Scenario-based Safety Evaluation of Highly Automated Vehicles

by

Xinpeng Wang

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Mechanical Engineering)
in the University of Michigan
2023

Doctoral Committee:

Professor Gabor Orosz, Co-Chair
Professor Huei Peng, Co-Chair
Associate Research Scientist Tulga Ersal
Professor Ilya Kolmanovsky
Professor Henry Liu
Professor Jing Sun

Xinpeng Wang
xinpengw@umich.edu
ORCID iD: 0000-0001-8494-0494

DEDICATION

I dedicate this dissertation to my grandmother, Mrs. Huiqin Fang, who nursed me with her love and wisdom, and will be a lifelong inspiration for me.

ACKNOWLEDGMENTS

First, I want to express my deepest gratitude and remembrance to my late advisor, Professor Huei Peng. Professor Peng has been a father figure for me. He enlightened me on how to do good research with his patient guidance, insightful suggestions, and sometimes hard questions. He taught me how to be a good researcher with his devotion to work, enthusiasm for engineering, and curiosity about new topics. He is one of the most dependable, loving and noble persons I have ever met, which has inspired me to be my better self. Having Professor Peng as my advisor is the honor of my life. He will always be an inspiration and role model for me.

Next, I extend my heartfelt thanks to my committee members: Professor Gabor Orosz, Dr. Tulga Ersal, Professor Ilya Kolmanovsky, Professor Henry Liu, and Professor Jing Sun. Their insightful guidance has inspired me to approach my research topic from different aspects, and pushed me to think more deeply about problems. Specifically, after the passing of my advisor, they have been extremely supportive to me both academically and emotionally, and helped me through the darkest days.

I would like to thank Mcity, Ford Motor Company and Toyota Research Institute for funding my research and providing me with the best support and resources to realize my ideas. Without their help, I could never reach the point where I stand now.

Next, I want to thank my brothers and sisters in the Vehicle Dynamics Lab: Ding, Shaobing, Pingping, Yiqun, Steven, Yuxiao, Ziheng, Xianan, Su-Yang, Geunsoab, Songan, Nauman, Boqi, Yuanxin, Minghan, Lu, Han, Tinghan, Zhong, and Juhui. They are not only wonderful research partners, but also irreplaceable friends in my personal life. Thank you for being there with me through all the highs and lows.

I also want to thank all the amazing friends I met at U-M for all the wonderful moments we shared together, from my amazing roommate Zhen, to my great buddies Yingdong, Fucong, Ting, Zhuo, Minghao and Jiankan. Because of you, my life in Ann Arbor has been so colorful.

Finally, my greatest appreciation goes to my family for their unconditional support and love. I would like to thank my mother Jun Hu, my father Junming Wang, and my grandparents Huiqin Fang and Zhiqiang Hu, who have made me the person I am. I also want to thank my fiancée, Jiang Mao, who is always by my side whenever and wherever.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURE

LIST OF TABLES

TABLE

LIST OF ABBREVIATIONS

ACC	Adaptive cruise control.
ADAS	Advanced driver assistance system.
AEB	Autonomous emergency braking.
BIC	Bayesian Information Criterion.
BMB	Behavioral mode boundary.
BRT	Backward reachable tube.
BSM	Basic safety message.
CA	Constant-acceleration.
CMC	Crude Monte-Carlo.
CP	Control Point.
CV	Constant-velocity.
DDQN	Double deep Q-network.
DQN	Deep Q-network.
EI	Expected improvement.
EM	Expectation–maximization.
FCP	Failure case probability.
FMC	Failure mode coverage.
GIDM	Generalized Intelligent Driver Model.
GP(R)	Gaussian process (regression).
GPS	Global Positioning System.
HAV	Highly automated vehicle.
IDM	Intelligent Driver Model.
IRL	Inverse reinforcement learning.
ISD	Importance sampling distribution.
LQR	Linear–quadratic regulator.
LTAP/OD	Left turn across path / opposite direction.
MAB	Multi-arm bandit.
MDP	Markov decision process.
MG	Markov game.
MLE	Maximum likelihood estimation.
MPC	Model Predictive Control.
NCAP	New Car Assessment Program.
ND	Naturalistic distribution.
N-FOT	Naturalistic Field Operational Test.

OCP	Optimal control problem.
ODD	Operational design domain.
OEDR	Object and event detection and response.
OV	Opponent vehicle.
PDF	Probability density function.
PID	Proportional–integral–derivative.
PMP	Pontryagin maximum principle.
PORU	Primary other road user.
POV	Primary other vehicle.
PSP	Parameterized speed profile.
PV	Primary vehicle.
RL	Reinforcement learning.
RLS	Risk level set.
ROS	Robotic Operating System.
ROW	Right-of-way.
RTK	Real-time kinematic positioning.
SVO	Social value orientation.
(T)GMM	(Truncated) Gaussian mixture model.
TS	Thompson sampling.
TTC	Time-to-collision.
UCB	Upper confidence bound.
ULT	Unprotected left turn.
VRU	Vulnerable road user.
VUT	Vehicle Under Test.

ABSTRACT

A highly automated vehicle (HAV) is a safety-critical system. Therefore, a verification and validation (V&V) process that rigorously evaluates the safety of HAVs is necessary before their mass deployment on public roads. This dissertation will present the methodology and implementation procedure of a scenario-based evaluation framework for HAVs.

First, an evaluation framework for reactive scenarios is proposed, where the risk level of test cases could be objectively categorized in advance. The pedestrian crossing scenario is used as a case study. We first build a statistical model for the pedestrian scenario based on naturalistic data. Next, reachability analysis is applied to partition the scenario testing space into different risk level sets, which are then combined with importance sampling to generate test cases efficiently and realistically. The proposed method achieves unbiased crash rate estimation in an accelerated fashion, while all the test cases are feasible and have controlled risk levels.

Then, a novel evaluation framework for interactive scenarios is proposed, including highway merging and roundabout entering. Instead of assuming that the primary other vehicle (POV) takes predetermined maneuvers, we model the POVs as game-theoretic agents. To capture a wide variety of interactions between the POV and the vehicle under test (VUT), we use level-k game theory and the social value orientation (SVO) concept to model the POV, and generate a diverse library of POV policies using reinforcement learning. On the other hand, an adaptive test case generation method is developed based on adaptive sampling, stochastic optimization and upper confidence bound (UCB) algorithm to generate customized challenging cases for the VUT from the testing space. In simulations, the proposed POV library captures a wide range of interactive patterns for both highway merge and roundabout entering scenarios. The proposed test case generation method covers the failure modes of a black-box VUT more effectively compared to other approaches.

In addition, the problem of generating corner cases for interactive scenarios systematically is considered. an ambiguity-guided adversarial planning algorithm is developed to generate confusing behaviors for the primary other road user (PORU) in interactive scenarios. The PORU is modeled as a cost-minimizing agent with hierarchical intentions. The adversarial PORU plans actions to confuse the HAVs by maximizing the ambiguity with respect to its intentions, while also taking nominal behavior planning goals into consideration. Two interactive scenarios are studied: highway merging and pedestrian crossing. A corner case testing scheme is designed and imple-

mented for both scenarios to evaluate the performance of different HAVs comprehensively and objectively.

Finally, the procedure of implementing behavior competence testing in the real world is presented for reactive scenarios. Speed planning algorithms for the POV are developed to synchronize its motion with the VUT. A speed tracking controller for experimental vehicles is designed based on the preview control algorithm. It is demonstrated that tests can be executed for multiple reactive scenarios with real vehicles on the Mcity test track in an accurate, repeatable and automated fashion. In addition, a digital twin of Mcity in the CARLA simulator is created, and the same testing capability in the simulation is demonstrated.

CHAPTER 1

Introduction

1.1 Background and Motivation

The recent advancement of highly automated vehicles (HAVs) has reaffirmed the future of safer, cleaner, more efficient and more equitable ground transportation [2]. They are set to relieve humans from the tedious task of driving, and to transform the vista of future mobility. The Society of Automobile Engineers (SAE) defined six levels of driving automation in J3016 [1] in 2014, which categorizes the degree of automation from level 0 (no automation) to level 5 (full automation), as shown in Table 1.1. For level 2 and below, the automation needs to be supervised by the driver. Most of the driving automation systems on the market belong to this category, including Ford's BlueCruise, GM's Super Cruise and Tesla's Autopilot, etc. For level 3 automation, the vehicle is required to drive itself and conduct the task of Object and Event Detection and Response (OEDR) under certain conditions, while the human driver still needs to take over the vehicle when requested. Despite the difficulty with liability and the potential risk with driver take-over, level 3 self-driving has been granted regulatory approval in more markets. By the start of 2023, available level 3 systems include Honda Sensing Elite, Drive Pilot from Mercedes Benz [3]. For level 4 and above, which are referred to as highly automated vehicles (HAVs) in this dissertation, the vehicle should be able to drive fully autonomously without the intervention of the driver in some conditions (level 4) or all conditions (level 5).

HAVs have been the development focus of many companies. The robo-taxi has been a key application for HAVs. From the year 2020, Waymo has been providing ride-hailing service in Arizona, US. Multiple other companies have launched or are planning to launch commercialized robo-taxi services all over the world, as summarized in Table 1.2. Autonomous trucking and freight transportation is another key application, where companies including Waymo, TuSimple, Aurora, etc., are working on its commercialization [4].

Due to the absence of human supervision, the safety assurance of HAVs is an extremely crucial prerequisite for their wide development. In March 2018, a level 4 prototype HAV from Uber

SAE level	Sustained lateral and longitudinal control	OEDR	Fallback for dynamic driving task	ODD	Example features
0	Driver	Driver	Driver	n/a	AEB, BSW
1	Driver and System	Driver	Driver	Limited	ACC, LKA
2	System	Driver	Driver	Limited	ACC + LKA
3	System	System	Fallback-ready driver	Limited	Traffic jam assist
4	System	System	System	Limited	Locally driverless taxi
5	System	System	System	Unlimited	Fully driverless car

Table 1.1: SAE levels of automation for on-road motor vehicles [1].

Company	First launch time (est.)	Location
Waymo	2020	Arizona, US
Cruise	2022	California, US
Mobileye	(2023)	Germany & Israel
Motional	(2023)	Nevada, US
Baidu Apollo [6]	2021	Beijing, China
Momenta	2021	Shanghai, China

Table 1.2: Ride-hailing service launch status (by Q1 2023).

was involved in a fatal crash, which not only caused the shutdown of further testing from Uber ATG [5], but also led to a crisis on public trust in HAVs. Since HAVs are highly complex and safety-critical, the rigorous verification and validation (V&V) approach is crucial to prove their reliability and robustness and to gain public trust. Moreover, it is important for the V&V to be conducted by a government agency or a trusted third-party as acceptance testing, in addition to the existing self-certification process adopted by OEMs and technology start-ups.

