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 a POMDP-based method to generate colliding trajectories for the purpose of safety assessment. The proposed mechanism and method can be applied to any type of robot, but in this paper, we focus on autonomous cars. Given a range of autonomous robots with the same hardware capability but different software, our safety assessment mechanism outputs a ranking of the robots in terms of their safety. Many safety assessment mechanisms rely solely on accident data, which is rare. In contrast, to provide the safety ranking fast, we rely on the observation that in general, the smaller the distance between adversaries' safe trajectories and their closest colliding trajectories, the less safe the autonomous car being assessed is, because there is less margin for errors. To compute colliding adversarial trajectories that are close to the adversaries' safe trajectories, we take into account that the driving strategy of the vehicle being assessed is not fully known, and therefore propose a Multi-Objective Partially Observable Markov Decision Process (MOPOMDP) framing of the problem. To solve this MOPOMDP problem fast, we propose an on-line planning method, called Constraint-Aware Tree (CAT). Evaluations of CAT on the nuScenes dataset indicate that for the purpose of generating collision trajectories, CAT outperforms STRIVE, which is a state-of-the-art learning-based method that was also designed to alleviate the data-scarcity problem. Moreover, evaluations of four 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 safety assessment mechanism.

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 assessment to help inform consumers. Specifically, given a set of multiple autonomous robots with similar hardware capabilities but different autonomous software, our goal is to rank the safety of these autonomous robots under a given set of testing

scenarios. The software system (e.g., the perception, planning, and control) of the robot is treated as a black-box, 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 [30]. This rating system originally focused on the physical structure of the vehicle, such as the famous crash test. In recent years, NCAP has expanded its testing to accommodate autonomous technologies. However, it still relies on hand-crafted adversary strategies that are not adaptable to the behaviors of the cars being assessed [32]. 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 autonomous car technologies with level 3 autonomy and above.

Methods to 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., [52]). However, large datasets for such training are generally difficult to obtain because accidents, and more importantly catastrophic accidents, are rare, which is fortunate for users, but leads to limited accident data for safety assessment [22]. To alleviate the data-scarcity problem, machine learning methods such as STRIVE [35] have been designed to generate collision scenarios by learning only from data of safe (non-accident) scenarios, which is abundant. However, it still requires a long training time, which will likely hinder frequent assessment such as every time the robot's software is updated.

To alleviate the above difficulties, 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 accidents. This 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 adversaries' safe trajectories and their associated colliding trajectories indicates the relative safety of the assessed autonomous car, where smaller average distance indicates less safe vehicles.

The proposed safety assessment mechanism requires 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 adversarial 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 propose a novel online planning method,

called Constraint-Aware Tree (CAT), to approximately solve such a POMDP fast. CAT transforms the multi-objective POMDP into multiple Constrained-POMDPs (CPOMDPs), where the difference between them is only in the values of the constraints. Exploiting the features of the constraints and the similarity between the different POMDP problems, CAT computes a good solution for all the CPOMDP problems at once.

We evaluate our proposed POMDP solver CAT and our testing mechanism in simulation. We evaluate CAT by generating collision trajectories on complex vehicle-vehicle traffic scenarios, on the nuScenes dataset [5]. We compare it against STRIVE [35], a state-of-the-art learning-based method designed to alleviate the issue of rare accident data in safety assessment. The results indicate that CAT can generate significantly more collision trajectories while abiding traffic rules and uses much less computational time. To validate our proposed safety assessment mechanism, we benchmark our results against the ranking of top-rated autonomous driving software submitted to the CARLA Driving Leaderboard [13]. Evaluation on two scenarios: pedestrian crossing and lane merging, which are part of the National Highway Traffic Safety Administration (NHTSA) testable cases and scenarios [45], 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.

In short, our contributions are three-fold: (i) a mechanism to rank the safety of autonomous vehicle systems, (ii) a POMDP framing of the adversary's policy that accounts for the partial observability of the assessed vehicle's behavior and internal reasoning, and (iii) a POMDP-based method to automatically compute colliding adversarial trajectories that are close to safe trajectories.

