value needs to be properly tuned to obtain a good policy, which can be time-consuming.

2.4 Fréchet Distance

Fréchet distance [14] measures the similarity between two curves while adhering to the order of points on the curve. Suppose $P : [0,1] \to \mathbb{R}^n$ and $Q : [0,1] \to \mathbb{R}^n$ are two curves on the same space. Then the Fréchet distance between these two curves are:

$$d(P,Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} ||P(\alpha(t)) - Q(\beta(t))|| \tag{1}$$

among all possible $\alpha : [0,1] \to [0,1]$ and $\beta : [0,1] \to [0,1]$, which are continuous reparameterisations of P and Q that are non-decreasing and subjective.

For computational efficiency, in this paper, we use the approximation of Eq. (1) via discrete Fréchet distance, approximating P and Q as polygonal chains, resulting in the following definition. Suppose $P' : (p_1, p_2, p_3, \cdots, p_m)$ and $Q' : (q_1, q_2, q_3, \cdots, q_{m'})$, where $p_i, q_j \in \mathbb{R}^n$ for $i \in [1,m], j \in [1,m']$. Then the Fréchet distance between these two polygonal chains are:

$$\begin{aligned} &d(P',Q') = \min_{k,l} \max_{t} ||p_{k(t)} - q_{l(t)}|| \\ &\text{where:} \\ &(i)\ k(t+1) = k(t)+1 \text{ and } l(t+1) = l(t), \text{ or} \\ &(ii)\ k(t+1) = k(t) \text{ and } l(t+1) = l(t)+1 \end{aligned} \tag{2}$$

Suppose $m \geq m'$, then $t \in [1,m]$, while $k(t)$ and $l(t)$ maps the index t to an index of the points in P' and Q', respectively. This distance can be computed in $O(\frac{mm' \log\log m}{\log m})$ time and $O(m+m')$ space[1].

2.5 The ϵ-constraint Method

Given the following Multi-Objective Mathematical Programming (MOMP) Problem [26]:

$$\max\ (f_1(x), f_2(x), \cdots, f_p(x)) \text{ s.t. } x \in S \tag{3}$$

where x is the vector of decision variables, $f_1(x), \cdots, f_p(x)$ are the p objective functions. S is the feasible region. In the ϵ-constraint method, one of the objective function is optimized using other objective functions as constraints. They are incorporated into the constraint part of the model shown below [7, 11]:

$$\max\ f_1(x) \text{ s.t. } f_2(x) \leq e_2, \cdots, f_p(x) \leq e_p, x \in S \tag{4}$$

The efficient solution can be obtained by parametrical variation of the constraint objective functions (e_i). Applying the general weighting method to the original feasible problem in linear problems will result in a corner solution. Instead, the ϵ-constraint method alters the original feasible region and is able to produce non-extreme efficient solutions, thus obtaining a more rich representation of the efficient set. The ϵ-constraint method also negates the need to scale the objective functions to a common scale before forming the weighted sum.

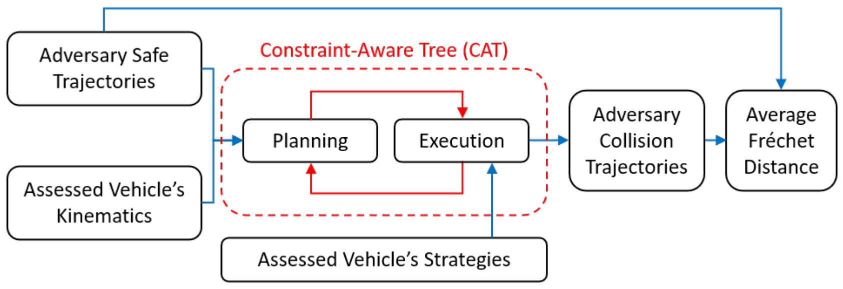

Adversary Safe Trajectories
Constraint-Aware Tree (CAT)
Planning
Execution
Adversary Collision Trajectories
Average Fréchet Distance
Assessed Vehicle's Kinematics
Assessed Vehicle's Strategies

Fig. 1. Proposed safety assessment mechanism

3 The Safety Assessment Mechanism

Fig. 1 illustrates the proposed safety assessment mechanism. This mechanism treats the autonomous driving system of the car being assessed as a black-box, while taking the vehicle's kinematics and dynamics constraints as inputs. Note that information about the kinematics and dynamics of the car does not need to be perfect. The safety of the autonomous car is assessed under a given problem scenario. The problem scenario can be specifically designed to test certain components of the vehicle, or a derivation of existing safety assessment standard, such as NCAP. We assume these testing scenarios are provided and defined as $\langle E, AV, \Pi_{i=1}^{k} adv_i \rangle$, where $E \subseteq \mathbb{R}^m$ with $m \in [2, 3]$ denotes the workspace of the assessed vehicle and the adversaries.

The vehicle being assessed is denoted as AV and defined as $\langle S_c, A_c, F_c \rangle$, where S_c is the set of states of the vehicle being assessed, A_c is the set of actions the assessed vehicle can perform, and $F_c(s_c, a_c, t, \Delta_t, \mathcal{P}_c)$ is the assessed vehicle's dynamics function, which outputs a possible next state after an action $a_c \in A_c$ is performed from state $s_c \in S_c$ at time t for a duration of Δ_t and perturbed by an error distribution $\mathcal{P}_c$. The mechanism does not know the exact action the car takes at any given time and state, but it can construct an internal model based on the observed behavior. This internal model is of course not exact, and therefore the assessment mechanism considers that the vehicle behavior to only be partially observed.