1.2 Literature Review

First, there are two main different aspects of safety related to HAVs. The first one is functional safety, which is formalized in ISO 26262 for passenger vehicles [7]. It describes safety as the absence of risk arising from software or hardware failures. On the other hand, ISO 21448 defines the notion of Safety of the Intended Function (SOTIF), which focuses on the risk due to functional insufficiencies or foreseeable misuse by humans [8]. It characterizes the safety performance of an HAV assuming all the components are working as intended and free from bugs. For the scope of this dissertation, we focus on the latter aspect of safety.

On the other hand, since the notion of "operating in any conditions" for level 5 autonomy is more of an ideal vision than a practical target, almost all current HAV are developed to operate under a certain set of circumstances, which is named the operational design domain (ODD). Therefore, the domain of safety evaluation for HAVs will be their target ODD [9]. The ODD is formally defined as the "operating conditions under which a given driving automation system or feature thereof is specifically designed to function, including environmental, geographical, and time-of-day restrictions, and/or the requisite presence or absence of certain traffic or roadway characteristics" [1]. The ODD determines the environment that the HAV will operate in, and the set of scenarios that the HAV might encounter. The primary goal of the safety evaluation is to assess the performance of the HAV inside its ODD thoroughly.

1.2.1 Naturalistic Field Operational Testing

A straightforward way to cover the ODD is to deploy the HAV inside its target environment, and test with millions of naturalistic miles, hoping to traverse all the combinations of circumstances in the ODD. These are known as Naturalistic Field Operational Tests (N-FOTs). There have been many N-FOTs projects conducted by the government and research institutes. In 2001, the 100-Car Naturalistic Driving Study was conducted to collect large-scale, naturalistic driving data to understand the cause of crashes [10]. Over 2 million miles of data were collected. In 2005, the Integrated Vehicle-Based Safety Systems (IVBSS) program [11] was conducted to evaluate the effectiveness of various collision warning systems, with more than 800k miles driven with both light vehicles and heavy trucks. In 2012, the Safety Pilot Model Deployment (SPMD) [12] was conducted to study the effects of connected and automated vehicle technologies.

Recently, N-FOTs have been adopted by most technology companies working on self-driving vehicles. By Jan 2022, 51 entities have acquired autonomous vehicle testing permits in California, while 7 of them have driverless testing permits [13]. They have logged over 1.9 million miles annually in both 2019 and 2020. By 2021, the HAVs from Waymo have covered over 20 million test miles on public roads in total [14]. These N-FOTs present the HAVs with the most realistic driving environments, and have provided millions of miles of driving data for research and development purposes. Many famous open driving datasets come from such tests by technology companies, including the Waymo Open dataset (1150 scenes), the nuScenes dataset from Motional (1000 scenes) [15], the Argoverse dataset from Argo AI (113 scenes) [16], etc.

However, N-FOTs are extremely expensive and inefficient. Collisions are rare in real-world driving. According to NHTSA, the (police-reported) collision rate in the US is 2.1 collisions per 1 million miles, and 1.1 fatalities per 100 million miles [17]. It is estimated by Kalra et al. [18] that to demonstrate that HAVs are safer than human drivers in terms of fatality rate with 95 % confidence

level, 275 million miles need to be driven by HAVs without any fatal collision. Moreover, since the software and hardware on HAVs are updated regularly (monthly or even weekly), testing each new version with hundreds of millions of miles is even more costly if not unrealistic.

1.2.2 Scenario-based Safety Evaluation

A more efficient testing approach is to decompose the ODD into single scenarios that are tractable for repeated testing and comprehensive assessment. The scenario-based evaluation is an accepted best practice for the V&V of HAVs [19]. There have been many international collaborative efforts on creating a scenario-based evaluation framework. In Germany, the PEGASUS Family includes multiple projects that aim to create methodologies, toolchains and simulation platforms for the testing of HAVs in both highway and urban environments [20]. In Japan, the Sakura project [21] aims at creating scenario-based safety assurance methods as well as a scenario database.

According to [22], a scenario is formally defined as "a temporal development between several scenes in a sequence of scenes", where a scene is defined as a snapshot of the environment, which includes both the static scenery and dynamic objects. Subsequently, the PEGASUS project [20] characterize scenarios into three levels of abstraction [23], with increasing level of details: functional scenarios, logical scenarios and concrete scenario. In this work, we will refer to functional scenarios as "scenarios", and combine the concepts of logical scenarios and concrete scenarios into "test cases". To most researchers, the interesting and valuable scenarios are those with other traffic agents, especially when they have traffic conflicts with the HAV under test. Here, the conflict is defined as "a situation in which road users approach each other in space and time such that collision is imminent if their movements remain unchanged" [24].

How are test scenarios selected? In 2019, NHTSA summarized 36 pre-crash scenarios by analyzing data from two crash databases in the US from 2011-2015: the Fatality Analysis Reporting System (FARS) and National Automotive Sampling System (NASS) General Estimates System (GES) crash databases [25]. The most frequent pre-crash scenario is the rear-end scenario. Since these scenarios are from historical crash data, they represent the most typical failure modes of human drivers, which do not necessarily apply to the HAVs directly. In 2020, researchers from TNO and NTU described 67 scenarios to be considered in the safety assessment of automated vehicles [26] based on accident databases as well as structured analysis. Moreover, NHTSA [27] proposed multiple test scenarios based on three dimensions of behavior competency, including tactical maneuvers, OEDR capabilities and failure mode behaviors. Waymo [28] extended from NHTSA's recommendation and curated a set of 47 scenarios for its self-assessment tests. Based on [28] and multiple other sources, Mcity [29] compiled a list of 50 behavior competence scenarios for HAV acceptance testing in 2019. From this list, 35 scenarios were selected to be implementable

in the Mcity test track, an HAV test facility at the University of Michigan. which include 16 scenarios involving inter-vehicle interactions and 4 scenarios involving interactions with vulnerable road users (VRUs). In most of the scenarios with other vehicles, there is the vehicle under test (VUT) and one or more surrounding vehicles that interact with the VUT, denoted as the primary other vehicles (POVs). The concept of POV can be extended to primary other road user (PORU), which also include motorcyclists, cyclists, pedestrians, etc.

After the test scenario is selected, there are two key subsequent research questions to answer for conducting the scenario-based HAV evaluation [19]: scenario modeling (what to test?), and test case generation (how to test?), which will be reviewed next.

Scenario Modeling Techniques

To build a model for the test scenario, the focus lies on the parameterization of the scenario and the organizing structure of all possible test cases. [30] summarized a 5-layer model for the varying parameters: road layout, traffic infrastructure, temporary manipulation of road and infrastructure, object, and environment (weather, lighting, etc). They can be categorized as initial conditions (static parameters) and dynamic behaviors of traffic agents.

Many of the previous studies focused on initial conditions. [31] used initial conditions with defined ranges to characterize the car-following scenarios. [32, 33] selected initial conditions to describe the cut-in scenario and used probabilistic models learned from naturalistic driving data to describe the structure of the test cases based on their exposure frequency. [34] used a combination of exposure frequency and maneuver challenge to assign a criticality score to each test case. Though the patterns of naturalistic driving were considered, the temporal behaviors of the surrounding vehicles/VRUs were not captured. In [35], the trajectory of the POV was modeled with discretized control points, which then form the scenario space. [36, 37] both discretized the trajectory of the leading vehicle in a car-following scenario and formed the scenario space with the action sampling distribution of the leading vehicle. [38] additionally considered the sensor noise at each time-step. These models were able to generate different dynamic behaviors, but the dimension of the sampling space becomes very high. On the other hand, [39] applied level-k game theory to create a library of surrounding vehicle models that cover different styles of interactions between the HAV and the surrounding vehicles. However, they only generated a few discrete types of interaction models, whose behaviors are not sufficiently diverse.

On the other hand, data-driven road agent simulation has drawn a lot of attention recently. They aim to extract diverse and realistic driving/moving behaviors of traffic participants from large-scale driving datasets with machine learning generative models. SimNet was proposed in [40], which used behavior cloning to generate reactive driving behaviors for surrounding vehicles, which help avoid unrealistic collisions that often happened in simulation testing based on log replay. Traf-

ficSim [41] generated joint behavior for multiple agents with a graph-based trajectory prediction model. BITS [42] and Symphony [43] both adopted hierarchical approaches for agent-centric behavior generation, where the high-level module infers the intent/goal of each agent, while the low-level module generates concrete action conditioned on the goal. A different hierarchical model was adopted in TrajGen [44], which consists of a trajectory prediction stage, and a trajectory modification stage based on reinforcement learning. These simulation approaches are useful not only for evaluating HAVs in realistic traffic environments, but also for setting up training environments for reinforcement learning-based decision-making and planning algorithms [45].

Test Case Generation Techniques

For test case generation, the test matrix approach has been widely used to evaluate advanced driver assistance systems (ADAS) by organizations including Euro NCAP [46, 47] and IIHS [48]. Test cases are determined a priori by listing the combinations of several key parameters of the scenario, and each VUT will be tested with the same set of cases. The goal is to achieve good coverage of the given ODD in a uniform and deterministic way. The fairness of the test is easily guaranteed. However, the VUT can be tuned to pass the predefined test cases, while it may fail under broader conditions in the real world. Moreover, the samples required to cover the testing space grow exponentially with the number of modeling parameters. Some space-filling methods can improve the efficiency of coverage-oriented test case generation, including Latin hypercube sampling [49], covering array [50], etc. However, since these methods do not make use of any information about the VUT's responses, their performance is relatively poor compared with the adaptive sampling methods introduced below.

Monte Carlo sampling-based evaluation methods have been proposed to estimate the real-world performance of the VUT. Test cases are acquired by sampling from the parameter distributions. In [51], collision avoidance algorithms were evaluated in a stochastic environment created by an errorable driver model built from naturalistic driving data. However, since challenging scenarios with conflicts are rare events in the naturalistic driving data, crude Monte Carlo sampling is found to be inefficient. Therefore, many recent studies applied accelerated approaches to achieve an unbiased estimate of the performance criteria (e.g., crash rate) with a faster convergence rate. Importance sampling was applied in [32, 52–54], where the naturalistic distribution was "skewed" to emphasize more challenging test scenarios and to achieve more efficient test case sampling. In [33, 55], subset simulation was used to efficiently estimate the collision/injury rate and solved the difficulties of importance sampling method in high-dimensional spaces [56]. However, the amount of samples required to reach convergence for the performance criteria estimation is still prohibitively large for real-world field testing. Moreover, the test parameter space will include cases that are impossible for the HAV to avoid a collision, e.g. a human vehicle that cuts in front

of the HAV 3 meters ahead with a relative speed of 20 m/s. The presence of such test cases could make the results misleading and make the test dangerous.