2 Background and Related Work

2.1 Autonomous Car Verification and Testing

The review in [19] covers a wide range of general testing approaches and simulation-based verification techniques. Work such as [2, 40, 50] explores the use of formal methods for autonomous car system verification. These methods require complete formal specifications to be accurate, and are often difficult to construct completely due to the complexity of autonomous car systems. Other work such as [7, 31, 44, 49] attempts to generate test scenarios leading to accidents within their assessment mechanism. A common difficulty with such methods is the rare event nature of accidents, causing accident scenarios difficult to generate. Therefore, [18] and [39] aim to estimate and predict future trajectories to form scenarios using particle filtering and probabilistic methods. In [6, 46, 52], accident scenarios are created by generating collision trajectories to attack the target vehicle. The work in [27] proposes a framework based on conformal prediction to guarantee a low false negative rate in a driver alert system using only a few data points, but still relies on the need for data from unsafe trajectories. Due to the scarcity of such data, an adversarial optimization method called STRIVE

[35] is proposed to produce collision scenarios by only using data from safe trajectories. It generates collision trajectories by optimizing the latent space of a learned traffic model in the form of a graph-based conditional VAE.

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, and can adopt work from [4, 14, 38]. Work on pursuit-evasion, such as [11] and [25], will assist with the problem formulation for adversarial trajectory generation.

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, A is the set of actions, and O is the set of observations. The notation $T(s, a, s')$ is the transition function denoting the conditional probability $P(s'|s, a)$ of the agent landing in state $s' \in S$ after performing action $a \in A$ from state $s \in S$, whilst $Z(s', a, o)$ is the observation function denoting the conditional probability $P(o|s', a)$ the agent perceives observation $o \in O$ after executing action $a \in A$ and lands in state $s' \in S$. The notation $R(s, a)$ is the reward function. At any given time, a POMDP agent is at a state $s \in S$, but it never actually knows which state it is in. In POMDP, this uncertainty is represented as a distribution over states, called a belief and denoted as b. The set of all possible beliefs is called the belief space and is denoted as $\mathcal{B}$. The notion b_0 is the agent's initial belief and $\gamma \in (0, 1)$ is the discount factor to ensure the objective function is well defined.

The solution to a POMDP problem is a policy π that 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 $\pi^* = \arg\max_{\pi \in \Pi} V(b, \pi)$, where Π is the set of policies defined as mappings from beliefs to actions.

2.3 Multi-Objective POMDPs (MOPOMDPs)

Few methods have been developed to specifically solve MOPOMDPs. A common way to solve MOPOMDPs [42] is to convert the vector-valued reward function to a scalar function via weight scalarization, and solve as single-objective POMDPs. The difficulty here is obviously in how to set the right weight. The work in [37] proposes 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.

An alternative method is based on the ϵ-constraint method (described in Section 2.4), where some of the objectives are set as constraints, forming a set of Constraint-POMDP (CPOMDP) problems. A 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.

In general, CPOMDPs are harder to solve compared to POMDPs. Multiple works have extended approximate POMDP solvers to CPOMDP solvers, such as PBVI [23], dynamic programming [20], approximate linear programming [33] or online search [47]. The approximate point-based method shown in [23] uses PBVI with admissible costs to obtain optimal policies in CPOMDPs. In [20], an exact dynamic update for CPOMDPs using a piecewise linear representation of the value function is proposed. The work in [47] 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. The above methods are agnostic to the particular type of constraints. For certain types of constraints, one can actually exploit the features of the constraints to make solving the particular CPOMDPs easier. Our proposed method, CAT, adopts the ϵ-constraint method and this second approach to compute CPOMDPs (see Section 4.1 for details).

2.4 The ϵ-constraint Method

Suppose we have a multi-objective optimization problem [28]:

$$\max\ (f_1(x), f_2(x), \cdots, f_p(x)) \text{ s.t. } x \in S \tag{1}$$

where x is the vector of decision variables, $f_1(x), \cdots, f_p(x)$ are the p objective functions, and 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 [8, 12]:

$$\max\ f_1(x) \text{ s.t. } f_2(x) \leq e_2, \cdots, f_p(x) \leq e_p, x \in S \tag{2}$$

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 richer 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.

2.5 Fréchet Distance

Fréchet distance [15] 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:

$$d(P, Q) = \inf_{\alpha,\beta} \max_{t\in[0,1]} ||P(\alpha(t)) - Q(\beta(t))|| \tag{3}$$

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

For computational efficiency, in this paper, we use the approximation of Eq. (3) 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:

$$\begin{aligned} &d(P', Q') = \min_{k,l} \max_{t} ||p_{k(t)} - q_{l(t)}||, \text{ where} \\ &(i)\ k(t+1) = k(t) + 1 \text{ and } l(t+1) = l(t), \text{ or} \\ &(ii)\ k(t+1) = k(t) \text{ and } l(t+1) = l(t) + 1 \end{aligned} \tag{4}$$

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].