The notation adv_i denotes an adversary and k denotes the number of adversaries in the problem scenario. An example of an adversary is the pedestrian in a pedestrian avoidance testing scenario. An adversary adv_i is defined as $\langle S_{adv_i}, A_{adv_i}, F_{adv_i} \rangle$, where S_{adv_i} is the set of states and A_{adv_i} is the set of actions of adversary-i. $F_{adv_i}(s_{adv}, a_{adv}, t, \Delta_t, \mathcal{P}_{adv_i})$ is a stochastic model of the adversary's dynamic function. It outputs the adversary's next state after an action $a_{adv} \in A_{adv_i}$ is performed from state $s_{adv} \in S_{adv_i}$ at time t for duration Δ_t, influenced by error distribution $\mathcal{P}_{adv_i}$.

To assess the vehicle's safety, our proposed mechanism controls the behaviors of the adversaries. Specifically, the mechanism computes a set of safe trajectories of the joint adversaries adv_i ($i \in [1, k]$), denoted as Φ. The mechanism can also use real safe trajectories data, if available. For each safe trajectory $\phi \in \Phi$, the mechanism computes a set $\Omega(\phi)$ of adversarial trajectories that are as close as possible to the safe trajectory but cause collisions with the assessed vehicle. The problem of computing such colliding trajectories is framed as a POMDP to account for the partial observability of the assessed vehicle's behavior and internal reasoning. Due to the conflicting nature of the objectives —being close to a safe trajectory while colliding with the assessed vehicle—, the POMDP policy is computed using CAT, which is designed for multi-objective POMDPs and presented in Section 4.

The safety indicator is then computed as the average Fréchet distance between the safe trajectories and the associated collision trajectories:

$$\overline{F}(\Phi, \Omega) = \frac{1}{|\Omega|} \sum_{\phi \in \Phi} \sum_{\omega \in \Omega(\phi)} F(\phi, \omega) \tag{5}$$

where $\Omega = \bigcup_{\phi \in \Phi} \Omega(\phi)$ with $\phi \in \Phi$ being a safe trajectory for the adversary(ies) and $\Omega(\phi)$ being the set of colliding trajectories associated with ϕ. The notation $F(\phi, \omega)$ represents the discrete Fréchet distance (Eq. (2)) between a safe and a corresponding colliding trajectories of the adversary(ies).

Based on the definition of Fréchet distance, $F(\phi, \omega) = \delta_{\phi,\omega}$ implies that $\forall_{p \in \phi} \exists_{q \in \omega}\ q \in B(p, \delta_{\phi,\omega})$, where $B(p, \delta_{\phi,\omega}) \subset E$ is a ball centered at p with radius $\delta_{\phi,\omega}$. Therefore, $\overline{F}(\Phi, \Omega) = \delta$ implies that on average, any point of a safe adversary trajectory is only δ distance away from a colliding trajectory. Assuming the set of safe and colliding adversarial trajectories used in the distance computation are sufficiently representative, an autonomous car with smaller δ will likely have less buffer to avoid accidents compared to those with larger δ.

4 Constraint-Aware Tree (CAT)

CAT is an online POMDP solver that computes a close to optimal policy for the adversaries to collide with the vehicle being assessed as fast as possible, while being as close as possible to a given safe trajectory $\phi \in \Phi$. This problem is essentially a multi-objective optimization problem and CAT adopts the ϵ-constraint approach [26] to compute a (close to) pareto optimal policy.

Specifically, CAT sets the distance to the safe trajectory ϕ as a constraint in computing a strategy for the adversary(ies) to collide with the vehicle being assessed as fast as possible. To identify a suitable distance constraint to use, CAT selects a finite set of distance constraints, denoted as $\mathcal{E} = \{\epsilon_1, \epsilon_2, \cdots, \epsilon_n\}$, where $\epsilon_i < \epsilon_{i+1}$ for all $i \in [1, n-1]$. Conceptually, CAT constructs a CPOMDP problem for each constraint $\epsilon_i \in \mathcal{E}$ and evaluates the values and collision rate of the CPOMDP's policy. To generate the adversaries' trajectories, at each time step, CAT selects the best action of the CPOMDP with the smallest distance constraint among those whose estimated collision rate at the current belief is larger than or equal to a given threshold.

Key to CAT is that although conceptually, it computes the best solution of n CPOMDPs, CAT computes all of them at once, utilising prior computations as much as possible. The details of CAT's procedure are provided below.

4.1 The Multiple CPOMDP Models and Their Features

Let's denote the CPOMDP problem associated with distance constraint $\epsilon_i \in \mathcal{E}$ as $\mathcal{P}_{\phi,\epsilon_i} = \langle S, A, T, O, Z, R, \gamma, C, \phi, \epsilon_i \rangle$, where $i \in [1, n]$ and $n = |\mathcal{E}|$. The constraint $C(s)$ at state $s \in S$ is satisfied if and only if $d(\phi, s) \leq \epsilon_i$, where $d(\phi, s)$ is the distance function as defined in Eq. (2) (s can be viewed as a degenerate polygonal chain). Moreover, a state $s \in S$ where $C(s)$ is not satisfied is modelled as a terminating state. Intuitively, the constraint limits the movement of the adversaries to only be inside the union of ϵ_i-balls around the safe trajectory ϕ, i.e., inside $\bigcup_{p \in \phi} B(p, \epsilon_i) \subset S$ where $B(p, \epsilon_i)$ is a ball center at p and radius ϵ_i. This framing of the constraints enables CAT to solve the CPOMDPs $\mathcal{P}_{\phi,\epsilon_i}$ as unconstrained POMDPs where states that do not satisfy the constraint becomes terminating states and entering them incur a penalty. For simplicity, we use the same notation to refer to the CPOMDP and its corresponding POMDP problem.