On the other hand, falsification-based evaluation methods attempt to generate initial conditions or POV behaviors that force the VUT to violate the safety requirements with limited test runs. Corner cases have been generated using different techniques. [57, 58] used forward reachability analysis to categorize the challenge level of a test scenario, and generated failure cases by minimizing the size of the solution space for the VUT using quadratic programming [57] or evolutionary algorithm [58]. [59] generated challenging cases by sampling near the boundary of the control invariant set of the VUT, and synthesizing adversarial controller for the POV by solving a dual game. In [60], to find the optimally challenging test cases for a black-box system, the authors formulated and solved a mini-max problem based on control barrier functions estimated from collected system demonstrations and given specifications. These methods create worst-case conditions for the dynamic model of the VUT, which can only be applied to scenarios where the VUT is trying to avoid collisions passively.

Moreover, some studies generate customized adversarial test cases for given black-box VUTs. [35] utilized the falsification tool S-Taliro to find falsifying test cases using the simulated annealing method. [61, 62] utilized rapid-exploring random tree (RRT) to achieve a similar goal. However, they all generate POVs that can behave unreasonably adversarial, which is not informative for measuring the real-world performance of the VUT. Reinforcement learning (RL) has been another popular method recently for creating adversarial test cases [38, 63–65]. In [64], a special reward design encouraged the POV to involve in collisions where the VUT is at fault, which helped create adversarial yet socially acceptable POV behaviors. In [38, 65], the diversity of the generated challenging cases was addressed. The limitation of RL-based methods is that a simulation model of the VUT is required for training adversarial environment or the POV agent, which can be too inefficient or unrealistic for some use cases like government-conducted acceptance testing. On the other hand, adaptive sampling has been used to search for failure cases on the fly in [31, 66–69], where a surrogate model of the performance surface of the VUT in a scenario is updated with test results, which then guides the adaptive search for new test cases. The diversity of identified failure modes was addressed using region elimination [66], performance boundary heuristics [68], or better-designed acquisition functions [31, 69]. However, they did not consider the presence of categorical parameters in the testing space. Moreover, the coverage of the failure modes has not been formally defined or quantified.

In addition, some recent studies generate challenging test cases by transferring real-world driving data. In [70], safety-critical scenarios for LiDAR-based HAVs were created by perturbing the trajectories of the initial real-world scenario using Bayesian optimization. In [71], the criticality of the original scenario was elevated with an adversarial RL algorithm. In [72], corner cases

were synthesized by combining the safe driving data with collision data using a generative model. In [73], solvable yet challenging test cases were created by optimizing within the latent space of a generative trajectory prediction model, which was learned from real driving data.

1.2.3 Other Types of Evaluation Methods

Kapinski et al. [74] categorized the analysis techniques for embedded control systems into three types: testing, falsification and verification. Testing assesses the satisfaction of the system to given properties (e.g. never involved in a collision) within a set of test cases; falsification attempts to find cases where the satisfaction to a given property is violated by the system. The methods we reviewed above all belong to these two types, which aim to assess the performance of the HAV in a best-effort fashion. On the other hand, (formal) verification attempts to prove the satisfaction of the system to the properties for all considered conditions (ODD). Various verification techniques have been applied to rigorously certify the correctness of automotive software and functionalities [75], including reachability analysis [76], model checking [77] and automatic theory proofing [78]. However, verification techniques require a fully specified system model of the VUT (white box), which is a very strong assumption, and could be inaccessible for the HAV evaluation conducted by third-party organizations. Moreover, the correctness guarantee based on simplified models and assumptions could potentially fail under the more complex, highly-interactive and dynamic real driving environment. To fix this problem, Shalev-Shwartz et al. [79] proposed the idea of "responsibility sensitive safety" (RSS), which formalized a set of driving rules that can achieve safety assurance for a multi-agent driving environment and assign blame to the agent that violates. However, the public expectation of HAVs has always been to maintain safety under as many conditions as possible, rather than to avoid blame in all circumstances.

Apart from scenario-based approaches, some research evaluates the safety performance of HAVs in simulated traffic environments. [80] assessed the impact of different levels of HAVs by conducting multi-agent simulations with realistic road network of a Japanese city. [53, 81] applied an accelerated evaluation procedure to a VUT driving in a microscopic highway traffic simulator.

1.3 HAV Evaluation Problem Formulation

1.3.1 Development Testing v.s. Acceptance Testing

The evaluation efforts for HAVs can be classified into two categories based on their purposes: development testing and acceptance testing. The major differences between them are shown in Table 1.3. The key evaluation techniques reviewed above are summarized according to the categories in

	Development testing	**Acceptance testing**
Tester	Company self-assessment	Government, 3rd-party organization
Resource & time budget	High	Low
Test thoroughness	High	Low
Transparency of the HAV stack	High	Low
Fairness requirement	Low	High
Testing environment	Simulation, test track, public road	Test track only
Subject of testing	Software & hardware components, functionality modules, full HAV	Full HAV

Table 1.3: Comparison between development testing and acceptance testing.

Figure 1.1.

Development tests are usually conducted in-house by OEMs or HAV companies. The goal is to thoroughly test the system within its ODD, and try to find all the bugs and deficiencies for future improvements, while higher testing costs can be tolerated. Since the details of the hardware and software on the VUT are known, the test can be conducted at different levels, from software & hardware components, to the functional modules, to the entire HAV. Popular methods suitable for this type of evaluation include N-FOTs, formal verification, Monte Carlo sampling and falsification-based testing, etc, as shown in the right half of Figure 1.1. According to GRVA of the United Nations [82], HAV safety evaluation includes three main modalities: public-road testing, physical certification testing (on test tracks), and audit or assessment in simulations or with analytic tools. All of them are applicable to development tests.

On the other hand, acceptance tests are commonly conducted by a government agency or a trusted third party. The goal is to certify the basic competency of the VUT in a standardized environment in a fair and efficient way, considering the limitation on time and resources. Therefore, public road testing will not be feasible for its high cost. Moreover, since the details of the VUT will not be exposed, and the simulation interface of systems from different OEMs can hardly be unified, simulation testing is also not applicable. Therefore, the system under test will be the entire HAV, and tests will only be conducted in closed test tracks. Methods including test matrix, small-scale Monte Carlo sampling and falsification-based testing methods can all be applied, as shown in the lower part of Figure 1.1 (excluding the ones requiring a white-box VUT). This dissertation mainly focuses on acceptance testing, but the proposed methods are also applicable to development testing.

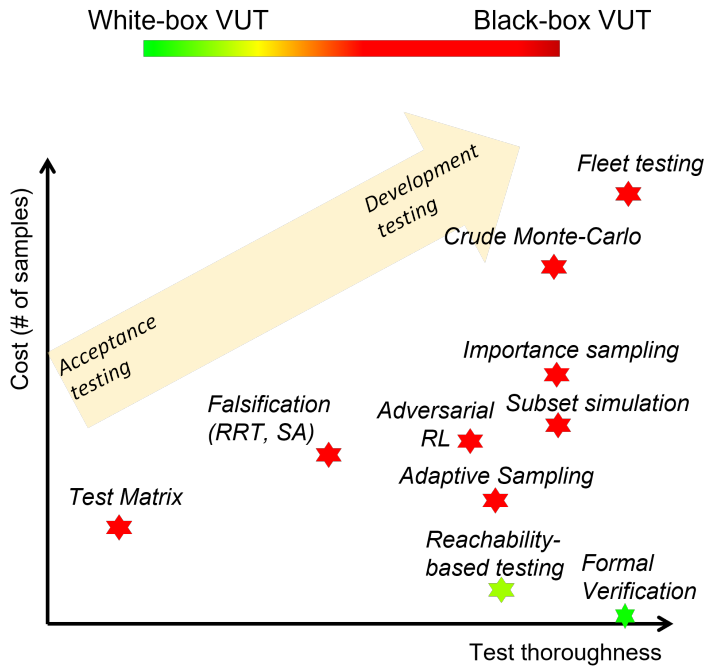


Figure 1.1: A summary of representative HAV evaluation techniques.

1.3.2 ABC Test

The original Mcity ABC test concept was proposed in 2019 as a practical framework for HAV acceptance testing [29]. It is a scenario-based testing framework designed primarily for tests in a closed track. The ABC test incorporates three major elements: **A**ccelerated evaluation, **B**ehavioral competence and **C**orner cases.

For the ABC test, the behavior competence test is the core and the foundation. The HAV is asked to operate in selected driving scenarios, each of which will have multiple stochastically-generated test cases sampled from its testing space. The test cases need to be fair, reasonable and associated with clear risk levels. The HAV is evaluated by how well it resolves conflicts and avoids crashes or near-misses across different cases and scenarios. Rooted from the same pipeline, the considerations of accelerated evaluation and corner cases can be achieved by modifying the testing space as well as the test case generation scheme. The accelerated evaluation aims at reducing the uninteresting test cases, and achieving an unbiased estimate of certain aggregated performance indices (e.g. crash rate, injury rate) efficiently, which can be achieved by designing smart sampling techniques. Corner case tests attempt to find the extreme failure modes of a VUT, which can be achieved by designing the testing space around the extremes of the ODD, and creating efficient falsification schemes to generate test cases. The three paradigms provide different views on the safety competence of an HAV, and they together create a comprehensive safety assessment. An analogous

effort was presented in [83], which classified test automation methods into naturalistic-assessment-oriented, coverage-oriented and unsafe-scenario-oriented categories. Though the "ABC" trio originated from acceptance testing, the concept directly applies to development testing as well.

Built upon the ABC test concept, this dissertation will work on evaluation methodologies for all three components of the ABC test. We make the following assumptions:

1. No prior knowledge is assumed about the VUT, i.e. the VUT is a black box to the tester.
2. The total number of test cases is limited for each test due to time and financial limitations.
3. The POV(s) have perfect kinematic information of the VUT (position, acceleration, speed, heading, etc.)
4. We ignore the variations in environmental factors including weather, lighting conditions, etc. Each scenario will be tested at the same spot on the test track.

Assumption (1) holds true for all acceptance tests, and for most development tests where the model of the vehicle (system) under test is too complex to be leveraged. Assumption (2) is based on the nature of the acceptance testing. For development testing, requiring fewer test cases is also desirable for cost saving. Assumption (3) can be realized because we can install a portable RTK GPS and a V2X communication module onto the VUT, which broadcasts high-accuracy vehicle states with low latency (details in Chapter 5). Assumption (4) is made because our evaluation methodologies primarily focus on testing the VUT's capability of handling the POVs. The accuracy and robustness of its sensing and perception capability are not addressed due to the budge-sensitive nature of acceptance testing. The test procedure can be repeated with selected environmental factor variations (e.g., at night, in snow) if time/weather permits.

1.3.3 The Decomposition of Scenario-based Evaluation Procedure

The implementation of scenario-based testing involves the following three tasks:

1. Scenario modeling: extract key attributes that characterize a given scenario, e.g., initial speed, distance margin, behavioral property of the POV, etc.; then form the testing space.
2. Test case generation: generate test samples from the testing space in a comprehensive and efficient way.
3. Test execution: realize the target test cases in a simulated or real-world testing environment by controlling the behaviors of the POVs; assess the performance of the VUT.