3 The Safety Assessment Mechanism

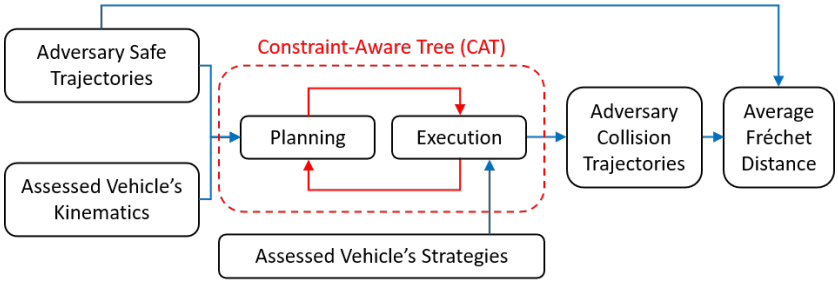


Fig. 1. Proposed safety assessment mechanism

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 standards, 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 and A_c is the set of actions the assessed vehicle can perform. $F_c(s_c, a_c, t, \Delta_t, \mathcal{P}_c)$ 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 safety assessment mechanism knows S_c and A_c of the assessed car AV, but it does not know AV's internal model $F_c(s_c, a_c, t, \Delta_t, \mathcal{P}_c)$ nor the exact action AV takes at any given time and state, but can construct an internal model based on the observed behavior. This internal model is of course not exact, and therefore the assessment mechanism considers the vehicle's 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, in this work, the mechanism computes a set of safe trajectories of the joint adversaries adv_i ($i \in [1, k]$), denoted as Φ. Here, we refer to safe trajectories as non-colliding trajectories between the adversaries and the assessed vehicle. In the assessment mechanism, it is straightforward to replace the safe trajectory generation with safe trajectory data provided by governing bodies or collected by various means. 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.

Specifically, the POMDP of the joint adversaries adv_i ($i \in [1, k]$) can be defined as $P = \langle S, A, O, T, Z, R, b_0, \gamma \rangle$. The state space $S_c \times \Pi_{i=1}^{k} S_{adv_i}$, and action space $A = \Pi_{i=1}^{k} A_{adv_i}$. The transition function T follows the dynamic functions F_c and F_{adv_i}. The observation space and function of the adversaries depend on the type of sensors equipped. Here, the imperfect information of the assessed vehicle's location is observed. The reward function R is designed to encourage collision and is set such that a high reward is given if the adversary collides with the assessed vehicle. Here, we are trying to achieve 2 objectives at the same time, colliding with the assessed vehicle while being close to the safe trajectory. We propose CAT (Section 4), which is designed to compute a policy quickly for this type of multi-objective POMDPs. We then execute the policy in simulation to generate the collision trajectories.

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. (4)) between the safe and 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. As-

Algorithm 1 CAT Online Planning ($\{\epsilon_{min}, \cdots, \epsilon_{max}\}$)

while not in collision AND not timeout **do**
  Select distance constraint set $\{\epsilon_1, \epsilon_2, \cdots, \epsilon_n\}; \epsilon_1 = \epsilon_{min}, \epsilon_n = \epsilon_{max}$
  Initialize POMDP problem $\mathcal{P}_{\phi,\epsilon_i}$ for $i \in [1, n]$
  while there is still time **do**
    Select i for $\mathcal{P}_{\phi,\epsilon_i}$
    Sample Episode $(\mathcal{P}_{\phi,\epsilon_i}, \mathcal{T}, b)$
    Backup and Update Statistics
  end while
  Execute Best Action a
end while

suming the set of safe and colliding adversaries' trajectories used in the distance computation is sufficiently representative, an autonomous car with smaller δ will likely have less buffer to avoid accidents compared to those with larger δ.

Note that $F(\phi, \omega)$ is the average distance between safe and colliding trajectories of the adversaries, which can be pedestrians, cars, or other traffic participants. Hence, for simplicity, the underlying metric used to compute the Fréchet distance is Euclidean distance, though, of course, other metrics are possible too.

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 [28], as detailed in Section 2.4.

Specifically, CAT sets the distance to a safe trajectory $\phi \in \Phi$ 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 at least equal to a given threshold. The key to CAT is that although conceptually, it computes the best solution of n CPOMDPs, CAT computes all of them at once, utilizing prior computations as much as possible. Algorithm 1 shows the overall procedure of CAT, whilst 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. (4) (s can be viewed as a degenerate polygonal chain). Moreover, any state $s \in S$ where the constraint $C(s)$ is not satisfied is modeled as a terminal 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 become 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 the 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 substructure property of the problem: The set of reachable beliefs of the POMDPs associated with smaller distance constraints is a subset 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 reuse prior computations and compute the solution to all n POMDPs at once.