Although the above POMDP problem can be solved using existing solvers, solving n POMDP problems separately for each safe trajectory is too expensive. But, notice two features of these problems. First, all n POMDP problems (converted from the CPOMDPs $\mathcal{P}_{\phi,\epsilon_i}$ for $i \in [1, n]$) have the same state, action, and observation spaces, transition and observation functions, and differ only in their reward functions. The second feature is the set of reachable beliefs of the POMDPs associated with smaller distance constraints is a subset of those of POMDPs associated with larger distance constraints. Therefore, suppose $\mathcal{R}(\mathcal{P}_{\phi,\epsilon_i}, b_0)$ is the set of beliefs reachable from the initial belief b_0 for POMDP $\mathcal{P}_{\phi,\epsilon_i}$, since the set of distance constraints $\mathcal{E}$ is constructed such that $\epsilon_1 \leq \epsilon_2 \leq \cdots \leq \epsilon_n$, then $\mathcal{R}(\mathcal{P}_{\phi,\epsilon_i}, b_0) \subseteq \mathcal{R}(\mathcal{P}_{\phi,\epsilon_{i+1}}, b_0)$ for $i \in [1, n-1]$. CAT exploits these two features to compute the solution to all n POMDPs at once and to reuse prior computation.

4.2 Belief Tree Construction and Backup

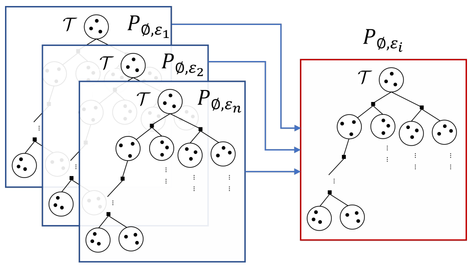

$\mathcal{T}$
$P_{\emptyset,\varepsilon_1}$
$P_{\emptyset,\varepsilon_2}$
$P_{\emptyset,\varepsilon_n}$
$P_{\emptyset,\varepsilon_i}$

Fig. 2. CAT computes all n CPOMDPs at once.

Similar to almost any online POMDP solver today, CAT constructs a belief tree and uses the tree to represent the POMDP policy. Due to the similarity of the POMDPs and the subset relation of the reachable beliefs of the different POMDPs, given a safe trajectory, it is sufficient for CAT to construct only a single belief tree (denoted as $\mathcal{T}$) for solving all n POMDPs.

The belief tree $\mathcal{T}$ is a tree where each node b represents a belief in the belief space $\mathcal{B}$. For simplicity, we use the same notation for a node in $\mathcal{T}$ and its corresponding belief in $\mathcal{B}$. An edge in $\mathcal{T}$ is labeled with a pair of action–observation (a, o) where $a \in A$ and $o \in O$. Also, an edge labeled (a, o) from a node b to b' means $b' = \tau(b, a, o)$. To reflect the different rewards, and hence value functions of the different POMDP problems, at each node b of $\mathcal{T}$, CAT maintains both the value functions $[V_1(b), V_2(b), \cdots, V_n(b)]$ and Q-value estimates $[Q_1(b), Q_2(b), \cdots, Q_n(b)]$, where V_i and Q_i for $i \in [1, n]$ denotes the corresponding estimated values of b for the POMDP problem that corresponds to $\mathcal{P}_{\phi,\epsilon_i}$ where $\epsilon_i \in \mathcal{E}$. In addition, to help decide which problem to evaluate when, CAT maintains the total number N_i of times $\mathcal{P}_{\phi,\epsilon_i}$ is evaluated and the number $N_i(b)$ that a node b in $\mathcal{T}$ is visited for evaluating $\mathcal{P}_{\phi,\epsilon_i}$ for $i \in [1, n]$. Fig. 2 illustrates the belief tree CAT constructs.

To construct $\mathcal{T}$, CAT starts by selecting the particular POMDP problem $\mathcal{P}_{\phi,\epsilon_i}$ ($i \in [1, n]$) to evaluate using a bandit algorithm based on [44]:

$$i = \underset{j\in[1,n]}{argmax} \left(R_j(b_0) + C\sqrt{\frac{log(N_j)}{N_j(b_0)}} + D \cdot \text{diff}(R) \right) \tag{6}$$

where $R_j(b_0)$ is the estimated collision rate for performing the action with the highest Q-value at belief b_0, and $\text{diff}(R) = H - R_j(b_0)$ is the difference of the estimated collision rate to collision threshold H. N_j is the total number of times $\mathcal{P}_{\phi,\epsilon_i}$ is evaluated and $N_j(b)$ is the number of times node b is visited for evaluating $\mathcal{P}_{\phi,\epsilon_i}$. The constant C is an exploration constant, with larger values of C biases the algorithm towards exploration. The constant D balances sampling towards problems $\mathcal{P}_{\phi,\epsilon_i}$ with collision rate less than H, with sufficient sampling possibly bringing it over the threshold.

Suppose the problem selected is $\mathcal{P}_{\phi,\epsilon_i}$. Then, to expand $\mathcal{T}$ under the POMDP problem $\mathcal{P}_{\phi,\epsilon_i}$, CAT uses a POMCP-like method [39] and perform episode sampling — a sequence of $\langle s, a, o, r\rangle$, where $s \in S$, $a \in A$, $o \in O$, and $r = R(s, a)$. Each node of $\mathcal{T}$ is represented as a set of state particles, and to sample an episode, CAT starts by sampling a particle from the root node b_0 of $\mathcal{T}$. Suppose the sampled particle is $s \in S$, then CAT selects an action $a \in A$ to use based on UCB1 [3] strategy: $a = \underset{a'\in A}{argmax} \left(Q_i(b_0, a') + c' \cdot \sqrt{\frac{log(N_i(b_0))}{N_i(b_0,a')}} \right)$, where $Q_i(b_0, a')$ is the estimated Q-value for performing a' at belief b_0 under the POMDP problem $\mathcal{P}_{\phi,\epsilon_i}$. Once an action is selected, CAT samples a next state $s' \in S$ based on $T(s, a, s')$, samples an observation $o \in O$ based on $Z(s', a, o)$, and a reward $r = R(s, a)$ is then incurred. If the pair (a, o) has been used to expand b_0, s' is added to the set of particles representing node b, which is the child of b_0 via (a, o). In this case, if s' is not a terminating state, the sampling process repeats starting from b. Otherwise, backup is performed to revise the value estimate. If the pair (a, o) has not been used to expand b_0, a new node b is added as a child of b_0 in $\mathcal{T}$ via an edge labelled (a, o). A default (roll-out) strategy is then performed to provide an initial value estimate for b, and backup is performed to revise the value estimate of the nodes visited by the sampling process.