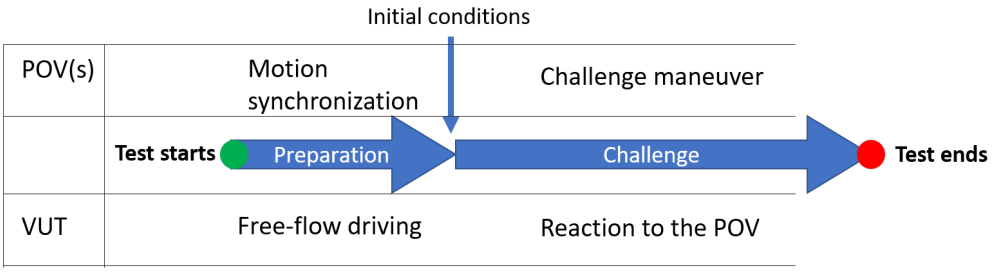


Figure 1.2: The execution pipeline for the ABC test.

The "ABC" trio will have different approaches for solving the first two tasks, as discussed in the above section, but they share the same execution procedure (3rd task) with the basic behavioral competence tests. All three tasks will be covered in this dissertation.

The overall execution procedure for each test case is illustrated in Figure 1.2. Each test run includes two phases, the preparation phase and the challenge phase. In the preparation phase, the VUT drives in a free-flow environment where its behavior is not impacted by the POV. The POV will conduct motion synchronization with the VUT to ensure that the predetermined initial condition of the test can be accurately triggered. The challenge phase starts when the initial conditions are achieved by the POV. Then, the POV executes the challenge maneuver while the VUT will need to react. The tester only has control over the motion of the POV, while VUT's motion can only be observed by the POV and the tester.

1.3.4 Scenario Categorization

From the list of 50 scenarios for the ABC test [29], we focus on the ones involving at least one other road user. The road user is called POV even though it could be a VRU. We classify these scenarios into the following categories based on the interaction between the POV and the VUT:

- Reactive scenarios: the VUT reacts to a non-compliant POV. Examples: cut-in, unprotected left turn, pedestrian crossing, etc.
- Semi-interactive scenarios: the VUT reacts to the POV, while the POV's behavior is not affected by the VUT. Example: car-following.
- (Bilateral) interactive scenarios: the POV(s) and the VUT will have mutual influence on the future decision-making and motion-planning of each other. Examples: highway merging, roundabout entering, etc.

In a reactive scenario, the traffic conflict is solely created by the POV. The VUT is challenged by the POV "in a surprise", while the VUT has the right of way. The VUT is nevertheless expected to

react safely and promptly to the encroachment of the POV. Because reactive scenarios have a short challenge phase, the behavior of the POV can be modeled as a predetermined maneuver. Evaluation for reactive scenarios has been widely studied for ADAS with level 1 and level 2 autonomy.

For an interactive scenario, the VUT needs to negotiate the passing order with the POV in the challenge phase. This interaction takes place over a time horizon, which cannot be ignored. In the settings presented in this dissertation, the right-of-way (ROW) always belongs to the POV (e.g. the VUT tries to merge from the ramp when the POV is on the main road). In other words, traffic conflicts are partly or fully caused by the VUT. Therefore, the VUT has to predict the intention of the POV and plan for a safe trajectory accordingly. For SAE level 3 and above automated vehicles [1], their ODD includes these interactive scenarios. Therefore, the evaluation procedure for such scenarios will have a high demand in the near future as the level of autonomy of the state-of-the-art systems progress. Similar taxonomy of traffic scenarios was also adopted by [84].

In this dissertation, we will focus on the scenario modeling and test case sampling method for both reactive scenarios and (bilateral) interactive scenarios, which represent two extremes with respect to the POV/VUT relationship. The semi-interactive scenarios are in the middle-ground between the above two kinds, thus methodologies we develop can be naturally extended to semi-interactive scenarios.

By combining two types of scenarios with three components of the ABC test, we have 6 types of evaluation problems, as shown in Table 1.4. The scope of this work covers 4 of the 6 types. The corner case testing for reactive scenarios has been extensively studied in the literature [37, 57, 59], thus it will not be covered here. The accelerated evaluation for interactive scenarios will be studied in future work.

	Reactive Scenarios	**Interactive Scenarios**
Accelerated evaluation	Chapter 2	Not covered
Behavior competence testing	Chapter 5	Chapter 3
Corner case testing	Not covered	Chapter 4

Table 1.4: Matrix of ABC test to be covered in the dissertation.

1.4 Contributions

In this dissertation, we propose a suite of safety evaluation methods for HAVs. Our main contributions are the following:

1. We propose an evaluation method for reactive scenarios that ensure the feasibility and interpretability of test cases, while achieving accelerated crash rate estimation.

2. We propose an evaluation framework for interactive scenarios. On one hand, the diversity and richness of driver interactions are captured by the proposed POV behavior library. On the other hand, we achieve comprehensive failure modes identification with an adaptive test case generation method.

3. We present a novel formulation of corner cases for interactive scenarios based on the notion of ambiguity. We successfully generate interaction-aware corner cases at pedestrian crossing and highway merging scenarios using the proposed adversarial planning method for the primary other road users (PORUs).

4. We implement the behavior competence testing procedure for multiple test scenarios both in the CARLA simulator and on the Mcity test track. Test cases can be executed accurately and repeatedly in an automated fashion.

1.5 Outline of the Dissertation

In the remainder of the dissertation, Chapter 2 presents the evaluation framework for reactive scenarios. We use the pedestrian crossing scenario to introduce the methodology of scenario modeling and test case generation, and later extend to two other scenarios: unprotected left turn and cut-in. Chapter 3 introduces the evaluation framework for interactive scenarios. We present the POV library generation scheme based on game theory and reinforcement learning, and present the adaptive test case generation scheme. Chapter 4 presents the interactive-aware corner case generation method. We describe the ambiguity-guided planning algorithm and its implementation in two interactive scenarios. Chapter 5 presents the method and results for implementing the evaluation framework in simulation and in real-world testing. Finally, Chapter 6 makes concluding remarks and lists future work.

CHAPTER 2

Safety Evaluation for Reactive Scenarios

2.1 Problem Formulation

This chapter will focus on the evaluation of reactive scenarios. For a reactive scenario, since the challenge maneuver from the POV lasts for a short period, it is assumed to be predetermined. Therefore, test cases are defined only by the initial conditions. We will generate test initial conditions following the accelerated evaluation paradigm. We first use the pedestrian crossing scenario as the main example to demonstrate the workflow. Then, the methodology will be extended to two other scenarios, unprotected left-turn (ULT) and cut-in.

2.2 Model of the Pedestrian Crossing Scenario

2.2.1 Literature Review

According to [85], 5,977 pedestrians were killed in motor vehicle crashes in 2017 in the US, corresponding to 16% of all traffic fatalities. The fatal crash involving an Uber HAV [5] also raised concerns about pedestrian safety protection of HAV systems.

Pedestrian crossing behaviors have been recorded and studied in the past. Some studies used an onboard camera for pedestrian data collection, including the Daimler pedestrian dataset [86], and the Caltech pedestrian detection benchmark [87]. In addition, more than 2,900 pedestrian-crossing events were extracted from the Safety Pilot Model Deployment naturalistic driving database in [88]. By using an onboard camera, vehicular kinematic information can be accurately recorded. However, as vehicles pitch under braking or with road undulation, the measurements of pedestrian motion can be inaccurate. Other data collection efforts used cameras/sensors fixed on the roadside. Human observation, fixed traffic cameras or speed guns were all used to record the motion of crossing pedestrians and approaching vehicles [89, 90]. These data capture the motion of pedestrians more accurately due to fixed sensor placement and fixed scenarios. However, the cost of data

collection can be higher.

2.2.2 Collection of Pedestrian Crossing Events

In this paper, we collect pedestrian-vehicle interaction event data from open-access Network IP cameras. There are thousands of open traffic cameras online and live-streaming, some of which are pointing at intersections or crosswalks. By utilizing the data online, the diversity and quantity of the collected data are higher. However, video processing can be more involved.

The workflow of event extraction is adopted from [91]. Video data are first recorded, then vehicles and pedestrians are detected and tracked; camera calibration is then conducted remotely; finally, the velocities of objects are estimated.

Data Collection

We utilize data from one camera placed on the main road of Banff, Alberta, Canada, which is live-streaming at [92]. As shown in Figure 2.1, a mid-block crosswalk is captured, along with all approaching vehicles from one direction. Moreover, as there is a barrier between the two sides of the road, pedestrian-crossing and vehicle-yielding behaviors on the left side of the road can be assumed to be decoupled from the right side. We will focus on pedestrian-vehicle interaction on the left side only. 40 hours of videos were collected at this site, with a frame rate of 15 FPS and resolution of 1120×840.

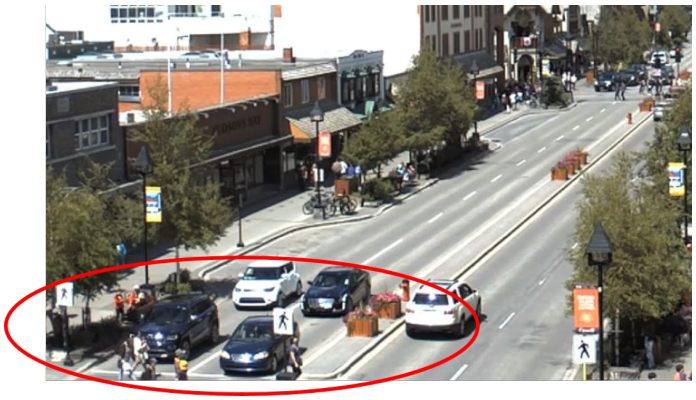


Figure 2.1: The data collection site in Banff, Alberta, Canada: the target crosswalk is shown in the red circle

Object Detection

For object detection, a popular version of You Only Look Once (YOLO v3) is used to detect and extract all traffic agents from the video data [93]. YOLO is a state-of-the-art object detection

system, featuring real-time performance and high accuracy. The output of YOLO v3 is bounding boxes around pedestrians and vehicles, which becomes the input of object tracking.

Camera Calibration

To relate the positions of objects on 2-D video frames to 3-D real-world locations, camera calibration is conducted. Camera calibration is the process of finding the transformation between the camera coordinate frame and the world coordinate frame by estimating the intrinsic and extrinsic parameters of the camera. A pinhole model is used to model the camera, and the intrinsic and extrinsic parameters can be lumped together into a 3×4 matrix P. For a point in the world frame with coordinate $[x_w, y_w, z_w]^T$, its coordinate in the camera frame $[u_c, v_c]^T$, can be computed as follows:

$$\begin{bmatrix} u_v \\ v_c \\ 1 \end{bmatrix} = P_{3\times4} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.1}$$

To estimate the matrix P, the direct linear transformation (DLT) method is used [94]. To run DLT, several control points (CPs) are required, for which both the camera coordinates and the world coordinates must be known. As the data are recorded by a third party and the spot is thousands of miles away, we do not have access to camera parameters and the geometry of the camera location. Therefore, we estimate the world coordinates of CPs for calibration based on multiple information sources, including geographic information from Google Maps, pavement marking standards, the configuration of certain vehicle models, and other heuristics. 14 CPs are located, which is more than the theoretical requirement of 6 CPs. As a result, the average re-projection error is 9.6 pixels in a 1120×840 frame.