4.2 Belief Tree Construction and Backup

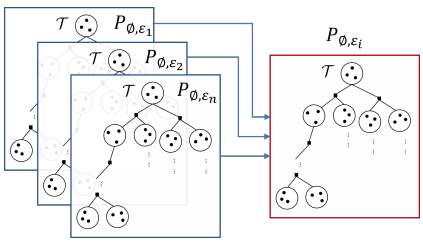


Fig. 2. CAT uses only a single belief tree.

Similar to almost any online POMDP solver today, CAT constructs a belief tree. 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 from b to b' in $\mathcal{T}$ that is labeled with a pair of action–observation (a, o) where $a \in A$ and $o \in O$ 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 the value functions $[V_1(b), V_2(b), \cdots, V_n(b)]$, and at its out-edge with label $(a, *)$, it maintains Q-value estimates $[Q_1(b, a), Q_2(b, a), \cdots, Q_n(b, a)]$, where V_i and Q_i for $i \in [1, n]$ denotes the corresponding estimated values 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, 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 (line 5 of Algorithm 1) using a bandit algorithm based on [48]:

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

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 problem $\mathcal{P}_{\phi,\epsilon_i}$, CAT can use any on-line tree-based POMDP solvers, such as POMCP[41], DESPOT[43], or ABT[26], to sample an episode (line-6 of Algorithm 1). In this paper, CAT uses ABT. Details of the episode sampling are in Appendix A.

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 a large negative reward is received 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 [26] and evaluation in [17]. 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

To select the set of distance constraints $\mathcal{E}$, 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. For this purpose, we compare CAT with an on-line POMDP solver, ABT – an online POMDP solver used for the episode sampling of CAT–, and STRIVE – a machine-learning method designed to alleviate the scarcity of accident data, on the nuScenes dataset. The second purpose of our experiments is to test the viability of the proposed safety assessment mechanism –based on the average Fréchet distance of safe and colliding adversaries trajectories. To this end, we design testing scenarios based on the NHTSA pre-crash scenario typology [29] within CARLA Driving Simulator and benchmark top-performing controllers submitted to the CARLA leaderboard.

CAT and ABT are implemented using the software toolkit OPPT [16]. For STRIVE, we use the authors' code and the accompanying trained model, available in [36]. If we were to retrain the model on our machine, it would have taken approximately 150 hours to complete the training. The adversarial optimization process of STRIVE is ran on a machine with a 16 Core AMD EPYC 7282 CPU, RTX 3090 GPU and 64GB DDR4 RAM. All other experiments are performed on a machine with an 8 Core Intel Xeon Silver 4110 CPU, RTX 2070 GPU, and 128GB DDR4 RAM.

5.1 Effectiveness in Generating Collision Trajectories

Experimental Setup We evaluate CAT, ABT, and STRIVE on nuScenes dataset, a public autonomous driving dataset containing a diverse set of traffic scenes collected from Singapore and Boston. Specifically, we evaluate the success rate and time for generating collision trajectories between one adversary and the vehicle being assessed while navigating through traffic and avoiding collision with all other vehicles and the environment.

For fairness, the selected adversary used for evaluation is the same for all methods, which is the single adversary selected by STRIVE to be controlled and collided with the vehicle being assessed. In all experiments, this selected adversary vehicle uses the same kinematics model and parameters as the ones used in STRIVE. The kinematic model is based on a 2D bicycle model. STRIVE uses continuous action space, but the POMDP-based methods use 25 discrete actions, discretized uniformly from the continuous action space used by STRIVE.

All methods are evaluated on the same 368 scenes that STRIVE uses for evaluation but, we excluded 41 scenes where the agent's initial state is already in collision with the environment from our analysis.