Since all n POMDP definitions differ only in the reward functions and due to the subset relation of the reachable beliefs of the POMDPs (discussed in Section 4.1), the outcome of sampling for one POMDP can be propagated to other POMDP problems too. For example, if the episode for problem $\mathcal{P}_{\phi,\epsilon_i}$ terminates and receive a large negative reward due to violating the constraint ϵ_i, then this termination and reward would apply to all $\mathcal{P}_{\phi,\epsilon_k}$ where $k < i \in [1,n]$ too. Conversely, reaching a goal for $\mathcal{P}_{\phi,\epsilon_k}$ without violating ϵ_i would mean the goal can be reached for all $\mathcal{P}_{\phi,\epsilon_k}$ for $k > i \in [1,n]$. This feature enables CAT to perform sampling for multiple CPOMDP problems simultaneously to increase sampling efficiency and accuracy.

After an episode for a POMDP problem, say $\mathcal{P}_{\phi,\epsilon_i}$, is sampled, CAT updates the value functions, Q-value estimates and the statistics of all the nodes in $\mathcal{T}$ along the path visited by the episode, under the problem $\mathcal{P}_{\phi,\epsilon_i}$. The Q-value function is updated using a stochastic version of the Bellman backup, following the implementation of ABT [24] and evaluation in [16]. Suppose an element of the sampled episode is $\langle s, a, o, r \rangle$, where $s \in S$ is a particle of node b in $\mathcal{T}$, $a \in A$, $o \in O$, $r = R(s,a)$, and suppose b' is the child node of b in $\mathcal{T}$ via edge (a,o). Then, the updated Q-value is computed as: $Q_i(b,a) = Q_i(b,a) + \frac{1}{N_i(b,a)}\left(r + \gamma V_i(b') - Q_i(b,a)\right)$, where $N_i(b,a)$ is the number of times action a is used to expand node b under problem $\mathcal{P}_{\phi,\epsilon_i}$.

4.3 Selecting the Set of Distance Constraints

Last but not least, the question is how do we select the set of distance constraints $\mathcal{E}$. For this purpose, CAT first computes the minimum ϵ_{min} and maximum ϵ_{max} distance that the adversaries need to move in order to collide with the assessed vehicle assuming deterministic motion, based on the known kinematics and dynamics of the vehicle. CAT then discretizes the range $[\epsilon_{min}, \epsilon_{max}] \subset \mathbb{R}$ based on a given resolution and sets the discretized points as the distance constraints.

5 Experiments and Results

The purpose of our experiments is twofold. The first is to evaluate CAT in generating colliding adversarial trajectories that are as close as possible to a given safe trajectory. We use nuScenes dataset, a public autonomous driving dataset containing a diverse set of traffic scenes. The second purpose is to test if the average Fréchet distance, as described in Eq. (5), can be used as a safety indicator for different autonomous driving software systems (these systems refer to the full-stack software, which includes perception, planning and control). We design testing scenarios based on the NHTSA pre-crash scenario typology [27] within CARLA Driving Simulator to benchmark top performing controllers submitted to the CARLA leaderboard. The experiments are performed on a desktop computer with an 8 Core Intel Xeon Silver 4110 Processor and 128GB DDR4 RAM. Our proposed solver, CAT, and the POMDP-based motion planning problems are implemented using the software toolkit OPPT [15].

5.1 Adversarial Scenario Generation using CAT

We first highlight the capabilities of CAT in generating adversarial collision trajectories. We compare CAT against STRIVE, a state-of-the art Machine

Table 1. Collision trajectory generation comparison on nuScenes dataset

Method	t (s)	Scenes where collision trajectory is present	Collision rate (%)	Fréchet distance, δ
STRIVE (reported)	420 (offline)	162/327	-	-
STRIVE (validated)	420 (offline)	142/327	-	9.936
ABT (w=1)	3 (online)	192/327	86.88 ± 3.688	8.022 ± 5.686
ABT (w=5)	3 (online)	179/327	90.66 ± 3.273	6.930 ± 4.303
ABT (w=10)	3 (online)	146/327	79.52 ± 4.766	4.228 ± 3.029
CAT (Ours)	3 (online)	205/327	88.06 ± 3.349	7.233 ± 4.594
CAT (Ours)	6 (online)	211/327	88.25 ± 3.258	7.336 ± 5.064
CAT (Ours)	9 (online)	211/327	90.03 ± 2.916	7.249 ± 4.992

Learning-based method, and ABT, a POMDP method which acts as a baseline. For POMDP-based methods, each scene is repeated for 10 runs and a planning time per step t of 3 seconds. This set of experiment is conducted on the nuScenes dataset which contains a diverse set of road traffic scenes collected from Singapore and Boston. This dataset is simplified to only include car and truck vehicles operating on rasterized drivable area, carpark area, road divider and lane divider map layers. The goal of the controlled adversarial agent is to generate collision trajectories with the identified target vehicle (the vehicle being assessed), while navigating through traffic and avoiding collision with other vehicles and the environment.

STRIVE STRIVE generates collision trajectories by optimizing the latent space of a learned traffic model in the form of a graph-based conditional VAE. It uses 2s (4 steps) of prior motion to predict 6s (12 steps) of potential collision trajectories. We replicate and analyze the collision trajectories generated by STRIVE and exclude scenes where the agent is in environmental collision in its inital state. We observe that STRIVE is able to generate collision trajectories for 162 of the 368 scenes tested as seen in Table 1. However, upon visual inspection of the trajectories, we found that in 20 of these scenes, the collision trajectories violate traffic rules. For instance, the agent cuts across road dividers and going off-road as seen in examples in Fig. 3. If we were to label these collision trajectories as false positives, and therefore no collision trajectories are present in those 20 scenes, STRIVE is only able to find solutions in 142 of the 327 scenes.