Object Tracking

The object tracking algorithm is based on the "Motion-Based Multiple Object Tracking" example from MATLAB. The bounding boxes serve as the input to the object tracker. For each tracked target, a Kalman filter is maintained to predict the next location of the target. With every new frame, the new detection results will be associated with the current targets based on a cost matrix using the Hungarian algorithm [95]. Next, tracks are either updated, destroyed or enriched. To filter out unnecessary information for modeling, we constrain the region for detection and tracking only at the left side of the road, 100 meters within the crosswalk. In Figure 2.2, an example tracking result is shown. All the vehicles and pedestrians in the target area are detected and indexed with tracking ids.

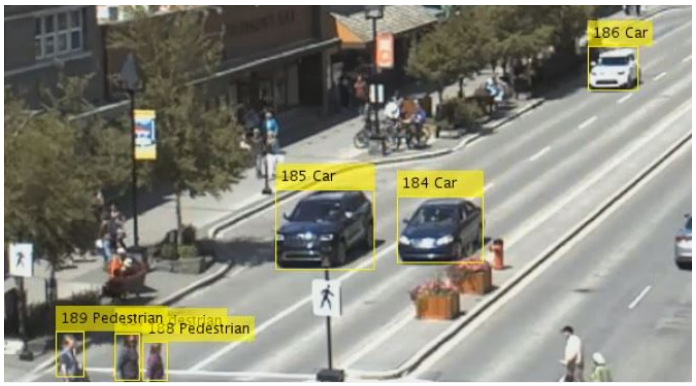


Figure 2.2: An example result of object tracking

Motion Estimation

Object tracking provides trajectories of each pedestrian and vehicle. Then, to characterize their interaction near a crosswalk, the speed profiles of pedestrians and vehicles need to be estimated. We use the following filtering framework for speed estimation:

1. The position data are filtered using a zero-phase digital filter with a Kaiser window.
2. To estimate the speed trajectories, a rolling-window linear regression is conducted to each position trace with a window length of 3.3s.
3. The speed trajectory is filtered again with a zero-phase digital filter with a Kaiser window and a cutoff frequency of 0.3Hz.

Example results are shown in Figure 2.3. In (a) and (b), both pedestrians walk down the crosswalk from right to left at a steady speed; in (c) and (d), both vehicles decelerate as they approach the crosswalk.

In the end, we collected 2689 events with one pedestrian crossing the road while a vehicle is approaching the intersection. This data collection and processing procedure can be extended to other traffic scenarios. Moreover, it is relatively easy to augment the data, as no field data collection or complex sensor placement is needed. On the other hand, this method also has drawbacks. First, calibration of an unknown camera involves applying some heuristics, which can be time-consuming and inaccurate. Second, object tracking based on bounding boxes does not account for a lot of visual features of the targets, including color, texture, etc. Thus, the performance of tracking can deteriorate when the scene becomes cluttered.

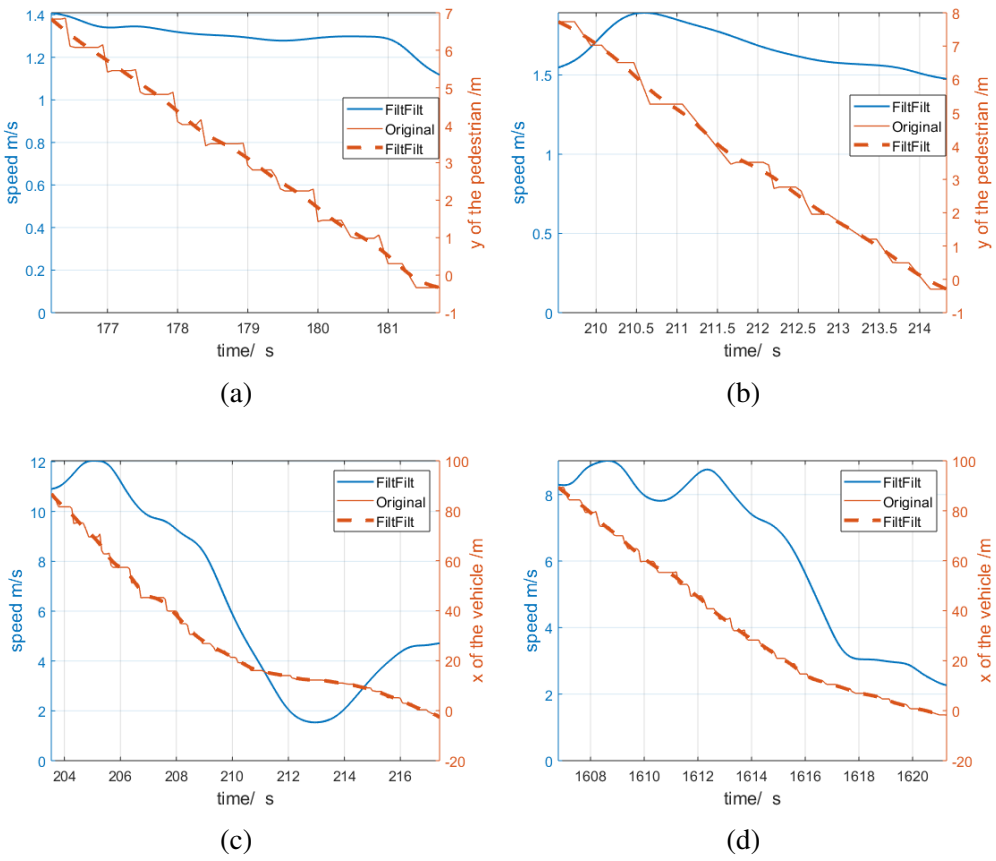


Figure 2.3: Example results of motion estimation: red solid lines show observed position; red dashed lines show filtered position; blue lines show estimated speed.

2.2.3 Statistical Model of the Scenario

Extraction of the Key Variables

In the pedestrian crossing scenario, the desired behavior of the HAV is to stop for the pedestrian once she/he starts to cross. We focus on the interaction between a single pedestrian and a single primary vehicle (PV). Denoting the moment when a pedestrian enters the crosswalk as t_x, we make the following assumptions about the pedestrian and the vehicle:

1. The pedestrian walks at a constant speed after t_x.
2. The pedestrian decides to cross or not according to the speed and position of the PV at t_x, and the walking speed of the pedestrian.
3. Each crossing pedestrian is viewed as a separate event.

Then, we model the pedestrian crossing scenario with the following 4 variables.

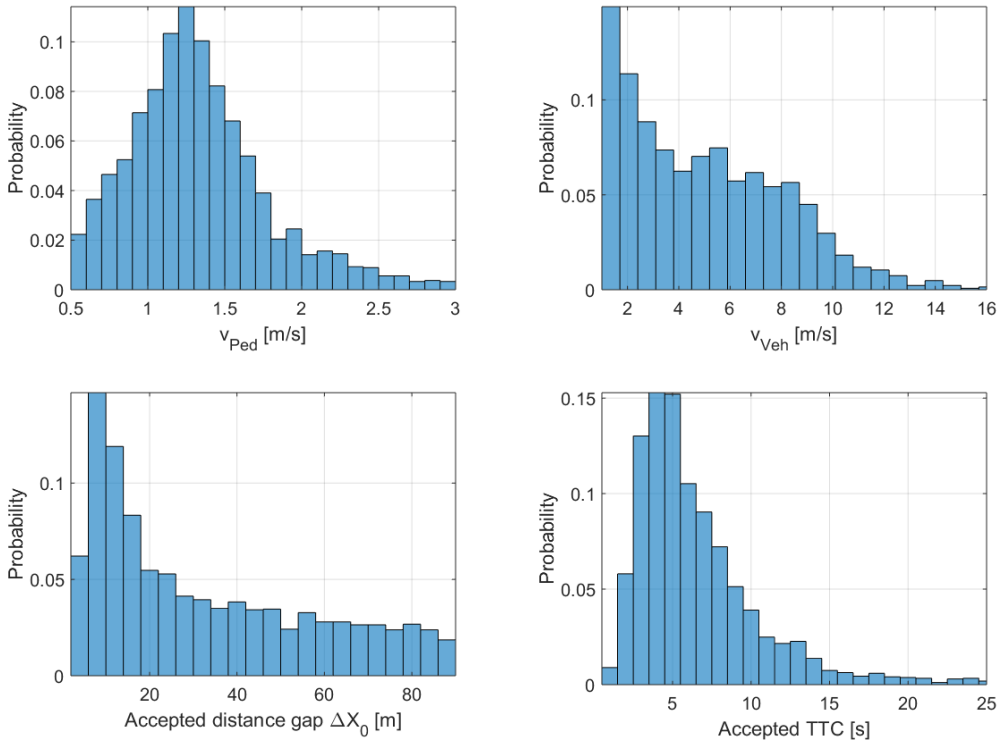


Figure 2.4: Histograms of the 4 key variables for the pedestrian crossing scenario.

1. Δx_0: longitudinal distance between the pedestrian and the PV at t_x.
2. v_{Ped}: walking speed of the pedestrian.
3. v_{Veh}: speed of the PV at t_x.
4. $TTC = \frac{\Delta x_0}{v_{Veh}}$: time-to-crosswalk estimated at t_x for the PV.

All 2689 pedestrian-vehicle events are used to build the pedestrian model. The marginal distributions of the aforementioned 4 variables are shown in Figure 2.4. As shown in the top-left plot, Most pedestrians walk between 1 and 1.5 m/s, and the average walking speed is 1.33 m/s, which agrees with previous research [90, 96]. Most vehicles travel under 11 m/s (top-right plot), which is slightly above 8.33 m/s (25 km/h), the posted speed limit of this road.

Among TTC, Δx_0 and v_{Veh}, knowing any of the two can determine the third. Therefore, we express the pedestrian crossing model as a joint distribution of the 3 variables, which form the testing space of this scenario X_{TS}: $x_{TS} = [\Delta x_0^{-1}, v_{Ped}, v_{Veh}]^T$, where $x_{TS} \in X_{TS}$:

$$X_{TS} = [(\Delta x_0^{-1})_{lb}, (\Delta x_0^{-1})_{ub}] \times [(v_{Ped})_{lb}, (v_{Ped})_{ub}] \times [(v_{Veh})_{lb}, (v_{Veh})_{ub}] \tag{2.2}$$

Here we use Δx_0^{-1} instead of Δx_0 to put the higher risk cases (smaller Δx_0) to the 'tail' of the distribution, and give them higher resolution compared to the lower risk cases.