Table 1. Collision trajectory generation comparison on nuScenes dataset

Method	t (s)	Scenes where collision trajectory is present	Collision rate (%)	Fréchet distance, δ
Offline				
STRIVE (reported)	373 ± 22	162/327	-	-
STRIVE (validated)	373 ± 22	142/327	-	9.936
Online				
ABT (w=1)	3	192/327	86.88 ± 3.688	8.022 ± 5.686
ABT (w=5)	3	179/327	90.66 ± 3.273	6.930 ± 4.303
ABT (w=10)	3	146/327	79.52 ± 4.766	4.228 ± 3.029
CAT	3	205/327	88.06 ± 3.349	7.233 ± 4.594
CAT	6	211/327	88.25 ± 3.258	7.336 ± 5.064
CAT	9	211/327	90.03 ± 2.916	7.249 ± 4.992

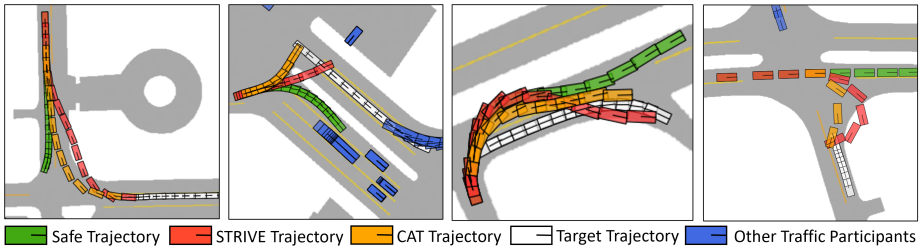


Fig. 3. Examples of invalid collision trajectories generated by STRIVE incorrectly labeled as successful. CAT finds valid collision trajectories in these scenes. The target trajectory is the trajectory of the identified target vehicle (the vehicle being assessed).

Since the POMDP-based methods are sampling-based, CAT and ABT repeat evaluations 10× on each scene with a pre-set planning time per step. We ran STRIVE once on each scene, as it is non-stochastic.

ABT accounts for the multi-objective nature of the problem via weights scalarization, i.e., the reward for moving one step is $r_{step} = w \cdot r_{base} \cdot d_{l2}$, where r_{base} is the base step penalty of -1 and d_{l2} is the Euclidean distance to the safe trajectory. The values of w used are 1,5,10. We ran a small number of preliminary runs to find suitable w for ABT, which we then used for evaluation.

CAT sets +1,000 reward for successful collision, -1000 for collision with other vehicles and the environment, and -1 step cost. We use $n_\epsilon = 10$ for all scenes.

Results and Discussion Table 1 presents the number of scenes where a collision trajectory is generated by the particular method, the average collision rate out of the 10 runs in scenes where at least one collision trajectory is found for POMDP-based solvers, and the average Fréchet distance.

Overall, CAT can generate collision trajectories in more scenes than ABT and STRIVE. STRIVE generates collision trajectories for 162 scenes (STRIVE (reported)), but in 20 of these scenes, the collision trajectories violate traffic rules. For instance, the agent cuts across road dividers and goes off-road such as those presented in Fig. 3. The numbers in STRIVE (validated) excluded such false positive scenes. ABT's ability to generate collision trajectories in varying

scenes highly depends on the weight w in its reward, though CAT still outperforms ABT. The collision rate of ABT is less influenced by the weight and is similar to the collision rate of CAT, which is relatively high. These results indicate that the collision trajectories generated in each scene are unlikely due to mere luck. The collision trajectories generated by CAT have the lowest δ unit distance at 7.233, compared to STRIVE's 9.936 and ABT's 8.022.

In terms of compute time, CAT requires much less computation time than STRIVE. The time presented in Table 1 for STRIVE refers to only the offline adversarial optimization process –that is, the time to generate a collision trajectory for the selected adversarial vehicle. Aside from the many days needed to train the traffic model, each collision trajectory STRIVE generates requires an adversarial optimization process, which on average takes 373 seconds. Comparatively, since CAT only needs an online planning time per step of 3 seconds, with average steps per scene of 8, in total, CAT requires less than 30 seconds on average to find a collision trajectory for each scene. These results indicate CAT can generate collision trajectories much faster than STRIVE without large amounts of prior data and computation.

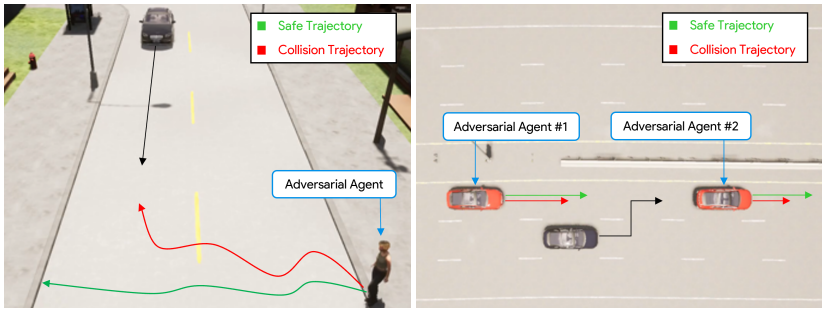
Overall, the results indicate CAT's effectiveness in solving the multi-objective problems by being able to generate a larger amount of collision trajectories while at the same time having these collision trajectories being closer to the safe trajectories. These results indicate the viability of CAT to alleviate the issues caused by rare collision data in assessing the safety of autonomous vehicles.