ABT Our mechanism rely on the ability to generate many colliding trajectories that are as close as possible to the safe trajectories for the adversaries. Due to the partially observable nature of the assessed vehicle strategies, this trajectory generation problem is essentially a POMDP problem. However, it has conflicting objectives. A baseline method for solving such POMDP problems is via weights scalarization, i.e., the reward for one step move is $r_{step} = w \cdot r_{base} \cdot d_{l2}$, where r_{base} is the base step penalty and d_{l2} is the Euclidean distance to the safe trajectory. It is designed to encourage collision while minimizing distance to the safe

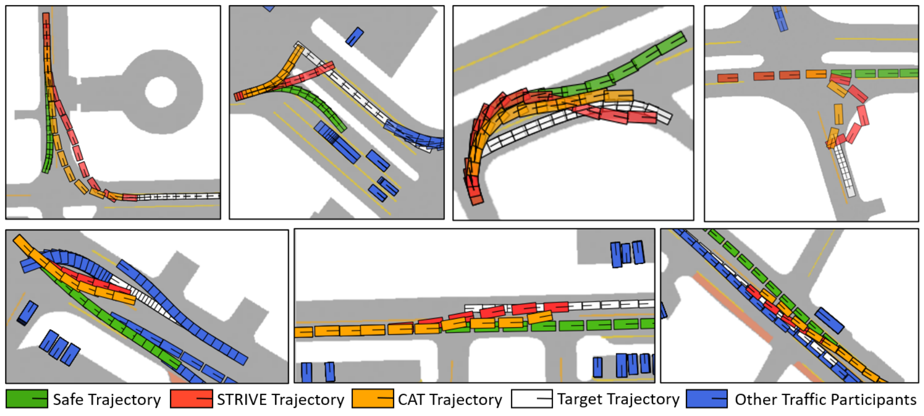


Fig. 3. Qualitative comparison of trajectories generated on nuScenes dataset. The target trajectory is the trajectory of the identified target vehicle (the vehicle being assessed). Row 1: Invalid collision trajectories generated by STRIVE incorrectly labeled as successful. CAT finds valid collision trajectories in these scenes. Row 2: The collision trajectories generated by CAT are closer to the safe trajectory compared to STRIVE, as indicated by the smaller average Fréchet distance.

trajectory. We model the vehicle to have the same driving parameters matching STRIVE which uses 2D bicycle kinematics, discretized evenly to form 25 distinct actions. To compute an optimal solution for ABT, we first need to take an extra step to find the suitable w. The performance comparison between ABT using this baseline techniques with $w = \{1, 5, 10\}$ is presented in Table 1. The results indicate that although increasing w results in collision trajectories that are closer to the corresponding safe trajectories, the number of scenes where collision trajectories are found, together with collision rate decreases.

CAT We used the proposed CAT solver to generate collision trajectories on the same set of scenes. The adversarial agent is given a large reward of +1000 for successful collision, -1000 for collision with other vehicles and the environment, and -1 step cost. We use $n_\epsilon = 10$ for all scenes.

Discussion We observe that at $t = 3$, CAT is able to generate collision trajectories in 205 out of the 327 scenes, a significant improvement compared to STRIVE's 142 and ABT's 192. CAT also has a high average collision rate at 88.06%. In many scenes, CAT is able to generate successful collision trajectories within road boundaries unlike STRIVE, as shown in Fig. 3. Unlike ABT, CAT does not need to find optimal weights since it does not rely on reward function to bias the objectives. Instead, it intelligently solves for all ϵ at the same time. At $t = 3$, CAT achieves a comparably high collision rate compared to ABT. At this planning time, CAT essentially solves 10 CPOMDPs at once.

One may wonder the collision rate comparison between ABT and CAT on the same scenes. If we consider only the scenes where both ABT and CAT find collision trajectories, then CAT achieves a collision rate of 92.98±2.532% at 9s planning time, while ABT(w=5) achieves a collision rate of 92.23±3.028% at 3s planning time.

Note that our goal is not just to generate collision trajectories, but collision trajectories that are close to the safe trajectory. We observe that the trajectories generated by CAT has the lowest δ unit distance at 7.233, compared to STRIVE's 9.936 and ABT's 8.022. Several examples shown in Fig. 3 shows that the collision trajectories generated by CAT deviates less from the original safe trajectories, maintaining naturalness while generating the collision trajectories. The results indicate CAT's effectiveness in solving the multi-objective problem by being able to generate a larger amount of collision trajectories while at the same time having these collision trajectories being closer to the safe trajectories. These results indicate the viability of CAT to alleviate the issues caused by rare collision data in assessing the safety of autonomous vehicles.

Last but not least, CAT requires much less computation time than STRIVE. Aside from hours needed to train the traffic model, each collision trajectory STRIVE generates require it to run an adversarial optimization process, which takes up to 7 minutes, depending on the number of vehicles in the scene. Comparatively, given a planning time of 3 seconds per step, with an average of 8 steps per scene, CAT is able to find a collision trajectory within 30 seconds on average for each scene. These results indicate CAT in able to generate collision trajectories much faster than STRIVE without large amount of prior data.

5.2 Benchmarking Autonomous Driving Software Systems in CARLA Simulator

Here, we evaluate if the average Fréchet distance, as described in Eq. (5), can be used as a safety indicator. We implemented 2 scenarios based on the NHTSA pre-crash scenario typology: Pedestrian Crossing and Lane Merging. Using these two scenarios, we benchmarked four of the top ranked state-of-the-art Machine Learning based methods submitted to the CARLA Autonomous Driving Leaderboard. The software system takes as input the measurements from the various sensors (Lidar, Radar, RGB sensors) on the vehicle and are trained using Deep Learning methods to navigate through various traffic scenarios. We use pre-trained weight provided by the respective authors for the experiments.