The Truncated Gaussian Mixture Model

As the data have upper and lower bounds due to the physics of the scenario, the joint distribution of the above three variables can be fitted using a truncated Gaussian mixture model (TGMM) bounded with X_{TS}. We chose this model due to its flexibility and advantages of expressing bounded data. The probability density function (PDF) of a TGMM can be written as:

$$f_p(p) = \sum_{i=1}^{K} \pi_i f_i(p) \tag{2.3}$$

where $\sum_{i=1}^{K} \pi_i = 1; \pi_i \geq 0, \forall i$. K is the number of mixing components, and each component is a truncated multivariate Gaussian distribution, parameterized by mean μ_i and covariance matrix Σ_i [97].

The parameters to be fitted are:

$$\Theta = \{\pi_1, ...\pi_K, \mu_1, ...\mu_K, \Sigma_1, ...\Sigma_K\}$$

The lower bound and upper bound of each variable are shown in Eq. (2.2). The optimal TGMM parameters are estimated using maximum likelihood estimation (MLE) by applying the expectation-maximization (EM) algorithm adopted from [97]. The hyper-parameter K is chosen based on the Bayesian information criterion (BIC), which balances model complexity and the likelihood of fitting results. $K = 4$ is selected for our model.

In Figure 2.5, the empirical data (1st row) and the fitted TGMM (2nd row) are shown. As the joint distribution is 3-dimension, we show the three 2-dimension marginal distributions when projected onto each other dimension.

2.3 Accelerated Evaluation Framework

To generate test cases for reactive scenarios in an accelerated and reasonable way, we propose the idea of combining reachability analysis with importance sampling to improve the original accelerated evaluation method [32]. For a target scenario, reachability analysis generates the feasible initial conditions that will be sampled from, as well as different risk level sets (RLSs), in which the risk is characterized by the intensity of actions required to avoid collisions. Then, importance sampling is conducted on each RLS to generate test cases in an efficient and flexible way.

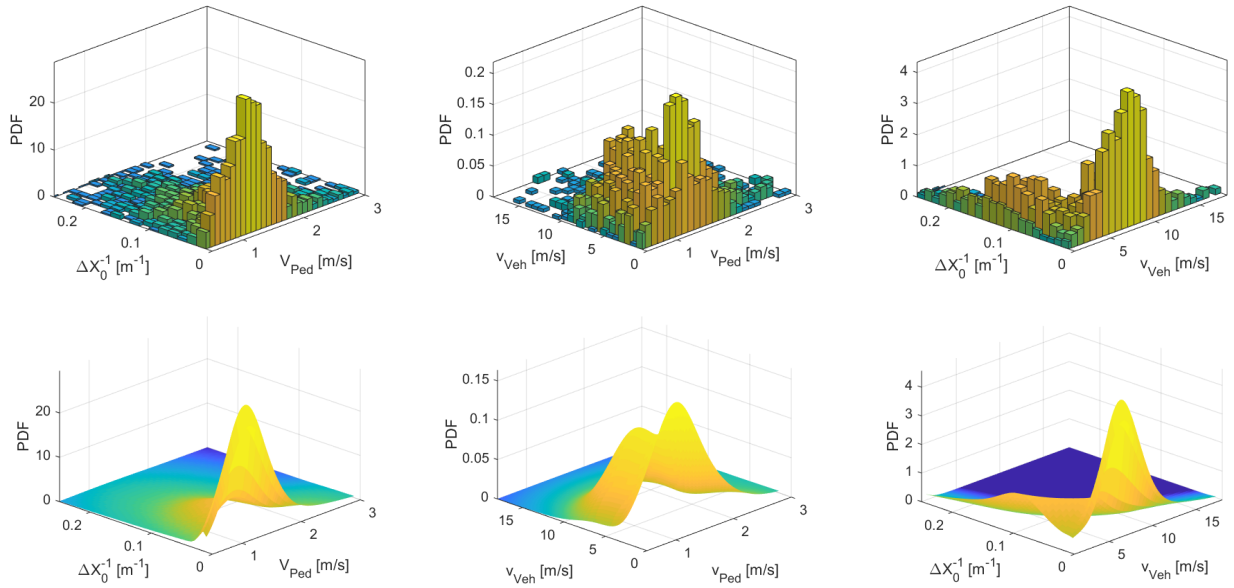


Figure 2.5: Empirical distribution and the TGMM model of the pedestrian crossing scenario.

2.3.1 Mathematical Tools

Importance Sampling

The crude Monte-Carlo (CMC) sampling method is a basic method for expectation estimation, and in our case, can be used to simulate real driving conditions and estimate the crash rate. Let $p \in D$ be an n-dimensional random variable following the distribution $f(p)$, where $D \subseteq \mathbb{R}^n$. Let ε be the event of interest, i.e. a collision, which is a subset of the sample space D. Whether the event ε happens can be expressed by:

$$I_\varepsilon(p) = \begin{cases} 1 & p \in \varepsilon \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

By running CMC simulations for N times, we can achieve an estimate of the crash/conflict rate by:

$$\gamma = \mathbb{E}_{p\sim f}(I_\varepsilon(p)) = \int_D I_\varepsilon(p) f(p) dp \approx \frac{1}{N} \sum_{i=1}^{N} I_\varepsilon(p_i) = \hat{\gamma}_N^{CMC},\ p_i \sim f(p) \tag{2.5}$$

Here, the estimate $\hat{\gamma}_N^{CMC}$ is a random variable with mean $\mathbb{E}(\hat{\gamma}_N^{CMC}) = \gamma$ and variance $\sigma^2(\hat{\gamma}_N^{CMC}) = \frac{\gamma(1-\gamma)}{N}$. Therefore, $\hat{\gamma}_N^{CMC}$ is an unbiased estimator. Its accuracy can be measured with the coeffi-

cient of variation (CV) [56], which is written as

$$CV^{CMC} = \frac{\sigma(\hat{\gamma}_N^{CMC})}{\mathbb{E}(\hat{\gamma}_N^{CMC})} = \sqrt{\frac{1-\gamma}{\gamma N}} \tag{2.6}$$

With an increasing number of samples, the CV will decrease to zero, which translates to an estimate that converges to the actual expectation. However, as the conflict/crash cases are rare in naturalistic driving (γ close to zero), it could take a huge number of simulations (N) for the variance to be small enough, and for the estimate to converge.

Therefore, the importance sampling technique has been widely used in the literature to reduce the variance and accelerate the convergence of the estimate [32, 98, 99]. Assume there is another distribution $f^*(p)$. Eq. (2.4) can be rewritten as:

$$\begin{aligned} \gamma &= \int_D I_\varepsilon(p) f(p) dp = \int_D I_\varepsilon(p) \frac{f(p)}{f^*(p)} f^*(p) dp \\ &= \mathbb{E}_{p\sim f^*}(I_\varepsilon(p)L(p)) \approx \frac{1}{N}\sum_{i=1}^{N} I_\varepsilon(p_i)L(p_i) = \hat{\gamma}_N^{IS},\ p_i \sim f^*(p) \end{aligned} \tag{2.7}$$

where the likelihood ratio is defined as

$$L(p) = \frac{f(p)}{f^*(p)} \tag{2.8}$$

We can draw samples according to distribution $f^*(p)$, and also acquire an empirical estimation of γ. We have the following:

$$\begin{aligned} &\mathbb{E}(\hat{\gamma}_N^{IS}) = \gamma, \\ &\sigma^2(\hat{\gamma}_N^{IS}) = \frac{1}{N}\int_D \frac{(I_\varepsilon(p)f(p) - \gamma f^*(p))^2}{f^*(p)} dp. \end{aligned} \tag{2.9}$$

Therefore, $\hat{\gamma}_N^{IS}$ is also an unbiased estimation. Moreover, by carefully choosing the importance sampling distribution (ISD) $f^*(p)$, such that it has a higher density at where $I_\varepsilon(p)f(p)$ is large, the odds to generate collisions can be vastly augmented, and the estimation will have a smaller variance and thus smaller CV. Then, it takes fewer samples than CMC to reach convergence with the same confidence level.

Backward Reachability Analysis

Reachability analysis has been applied to safety verification and synthesis for HAV systems [100, 101]. In this research, it is used to solve two problems:

1. To improve the safety and efficiency of the evaluation procedure, we want to exclude impossible cases from the tests, i.e., when a crash is unavoidable regardless of HAV's action.

2. To improve the interpretability of test cases, we want to characterize the sets of different risk levels to create a structured testing space.

Here we follow the formulation in [102]. Consider a discrete-time system with both control input and external disturbance input:

$$x(k+1) = f(x(k), u(k), w(k)). \tag{2.10}$$

Then state x, control input u, disturbance w satisfy:

$$x(k) \in X \subseteq \mathbb{R}^{n_x}, u(k) \in U \subseteq \mathbb{R}^{n_u}, w(k) \in W \subseteq \mathbb{R}^{n_w}, \forall k \in \mathbb{Z}_{\geq 0}.$$

For a set $S \subseteq X$, which usually represents some bad region that the control input tries to avoid and the adversarial disturbance tries to enter, the one-step robust backward reachable set (also known as predecessor set) of S is defined as:

$$\textbf{Pre}(S) = \{x \in X \, : \, \exists w \in W \text{ s.t. } f(x, u, w) \in S, \, \forall u \in U\} \tag{2.11}$$

Thus, $\textbf{Pre}(S)$ is the set such that if the system starts inside of it, no matter what control input is taken, there always exists some disturbance that drives the system into S in one time-step. Note that this is very similar to the unrecoverable set defined in [103]. Compared to the robust predecessor set defined in [102], we swap the disturbance and control input for our application.

Moreover, we can define the backward reachable tube (BRT) of S as the union of all the backward reachable sets.

$$\textbf{BRT}(S) = S \cup \textbf{Pre}(S) \cup \textbf{Pre}(\textbf{Pre}(S))... \tag{2.12}$$

which represent all initial conditions that will always end up inside the set S sometime in the future.

To make the computation tractable, we simplify the formulation in the following ways. First, if the system is linear time-invariant, the dynamics can be written as follows:

$$x(k+1) = Ax(k) + Bu(k) + Fw(k) \tag{2.13}$$

Then, if we assume U, W, S are all polytopes, we can compute the **Pre** sets with basic polytopic

operations [102]:

$$\begin{aligned}\textbf{Pre}(S) &= \{x \in X \,:\, \exists w \in W \text{ s.t. } Ax + Bu + Fw \in S,\, \forall u \in U\} \\ &= \{x \in X : \exists y \ \in S, \exists w \in W \text{ s.t. } Ax = y + (-Fw) - (Bu), \forall u \in U\} \\ &= \{x \in X : Ax \in (S \ominus BU) \oplus (-FW)\}\end{aligned} \tag{2.14}$$

Where $\oplus$ is the Minkowski sum operation, $\ominus$ is the Pontryagin difference operation. For two polytopes P and Q in $\mathbb{R}^n$, they are defined as follows:

$$P \oplus Q = \{p + q \in \mathbb{R}^n : p \in P,\, q \in Q\}. \tag{2.15}$$

$$P \ominus Q = \{x \in \mathbb{R}^n : x + q \in P,\, \forall q \in Q\} \tag{2.16}$$

2.3.2 Kinematic Model of the Scenario

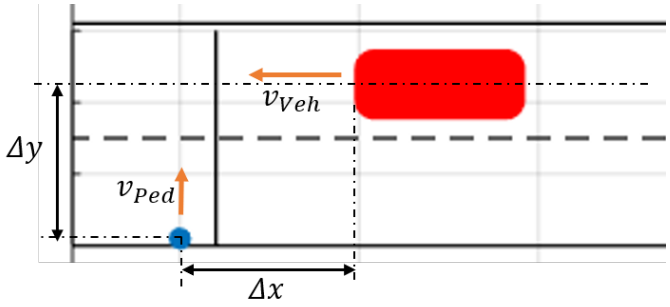


Figure 2.6: Kinematic model of the pedestrian scenario.