5.2 Benchmarking Autonomous Driving Software in CARLA

Experimental Setup To evaluate the effectiveness of our proposed safety assessment mechanism, we implemented and tested our safety assessment mechanism on 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 systems take as input the measurements from the various sensors (Lidar, Radar, RGB sensors) on the vehicles and are trained using Deep Learning methods to navigate through various traffic scenarios. We use the pre-trained weights provided by the respective authors for the experiments.

We use the top four autonomous driving software systems in CARLA Leaderboard: **TCP** [51], **NEAT** [9], **AIM** [34], and **TF++** [21]. A brief description of these software is presented in Appendix B. The testing scenarios are as follows:

Scenario 1: Pedestrian Crossing. This scenario is derived from a 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 it. 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 veloc-



(a) (b)

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

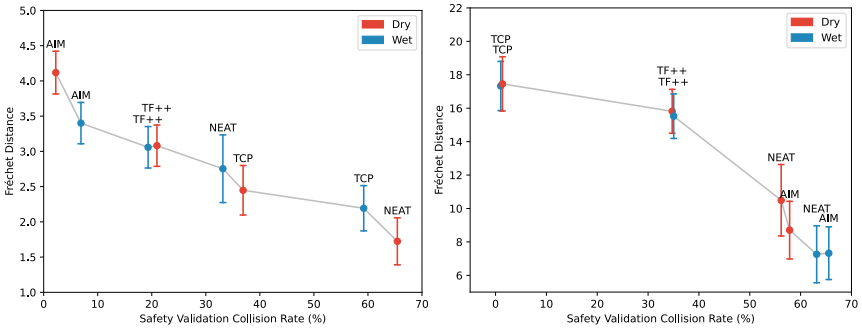
ity of $2.5m/s$. The pedestrian's motion is deterministic and it perceives a noisy observation of the distance between itself and the assessed vehicle.

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 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 scenario is much more challenging compared to Scenario 1, as CAT is controlling 2 agents instead of only 1, resulting in a much larger action space.

For each testing scenario above, we also evaluate them on wet and dry road conditions, resulting in 4 effective testing scenarios. In all scenarios, the experiments were conducted on a set Φ of 10 different safe trajectories. For each scenario and each safe trajectory $\phi \in \Phi$, CAT generates a set $\Omega(\phi)$ of 50 colliding trajectories with 10 seconds of planning time per step. The average Fréchet distance of these sets of trajectories is then used to rank the safety of the four software systems in each scenario. To establish a baseline of safeness for each software system and each scenario, we simulate 5,000 runs (50 runs for each of 100 random safe trajectories). A higher average collision rate indicates the software system is generally more dangerous in the tested scenario.

Results and Discussion Fig. 5 presents the results of our proposed safety assessment mechanism. For comparison, Table 2 presents the baseline and the Driving Score of the four software systems in the CARLA Leaderboard, where a higher score indicates higher infraction cases, which can indicate a less safe system. In all scenarios, the safer the software system is (given by the lower collision rate from validating random trajectories), the higher the Fréchet Distance, δ obtained by our proposed safety assessment mechanism, indicating the vehicle has larger room to account for errors and uncertainty. This consistent trend indicates the effectiveness of our proposed safety assessment mechanism.

The results indicate the Driving Score (Table 2) in the CARLA Leaderboard 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



(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

that for Scenario 2, our proposed mechanism is aligned with the safety indicator of the CARLA Leaderboard (Collision vehicles in Table 2). However, the trend is different in Scenario 1. In our particular test scenario, the only agent in the scene is the pedestrian and the assessed car. In the CARLA benchmark routes, there are commonly many other surrounding vehicles, causing the assessed car to travel at a lower average velocity. This enables the car to perform braking faster to avoid collision with the pedestrian, resulting in a lower collision rate.

Furthermore, there are other potential scenarios that we have yet to test, such as the pedestrian emerging from behind an obstacle, or the pedestrian encounter occurring right after performing a turn. We believe that these are the potential causes to the discrepancy in the trend, and would require further investigation.