Autonomous Driving Software Systems We give a short description of the software systems being assessed in this section. **TCP** [47] is a camera-only model. By observing that waypoints are stronger at collision avoidance compared to directly predicting controls, it proposes a situation-dependent network with two branches which generates the waypoints and control signal respectively. During run time, the two outputs are generated with a weighted average that varies based on whether the vehicle is turning. **NEAT** [8] proposes neural attention fields which enables reasoning for end-to-end imitation learning. It uses imitation-learning with attention and implicit functions to iteratively compress

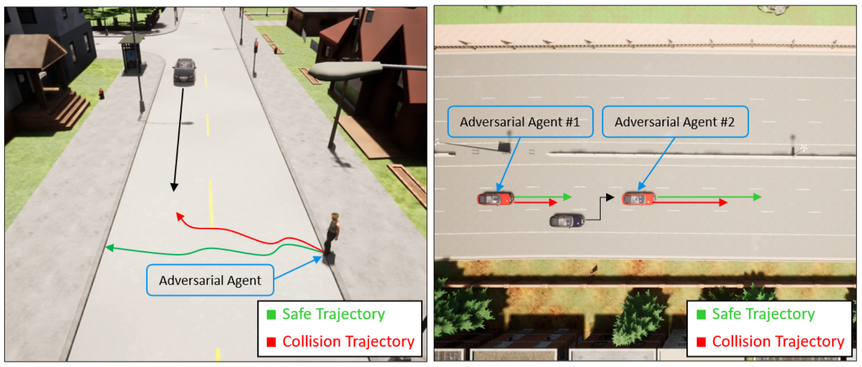
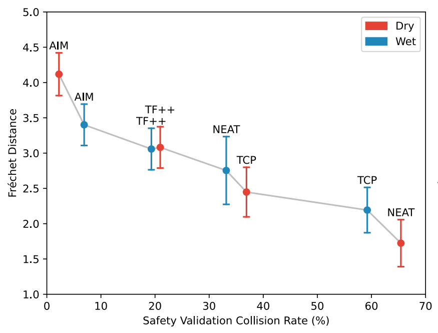


Fig. 4. Experiment Scenarios: (a) Pedestrian Crossing and (b) Lane Merging

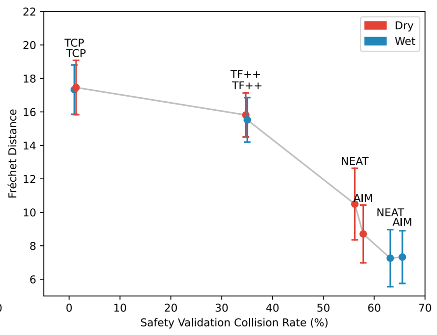
high dimensional 2D image features into a compact bird-eye-view representation for driving. The attention mechanism has been demonstrated to be a powerful module, however the utilization of a relatively dense representation drastically increases model complexity. **AIM** [33] takes the birds-eye-view of the target location as an input, similar to NEAT, which is then sent to a ResNet 34 encoder pre-trained on ImageNet. It outputs waypoints through four GRU decoders followed by PID controllers. Adding auxiliary tasks during training such as using a deconvolutional decoder to predict the 2D depth and semantic segmentation is shown to increase driving performance. **TF++** [20] is an improved variant of Transfuser [9] by modifying its architecture, output representation and training strategy. It uses a transformer decoder for pooling features to mitigate out of distribution errors that may arise when steering directly towards a target point. It also considers the prediction uncertainties into the final output by using a confidence weighted average of the predicted target speed as input to the controller as an attempt to reduce collisions.

Scenario 1: Pedestrian Crossing This well-known scenario is derived from a testing scenario used by NCAP to assess the emergency braking system of an autonomous vehicle. Fig. 4(a) illustrates an instantiation of this scenario. The assessed vehicle is moving in one lane of the street and approaches a pedestrian crossing the street in front of the vehicle. In this scenario, the adversary is the pedestrian, controlled by CAT, with the objective of causing a collision with the approaching assessed vehicle. The pedestrian has 9 actions, moving along equally spaced directions between and including East and West directions, at a fixed velocity of $2.5m/s$. The pedestrian's motion is deterministic and it perceives a noisy observation of the distance between itself and the assessed vehicle.

Scenario 2: Lane Merging Fig. 4(b) illustrates an instantiation of this scenario. The autonomous vehicle being assessed starts at the right lane of a two-lane road. Its objective is to merge to the left lane between the two vehicles



(a)



(b)

Fig. 5. Experiment Results for (a) Pedestrian Crossing and (b) Lane Merging

Table 2. Autonomous Driving Software System Driving Score on CARLA Leaderboard

Software System	Driving Score ↑	Collisions pedestrians ↓	Collisions vehicles ↓
TCP	75.14	0	0.32
TF++	66.32	0	0.5
NEAT	21.83	0.04	0.74
AIM	19.38	0.18	1.53

as seen in Fig. 4(b). The two adversarial vehicles have 5 varying speeds of v_{car} given by $\{3, 4, 5, 6, 7\}$m/s. Each action is also influenced by uncertainty given by $\mu\,[-0.02v_{car}, 0.02v_{car}]$. This is a challenging scenario for CAT as it is controlling two agents instead of only one, doubling the size of the action space.

Benchmark and Results For both scenarios, experiments were conducted on a set Φ of 10 different safe trajectories and 10 sets of 50 colliding trajectories, with each set $\Omega(\phi)$ of colliding trajectory corresponding to a safe trajectory $\phi \in \Phi$. For each software system, the experiment is repeated on both dry and wet road conditions. A planning time per step of 10s is given to CAT for both scenarios. At the same time, to establish a baseline of safeness of each software system, we simulate 5,000 runs (50 runs for each of 100 random safe trajectories) for both scenarios. A higher average collision rate would indicate that the software system is generally more dangerous at the tested road condition. We list the benchmark results for Scenarios 1 and 2 in Fig. 5(a) and Fig. 5(b) respectively. We observe that in both scenarios, the safer the software system is (given by the lower collision rate from validating random trajectories), the higher the Fréchet Distance, δ obtained from generating the collision trajectories using CAT, indicating the vehicle has larger room to account for errors and uncertainty. This consistent trend of results shown in both scenarios indicates the effectiveness of our proposed safety assessment mechanism.