We model the configuration and dimension of the pedestrian crossing scenario according to the data collection site above. As shown in Figure 2.6, The kinematic model of the scenario consists of 4 states:

$$x = [\Delta y, \Delta x, v_{Ped}, v_{Veh}]^T$$

The vehicle moves to the left, while the pedestrian walks to the top. The initial moment is t_x, i.e. when the pedestrian starts to cross. The system can be modeled with a discrete linear system with time-step dt. The matrices in Eq. (2.13) are expressed below.

$$A = \begin{bmatrix} 1 & 0 & dt & 0 \\ 0 & 1 & 0 & -dt \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ -0.5dt^2 \\ 0 \\ dt \end{bmatrix}, \quad F = \begin{bmatrix} 0.5dt^2 \\ 0 \\ dt \\ 0 \end{bmatrix}$$

Parameter	Value	Parameter	Value
$(\Delta x_0)_{lb}$	2.0 [m]	$(\Delta x_0)_{ub}$	100.0 [m]
$(v_{Ped})_{lb}$	0.5 [m/s]	$(v_{Ped})_{ub}$	3.0 [m/s]
$(v_{Veh})_{lb}$	1.0 [m/s]	$(v_{Veh})_{ub}$	18.5 [m/s]
T	5.0 [s]	dt	0.1 [s]
a_{min}	-6.4 [m/s^2]	a_{max}	3.0 [m/s^2]
L_0	5.0 [m]	W_0	2.0 [m]
R_{Ped}	0.3 [m]	Δy_0	-4.5 [m]
t_{react}	0.5 [s]		

Table 2.1: Values of parameters for modeling and simulation at the pedestrian crossing scenario.

The control input u is the acceleration of the vehicle, and disturbance w is pedestrian acceleration (assumed to be 0 for now).

$$u \in U = [a_{min}, a_{max}], \quad w = 0.$$

The parameter values of the model and the evaluation framework are shown in Table 2.1.

2.3.3 Overview of the Evaluation Framework

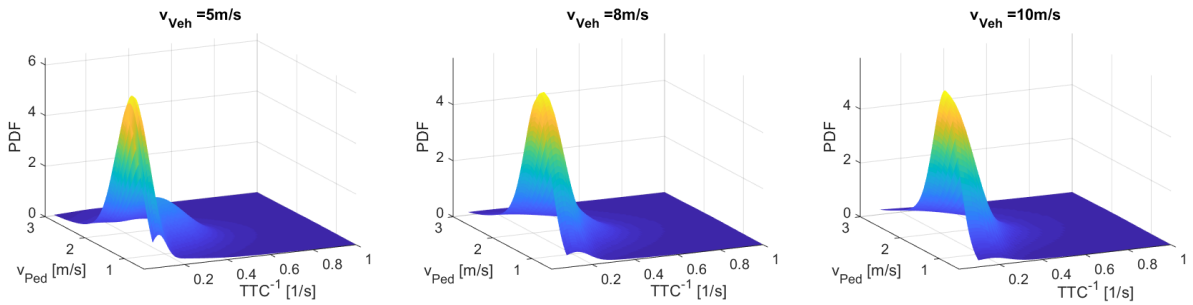
The proposed evaluation framework consists of the following steps:

1. Generate the naturalistic distribution (ND) f.
2. Find the set of feasible initial conditions for testing X_f.
3. Generate different risk level sets (RLSs) X_f^1, X_f^2, X_f^3...
4. Conduct importance sampling in each RLS to formulate the ISD f^*.
5. Simulation/real-world testing.
6. Test results interpretation.

Each step will be explained in the following sections.

2.3.4 Generating Naturalistic Distribution

We focus on step 1 in this section. As stated earlier, the pedestrian crossing scenario is modeled as a 3-dimension TGMM over $x_{TS} = [v_{Veh}, v_{Ped}, \Delta x_0^{-1}]^T = [s_1^T, s_2^T]^T$, where $s_1 = v_{Veh}$ and $s_2 = [v_{Ped}, \Delta x_0^{-1}]^T$. During a test, the VUT will regulate speed based on its driving policy when approaching the crosswalk. Thus, v_{Veh} is not sampled from the distribution, but observed from the



(a) (b) (c)

Figure 2.7: 2D distributions conditioned on different v_{Veh}.

VUT's behavior. The naturalistic distribution of the scenario is the conditional distribution of v_{Ped} and Δx_0^{-1} given v_{Veh}.

$$f_{\Delta x_0^{-1}, v_{Ped}}(\Delta x_0^{-1}, v_{Ped}|v_{Veh}) = f_{s_2}(s_2|s_1) \tag{2.17}$$

$f_{s_2}(s_2|s_1)$ can also be modeled as a TGMM. For each Gaussian component, the mixing coefficient is the same.

Moreover, as $TTC = \frac{\Delta x_0}{v_{Veh}}$, and TTC is a more popular metric for measuring the risk of the scenario [104] than Δx_0, we make a change of variable to the PDF of the ND:

$$f(TTC^{-1}, v_{Ped}|v_{Veh}) = \frac{f_{\Delta x_0^{-1}, v_{Ped}}(\Delta x_0^{-1}, v_{Ped}|v_{Veh})}{v_{Veh}}. \tag{2.18}$$

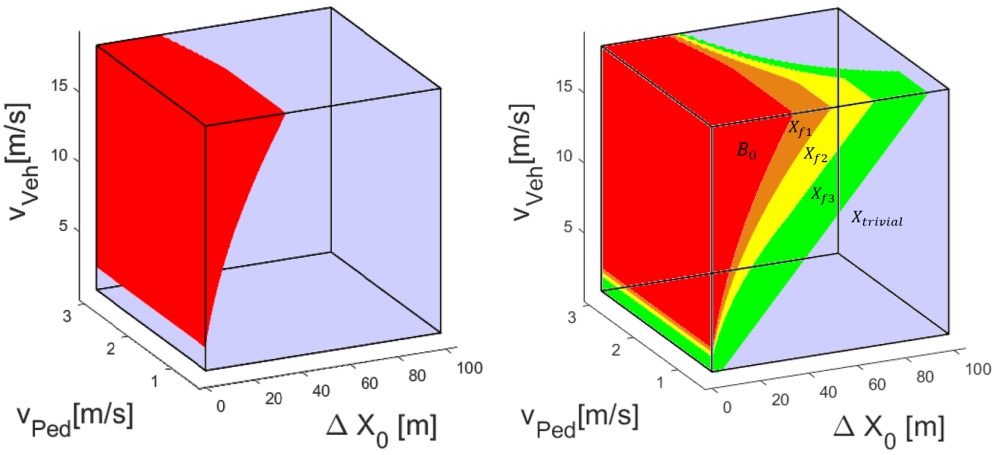
In the following step, we sample TTC^{-1} and v_{Ped} from the distribution f. The distributions conditioned on different v_{Veh} are shown in Figure 2.7.

2.3.5 Computing Risk Level Sets

We elaborate on step 2 and step 3 in this section. The reachability analysis is done using the Model Parametric Toolbox 3 (MPT3) [105]. First, we define the final collision set as a 4-dimension polytope B^0:

$$B^0 = \left\{ x : \quad |\Delta y| < \frac{W_0}{2} + R_{Ped} \wedge -(R_{Ped} + L_0) < \Delta x < R_{Ped} \right\}. \tag{2.19}$$

Here, the pedestrian is approximated as an axis-aligned square of width $2R_{Ped}$, and the vehicle is modeled as a rectangle with length L_0 and width W_0. A crash happens when these two regions overlap. Starting from B^0, we can back-propagate the dynamics and calculate the BRT for T seconds using Eq. (2.14) and Eq. (2.12), denoted by B_0'. In the BRT calculation, $W = \{0\}$ as



(a) Feasible initial condition set X_f (b) All risk level sets

Figure 2.8: RLSs for the pedestrian crossing scenario.

we assume the pedestrian always walks at a constant speed. For U, there are two phases with different treatments. First, we set $U = \{0\}$ during an initial reaction time t_{react}, i.e. the VUT keeps a constant speed, which accounts for the perception, communication and actuation delays. The onset of t_{react} is assumed to be t_x. Second, after t_{react}, $U = [a_{min}, a_{max}]$ as the VUT could utilize its full longitudinal acceleration capability to avoid a collision. B_0' is represented as a union of multiple polytopes. Then, slice B_0' at the position where the pedestrian starts, and we get:

$$B_0 = B_0'.\textbf{slice}(\Delta y = \Delta y_0).$$

B_0 is the set of initial conditions, from which VUT will always end up in a collision with the pedestrian regardless of VUT's action. B_0 is represented as a union of multiple 3-dimension polytopes. For the HAV evaluation, it is not valuable to sample initial conditions from B_0. Thus, instead of directly drawing samples from the testing space X_{TS}, we will only sample from the feasible set X_f, defined as:

$$X_f = \hat{X}_{TS} \setminus B_0 \tag{2.20}$$

Here, $\hat{X}_{TS}$ refers to the region defined by X_{TS} in Eq. (2.2), except with a change of variable from Δx_0^{-1} to Δx_0. X_f is demonstrated as the purple part in Figure 2.8a.

Moreover, if we shrink the set $U = [a_{min}, a_{max}]$, i.e. the acceleration/deceleration capability of the vehicle is reduced, the BRT from B^0 will become larger. With this observation, we generate a series of BRTs in the same way as B_0 but with decreased input set U, and result in the sets $B_1, B_2 ... B_n$. They satisfy:

$$B_0 \subseteq B_1 \subseteq B_2 ... \subseteq B_n. \tag{2.21}$$

Then, we can define the i^{th} risk level set (RLS) as:

$$X_f^i = B_i \setminus B_{i-1}, i = 1, 2, 3... \tag{2.22}$$

For any $x_{init} \in X_f^i$, it is possible for the VUT to avoid a collision with the pedestrian starting from x_{init} by definition, but it has to "take a certain level of effort" to do so. For our application, we divide X_f into four RLSs: the high risk set X_f^1, medium risk set X_f^2, low risk set X_f^3 and trivial set $X_{trivial} = \hat{X}_{TS} \setminus B_3$. The RLSs are shown in Figure 2.8b. The boundary between neighboring sets is determined by the minimum average deceleration required to avoid a collision after coasting for t_{react}. The thresholds are shown in Table 2.2, which are determined according to [106], where the deceleration profile of human drivers from the naturalistic driving database ICCFOT is analyzed. The hardest observed deceleration is -0.65g; 0.1% of all the decelerating cases exceed -0.41g and 1.0% of all the decelerating cases exceed -0.23g. Finally, the line between trivial and non-trivial cases is 0 m/s^2, which means if the VUT does nothing but coast, there still will not be a collision. Trivial cases are also not informative for safety evaluation. Therefore, in the next step, we will only sample from set X_f^1, X_f^2 and X_f^3.