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 lower collision rate (see Fig. 5(a)). We observe NEAT exhibits a more conservative approach in wet conditions. It drives with a lower average velocity and performs braking earlier to avoid collision. This is an example of the unique behavior that CAT is able to identify using our proposed safety assessment mechanism.

In CARLA Leaderboard, assessment is performed by having the software system traverse routes on 2 secret maps containing different test scenarios. Users have reported discrepancies in reported results by resubmitting the open-sourced methods using released or retrained model files [21]. Since the evaluation is secret, the exact reasons for these fluctuations are 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 do not need to be a secret. Also, since our mechanism is able to efficiently generate collision trajectories, evaluations can be done quickly after every software update.

6 Summary

This paper proposes a general safety assessment mechanism and collision trajectory generator for autonomous robots. Although the proposed mechanism and method are general, in this paper, we focus on autonomous cars. Given a set of autonomous cars with the same hardware capability but different autonomous software systems, our safety assessment mechanism outputs a ranking on the safety of the autonomous systems. The safety indicator is based on the average Fréchet distance between adversaries' safe trajectories and adversaries' collision trajectories that are as close as possible to the safe trajectories. The smaller the distance, the more likely that a slight error in the assessed car's motion causes accidents, and hence the less safe the system is.

The proposed mechanism considers the software system of the assessed car as a black-box, and views its behavior and inner workings as partially observed. Therefore, key to this mechanism is a POMDP-based method to generate adversaries' collision trajectories that are as close as possible to a given safe trajectory, where the behavior and inner workings of the assessed vehicle are treated as partially observed variables. To this end, we propose an on-line method, called CAT.

Experimental results on nuScenes dataset indicate that CAT can generate collision trajectories at a higher success rate and faster compared to ABT, a state-of-the-art online POMDP solver, and STRIVE, a state-of-the-art Deep Learning based technique that has been designed to alleviate the difficulty due to data scarcity in generating collision scenarios. Our results also indicate that the proposed safety assessment mechanism can be applied to reasonably assess the safety of different autonomous driving software in CARLA Simulator.

The positive results of CAT indicate the benefit of explicitly representing uncertainty caused by our limited information on the car being assessed. Although CAT is a non-learning technique, these results should not be viewed as learning vs. non-learning techniques per se. Rather, as an indication of the potential benefit of understanding what approach works well for which component of the safety assessment mechanism. For instance, to generate the evolution of the entire road scenario, techniques such as STRIVE would likely be beneficial. Whilst, to generate collision trajectories between adversaries and the assessed vehicle, it is more critical to frame the behavior and inner workings of the assessed vehicle as partially observed variables.

Future works abound. For instance, it would be useful to explore more suitable alternatives to Fréchet distance. Expansion and application to assess the safety of other types of robots would be of interest too.

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

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) (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: CVPR. pp. 11621–11631 (2020)
6. Cao, Y., Xu, D., Weng, X., Mao, Z., Anandkumar, A., Xiao, C., Pavone, M.: Robust trajectory prediction against adversarial attacks. In: Conference on Robot Learning. pp. 128–137. PMLR (2023)
7. Capito, L., Weng, B., Ozguner, U., Redmill, K.: A modeled approach for online adversarial test of operational vehicle safety (2020)
8. Chankong, V., Haimes, Y.Y.: Multiobjective decision making: theory and methodology. Courier Dover Publications (2008)
9. Chitta, K., Prakash, A., Geiger, A.: Neat: Neural attention fields for end-to-end autonomous driving. In: ICCV. pp. 15793–15803 (2021)
10. 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)
11. Chung, T.H., Hollinger, G.A., Isler, V.: Search and pursuit-evasion in mobile robotics. Autonomous robots **31**(4), 299–316 (2011)
12. Cohon, J.L.: Multiobjective programming and planning, vol. 140. Courier Corporation (2004)
13. 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)
14. 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)
15. Har-Peled, S.: Geometric approximation algorithms. American Mathematical Soc. (2011)
16. Hoerger, M., Kurniawati, H., Elfes, A.: A software framework for planning under partial observability. In: IROS. pp. 1–9. IEEE (2018)
17. Hoerger, M., Kurniawati, H., Kroese, D., Ye, N.: Adaptive discretization using voronoi trees for continuous-action POMDPs. In: WAFR (2022)
18. Hoermann, S., Stumper, D., Dietmayer, K.: Probabilistic long-term prediction for autonomous vehicles. In: IV. pp. 237–243. IEEE (2017)
19. Ioannides, C., Eder, K.I.: Coverage-directed test generation automated by machine learning–a review. ACM Transactions on Design Automation of Electronic Systems (TODAES) **17**(1), 1–21 (2012)
20. Isom, J.D., Meyn, S.P., Braatz, R.D.: Piecewise linear dynamic programming for constrained POMDPs. In: AAAI. vol. 1, pp. 291–296 (2008)