An interesting observation is that while we would expect a vehicle to be safer or at least remain the same in the dry compared to wet conditions, NEAT software system exhibits the opposite behavior in Scenario 1. The Fréchet Distance obtained in wet conditions is larger compared to dry conditions, and is also validated by the increase in collision rate. This is an example of the unique behavior that CAT is able to identify using our proposed safety assessment mechanism.

The vehicle rankings scores obtained from CARLA Leaderboard is shown in Table 2. A higher score exhibits the software system's ability to complete the routes while minimizing infractions such as collisions with other pedestrians or vehicles. Here, we observe the Driving Score is inversely correlated to the number of collisions with other vehicles (normalized per km). The trend for our results in Scenario 2 is the same (Fig. 5(b)), indicating that for Scenario 2, our proposed mechanism is aligned with the safety indicator of the CARLA Leaderboard. However, the trend is different in Scenario 1. The reason for this difference in trend for Scenario 1 requires further investigation.

In CARLA Leaderboard, assessment is performed by having the software system traverse routes on 2 secret maps containing different test scenarios. Users has reported discrepancies in reported results by resubmitting the open-sourced methods using released or retrained model files [20]. Since the evaluation is secret, the exact reasons for these fluctuations is unknown. Furthermore, users are only allowed to make up to 5 submissions per month, with evaluation time taking up to 4 weeks. Since our safety assessment mechanism is adaptive to the behavior of the assessed vehicle, our tests does not need to be a secret. Furthermore, since our mechanism is able to efficiently generate collision trajectories, and evaluations can be done quickly after every software update.

6 Summary

This paper proposes a mechanism to test the safety of autonomous vehicle systems. The mechanism automatically generates crash scenarios and is adaptive to the assessed autonomous vehicle. Given a dataset of collision-free trajectories, rare-event collision trajectories of the adversaries are generated while accounting for the uncertainty of the autonomous system's strategies. Core to this mechanism is an on-line method to approximately solve multi-objective POMDPs, called CAT, to efficiently generate colliding trajectories of the adversaries that are as close as possible to a given safe trajectory. We show that CAT is able to generate better collision trajectories with high success rate compared to state-of-the-art Deep Learning based trajectory generation method, such as STRIVE. Our results also indicate that the proposed safety assessment mechanism and indicator can be applied to reasonably assess the safety of different autonomous driving software in CARLA Simulator.

The safety assessment mechanism and CAT can be applied to assess the safety of a wide variety of robots, though scalability might be an issue for very high DOFs robots. Moreover, although Fréchet distance seems suitable to be used in our framework (reasoning provided in Section 3), it is still useful to explore other forms of distance measure in the future.

Acknowledgments. This work was partially supported by the Assuring Autonomy International Program and ANU Futures Scheme.

References

1. Agarwal, P.K., Avraham, R.B., Kaplan, H., Sharir, M.: Computing the discrete fréchet distance in subquadratic time. SIAM Journal on Computing **43**(2), 429–449 (2014)
2. Althoff, M., Dolan, J.M.: Online verification of automated road vehicles using reachability analysis. IEEE Transactions on Robotics **30**(4), 903–918 (2014)
3. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Machine learning **47**(2), 235–256 (2002)
4. Bhattacharyya, R.P., Senanayake, R., Brown, K., Kochenderfer, M.J.: Online parameter estimation for human driver behavior prediction. In: 2020 American Control Conference (ACC). pp. 301–306 (2020)
5. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. pp. 11621–11631 (2020)
6. Capito, L., Weng, B., Ozguner, U., Redmill, K.: A modeled approach for online adversarial test of operational vehicle safety (2020)
7. Chankong, V., Haimes, Y.Y.: Multiobjective decision making: theory and methodology. Courier Dover Publications (2008)
8. Chitta, K., Prakash, A., Geiger, A.: Neat: Neural attention fields for end-to-end autonomous driving. In: ICCV. pp. 15793–15803 (2021)
9. Chitta, K., Prakash, A., Jaeger, B., Yu, Z., Renz, K., Geiger, A.: Transfuser: Imitation with transformer-based sensor fusion for autonomous driving. IEEE Transactions on Pattern Analysis and Machine Intelligence (2022)
10. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. Autonomous robots **31**(4), 299–316 (2011)
11. Cohon, J.L.: Multiobjective programming and planning, vol. 140. Courier Corporation (2004)
12. Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Proceedings of the 1st Annual Conference on Robot Learning. pp. 1–16 (2017)
13. Fridovich-Keil, D., Ratner, E., Peters, L., Dragan, A.D., Tomlin, C.J.: Efficient iterative linear-quadratic approximations for nonlinear multi-player general-sum differential games. In: ICRA. pp. 1475–1481 (2020)
14. Har-Peled, S.: Geometric approximation algorithms. American Mathematical Soc. (2011)
15. Hoerger, M., Kurniawati, H., Elfes, A.: A software framework for planning under partial observability. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). pp. 1–9. IEEE (2018)
16. Hoerger, M., Kurniawati, H., Kroese, D., Ye, N.: Adaptive discretization using voronoi trees for continuous-action pomdps. In: Proc. Int. Workshop on The Algorithmic Foundations of Robotics (WAFR) (2022)
17. Hoermann, S., Stumper, D., Dietmayer, K.: Probabilistic long-term prediction for autonomous vehicles. In: IV. pp. 237–243 (2017)
18. Hoermann, S., Stumper, D., Dietmayer, K.: Probabilistic long-term prediction for autonomous vehicles. In: IV. pp. 237–243. IEEE (2017)
19. Isom, J.D., Meyn, S.P., Braatz, R.D.: Piecewise linear dynamic programming for constrained pomdps. In: AAAI. vol. 1, pp. 291–296 (2008)

20. Jaeger, B., Chitta, K., Geiger, A.: Hidden biases of end-to-end driving models. In: ICCV. pp. 8240–8249 (2023)
21. Kalra, N., Paddock, S.M.: Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? Transportation Research Part A: Policy and Practice **94**, 182–193 (2016)
22. Kim, D., Lee, J., Kim, K.E., Poupart, P.: Point-based value iteration for constrained pomdps. In: IJCAI (2011)
23. Koren, M., Alsaif, S., Lee, R., Kochenderfer, M.J.: Adaptive stress testing for autonomous vehicles. In: IV. pp. 1–7 (2018)
24. Kurniawati, H., Yadav, V.: An online pomdp solver for uncertainty planning in dynamic environment. In: Proc. Int. Symp. on Robotics Research (2013)
25. Luo, R., Zhao, S., Kuck, J., Ivanovic, B., Savarese, S., Schmerling, E., Pavone, M.: Sample-efficient safety assurances using conformal prediction. In: International Workshop on the Algorithmic Foundations of Robotics. pp. 149–169. Springer (2022)
26. Mavrotas, G.: Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Applied mathematics and computation **213**(2), 455–465 (2009)
27. Najm, W.G., Smith, J.D., Yanagisawa, M., et al.: Pre-crash scenario typology for crash avoidance research. Tech. rep., United States. Department of Transportation. National Highway Traffic Safety ... (2007)
28. Global ncap, http://www.globalncap.org/
29. O' Kelly, M., Sinha, A., Namkoong, H., Tedrake, R., Duchi, J.C.: Scalable end-to-end autonomous vehicle testing via rare-event simulation. In: NeurIPS. vol. 31. Curran Associates, Inc. (2018)
30. Paine, M., Paine, D., Case, M., Haley, J., Newland, C., Worden, S.: Trends with ancap safety ratings and real-world crash performance for vehicle models in australia. Proceedings of 23rd ESV, Seoul (2013)
31. Pilipovic, M., Spasojevic, D., Velikic, I., Teslic, N.: Toward intelligent driver-assist technologies and piloted driving: Overview, motivation and challenges. In: Proceedings of the X International Symposium on Industrial Electronics (INDEL'14) (2014)
32. Poupart, P., Malhotra, A., Pei, P., Kim, K.E., Goh, B., Bowling, M.: Approximate linear programming for constrained partially observable markov decision processes. In: AAAI. vol. 29 (2015)
33. Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: CVPR. pp. 7077–7087 (2021)
34. Rempe, D., Philion, J., Guibas, L.J., Fidler, S., Litany, O.: Generating useful accident-prone driving scenarios via a learned traffic prior. In: CVPR. pp. 17305–17315 (2022)
35. Roijers, D.M., Whiteson, S., Oliehoek, F.A.: Point-based planning for multi-objective pomdps. In: Proceedings of the twenty-fourth international joint conference on artificial intelligence (IJCAI). vol. 2015, pp. 1666–1672. AAAI Press (2015)
36. Schulz, J., Hubmann, C., Löchner, J., Burschka, D.: Multiple model unscented kalman filtering in dynamic bayesian networks for intention estimation and trajectory prediction. In: ITSC. pp. 1467–1474 (2018)
37. Schulz, J., Hubmann, C., Löchner, J., Burschka, D.: Multiple model unscented kalman filtering in dynamic bayesian networks for intention estimation and trajectory prediction. In: ITSC. pp. 1467–1474. IEEE (2018)

38. Seshia, S.A., Sadigh, D., Sastry, S.S.: Formal methods for semi-autonomous driving. In: DAC. pp. 1–5. IEEE (2015)
39. Silver, D., Veness, J.: Monte-carlo planning in large pomdps. NeurIPS **23** (2010)
40. Soh, H., Demiris, Y.: Evolving policies for multi-reward partially observable markov decision processes (mr-pomdps). In: Proceedings of the 13th annual conference on Genetic and evolutionary computation. pp. 713–720 (2011)
41. Sun, H., Feng, S., Yan, X., Liu, H.X.: Corner case generation and analysis for safety assessment of autonomous vehicles (2021)
42. Thorn, E., Kimmel, S.C., Chaka, M., Hamilton, B.A., et al.: A framework for automated driving system testable cases and scenarios. Tech. rep., United States. Department of Transportation. National Highway Traffic Safety . . . (2018)
43. Undurti, A., How, J.P.: An online algorithm for constrained pomdps. In: ICRA. pp. 3966–3973 (2010)
44. Wang, T., Ye, W., Geng, D., Rudin, C.: Towards practical lipschitz bandits. In: Proceedings of the 2020 ACM-IMS on Foundations of Data Science Conference. pp. 129–138 (2020)
45. Weng, B., Rao, S.J., Deosthale, E., Schnelle, S., Barickman, F.: Model predictive instantaneous safety metric for evaluation of automated driving systems. In: IV. pp. 1899–1906 (2020)
46. Wongpiromsarn, T., Karaman, S., Frazzoli, E.: Synthesis of provably correct controllers for autonomous vehicles in urban environments. In: ITSC. pp. 1168–1173 (2011)
47. Wu, P., Jia, X., Chen, L., Yan, J., Li, H., Qiao, Y.: Trajectory-guided control prediction for end-to-end autonomous driving: A simple yet strong baseline. NeurIPS **35**, 6119–6132 (2022)
48. Zhang, Q., Hu, S., Sun, J., Chen, Q.A., Mao, Z.M.: On adversarial robustness of trajectory prediction for autonomous vehicles. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15159–15168 (2022)