21. Jaeger, B., Chitta, K., Geiger, A.: Hidden biases of end-to-end driving models. In: ICCV. pp. 8240–8249 (2023)
22. 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)
23. Kim, D., Lee, J., Kim, K.E., Poupart, P.: Point-based value iteration for constrained POMDPs. In: IJCAI (2011)
24. Kocsis, L., Szepesvári, C.: Bandit based monte-carlo planning. In: European conference on machine learning. pp. 282–293. Springer (2006)
25. Koren, M., Alsaif, S., Lee, R., Kochenderfer, M.J.: Adaptive stress testing for autonomous vehicles. In: IV. pp. 1–7 (2018)
26. Kurniawati, H., Yadav, V.: An online POMDP solver for uncertainty planning in dynamic environment. In: Proc. Int. Symp. on Robotics Research (2013)
27. Luo, R., Zhao, S., Kuck, J., Ivanovic, B., Savarese, S., Schmerling, E., Pavone, M.: Sample-efficient safety assurances using conformal prediction. In: WAFR. pp. 149–169. Springer (2022)
28. Mavrotas, G.: Effective implementation of the ε-constraint method in multi-objective mathematical programming problems. Applied mathematics and computation **213**(2), 455–465 (2009)
29. 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)
30. Global ncap, http://www.globalncap.org/
31. 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)
32. 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)
33. 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)
34. Prakash, A., Chitta, K., Geiger, A.: Multi-modal fusion transformer for end-to-end autonomous driving. In: CVPR. pp. 7077–7087 (2021)
35. 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)
36. Rempe, D., Philion, J., Guibas, L.J., Fidler, S., Litany, O.: Generating useful accident-prone driving scenarios via a learned traffic prior (cvpr 2022) (2022), https://github.com/nv-tlabs/STRIVE
37. Roijers, D.M., Whiteson, S., Oliehoek, F.A.: Point-based planning for multi-objective POMDPs. In: IJCAI. vol. 2015, pp. 1666–1672. AAAI Press (2015)
38. 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)
39. 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)
40. Seshia, S.A., Sadigh, D., Sastry, S.S.: Formal methods for semi-autonomous driving. In: DAC. pp. 1–5. IEEE (2015)

41. Silver, D., Veness, J.: Monte-carlo planning in large POMDPs. NeurIPS **23** (2010)
42. 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)
43. Somani, A., Ye, N., Hsu, D., Lee, W.S.: Despot: Online POMDP planning with regularization. NeurIPS **26** (2013)
44. Sun, H., Feng, S., Yan, X., Liu, H.X.: Corner case generation and analysis for safety assessment of autonomous vehicles (2021)
45. 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)
46. Turlej, W., Pankiewicz, N.: Adversarial trajectories generation for automotive applications. In: 2021 25th International Conference on Methods and Models in Automation and Robotics (MMAR). pp. 115–120. IEEE (2021)
47. Undurti, A., How, J.P.: An online algorithm for constrained POMDPs. In: ICRA. pp. 3966–3973 (2010)
48. 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)
49. 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)
50. Wongpiromsarn, T., Karaman, S., Frazzoli, E.: Synthesis of provably correct controllers for autonomous vehicles in urban environments. In: ITSC. pp. 1168–1173 (2011)
51. 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)
52. Zhang, Q., Hu, S., Sun, J., Chen, Q.A., Mao, Z.M.: On adversarial robustness of trajectory prediction for autonomous vehicles. In: CVPR. pp. 15159–15168 (2022)

A Sampling an Episode (line-6 of Algorithm 1)

Once a CPOMDP $\mathcal{P}_{\phi,\epsilon_i}$ to use is selected, CAT uses a POMCP-like method [41] to 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 $\mathcal{P}_{\phi,\epsilon_i}$. This strategy has been shown to enable convergence to the optimal policy [24, 41]. 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.

B Autonomous Driving Software used in Section 5.2

This section provides a short description of the software systems being used to evaluate the effectiveness of the proposed safety assessment mechanism.

TCP [51] 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 [9] 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 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 [34] 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++ [21] is an improved variant of Transfuser [10] by modifying its archi-

tecture, 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.