Risk level set	**Required acceleration**
Infeasible Set: B_0	$[-\infty, -0.65g)$
High Risk Set: X_f^1	$[-0.65g, -0.41g)$
Medium Risk Set: X_f^2	$[-0.41g, -0.23g)$
Low Risk Set: X_f^3	$[-0.23g, 0)$
Trivial Set: X_f^4	$[0, +\infty]$

Table 2.2: Risk level sets definition.

To understand the relationship between risk levels and real-world data, we checked all the collected pedestrian events to see which RLSs they lie in. The results are shown in Table 2.3. All events belong to the feasible set, and over $2/3$ of the events are in the trivial set.

Risk level set	**Number of events**
Infeasible Set: B_0	0
High Risk Set: X_f^1	18
Medium Risk Set: X_f^2	25
Low Risk Set: X_f^3	785
Trivial Set: X_f^4	1861

Table 2.3: Number of events in different RLSs from the collected data.

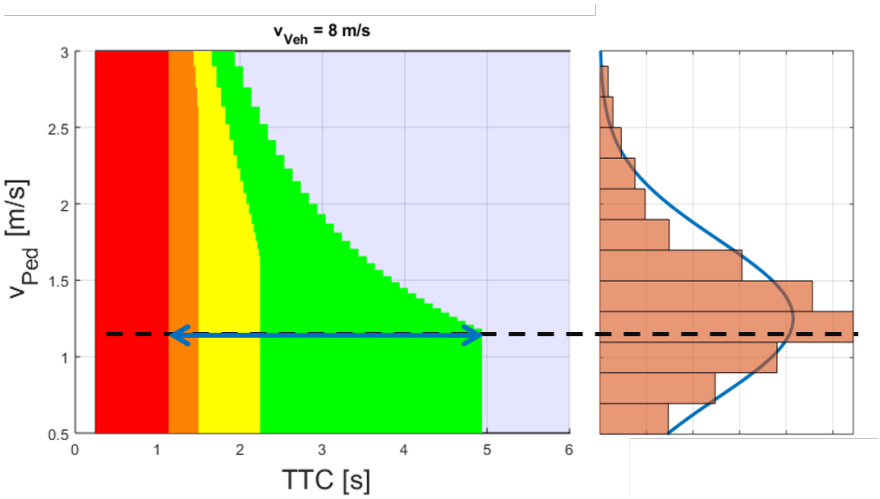


Figure 2.9: Importance sampling scheme.

2.3.6 Test Cases Generation with Importance Sampling

We focus on step 4 in this section. Given v_{Veh}, the RLSs are sliced into 2-dimensional regions, as shown in Figure 2.9 on the left. As there is no obvious relationship between the risk of a case and the pedestrian walking speed, we sample v_{Ped} from the naturalistic marginal distribution $f_{v_{Ped}}(v_{Ped})$, so that the test samples recover the way pedestrians walk in the real world. As the marginal distribution of a TGMM is not a TGMM anymore [107], we fit the data with one truncated Gaussian distribution, as shown in Figure 2.9 on the right.

Then, conditioned on the v_{Ped} and v_{Veh}, the feasible region for TTC to be sampled from becomes a 1-dimension interval, as shown with the blue arrow in Figure 2.9. The interval is again segmented by the three RLSs into subintervals:

$$
\begin{aligned}
I_{high} &= [TTC^{lb}_{high}, TTC^{ub}_{high}) \\
I_{mid} &= [TTC^{lb}_{mid}, TTC^{ub}_{mid}) \\
I_{low} &= [TTC^{lb}_{low}, TTC^{ub}_{low}].
\end{aligned}
$$

Then, we sample TTC from a 1-dimensional importance sampling distribution (ISD) in each interval.

Here, we propose a VUT-independent way to generate the ISD. If a total of N test cases are to be generated, we denote the ratio of test cases within high, medium and low-risk sets as: $R_{IS} = [r_{high} : r_{mid} : r_{low}]$, where $r_{high}+r_{mid}+r_{low} = 1$. R_{IS} determines the probability of sampling from each risk level, which is left to the testers to decide. It serves as a tuning knob for designing tests to the desired difficulty. Then, we propose to sample TTC^{-1} from a piece-wise uniform proposal

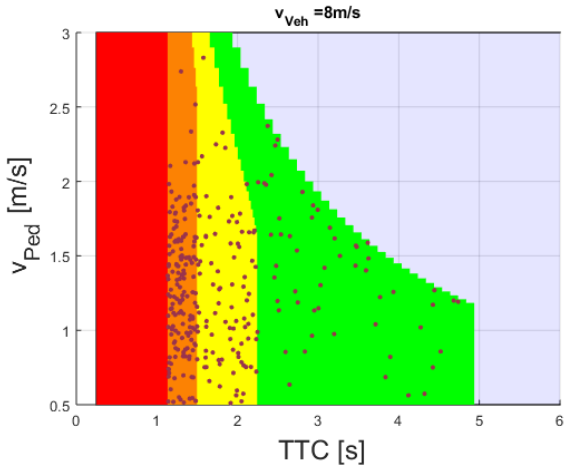


Figure 2.10: 300 test cases drawn from the ISD.

distribution.

$$f^*_{TTC^{-1}}(TTC^{-1}|v_{Ped}, v_{Veh}) = \begin{cases} \frac{k_n r_{high}}{\frac{1}{TTC^{lb}_{high}} - \frac{1}{TTC^{ub}_{high}}} & TTC \in I_{high} \\ \frac{k_n r_{mid}}{\frac{1}{TTC^{lb}_{mid}} - \frac{1}{TTC^{ub}_{mid}}} & TTC \in I_{mid} \\ \frac{k_n r_{low}}{\frac{1}{TTC^{lb}_{low}} - \frac{1}{TTC^{ub}_{low}}} & TTC \in I_{low} \\ 0 & \text{otherwise} \end{cases} \tag{2.23}$$

k_n is the normalizing factor that renders the integral of the PDF to be 1. Given the RLS to sampled from, the TTC^{-1} is sampled uniformly in the corresponding region. Finally, the formula for the proposed ISD is:

$$f^*(TTC^{-1}, v_{Ped}|v_{Veh}) = f_{v_{Ped}}(v_{Ped}) \times f^*_{TTC^{-1}}(TTC^{-1}|v_{Ped}, v_{Veh}) \tag{2.24}$$

After having both the ISD f^* from Eq. (2.24) and the ND f from Eq. (2.18), we can calculate the likelihood ratio for each test case according to Eq. (2.8), where $f^*(p) \neq 0$ if the sample is in X^1_f, X^2_f or X^3_f, according to the definition of f^*.

In Figure 2.10, we show an example of 300 samples generated with the aforementioned procedure. The ratio is $R_{IS} = [1/2 : 1/3 : 1/6]$. Most samples have v_{Ped} between 1 m/s and 2 m/s, as suggested by the distribution of v_{Ped}. All of the samples lie in the desired RLSs. In addition, the test cases are generated prior to the testing, and are independent of the VUT, which ensures fairness of the testing procedure for different VUTs from different companies.

Finally, we can run tests in simulation and estimate the crash rate, which will be covered in the

next section.

2.4 Simulation Results and Discussions

The usefulness of the proposed accelerated evaluation framework is demonstrated in simulation. The safety performance of the VUT is evaluated with the estimated crash rate $\hat{\gamma}_{crash}$ in the test. The controller of the VUT is assumed to be a combination of an Adaptive Cruise Control (ACC) algorithm and an Autonomous Emergency Braking (AEB) algorithm. The ACC is a PI controller targeting a desired time headway, while the AEB is from [108], which is similar to the system on a 2011 Volvo V60. Only the control algorithm is being evaluated without any perception module. Without loss of generality, the pedestrian is assumed to cross from left to right. A collision is registered if the VUT encroaches into the crosswalk whenever the pedestrian is still on the crosswalk, i.e., whether the pedestrian is in the left or right lane does not matter.

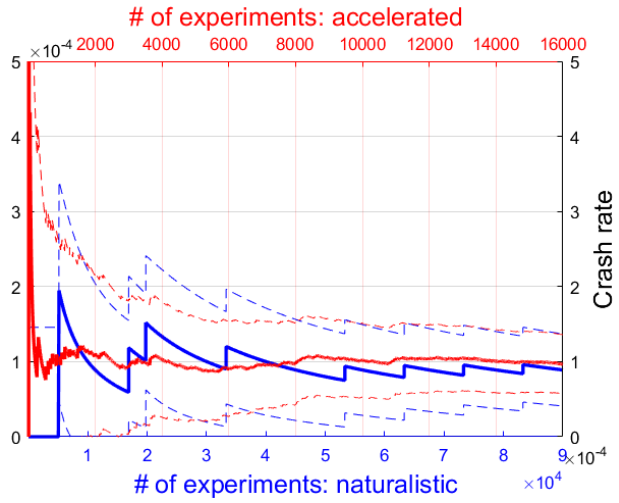


Figure 2.11: Simulation results: CMC v.s. proposed accelerated evaluation method.

In the simulations, we assume v_{Veh} is 8 m/s, i,e., around the posted speed limit of the road where data were collected. We compare the testing results using CMC and the proposed evaluation method. For the former, test cases are sampled according to the ND, which is truncated and constrained inside X_f, the region of feasible initial conditions. The probability density has been normalized according to the truncated region numerically. For the proposed method, we sample using the above ISD, where three RLSs are sampled with ratio $R_{IS} = [1/2 : 1/3 : 1/6]$. We run the simulation until the crash rate converges for both methods. The results are shown in Figure 2.11. The two simulation schemes both converge to the same crash rate estimates around 1.0×10^{-4}. The proposed method, shown in red, converges to the value earlier (after 7,000 samples), while

CMC takes around 70,000 samples. For both methods, the two dashed lines around the solid line show the error bound $\hat{\gamma}_{crash} \pm 1.64\hat{\sigma}_{crash}$, which is the region that with 90% of confidence level the real crash rate γ_{crash} lies in. The proposed accelerated evaluation method requires 1/10 of cases to reach the same confidence value as CMC sampling from the ND. In addition, it is observed that all test cases leading to a crash are from the high risk set, which demonstrated the significance of the risk level characterization.

Even before the crash rate reaches convergence, we are able to see how the VUT behaved. As all test cases are guaranteed to be feasible (within the assumed vehicle capability), any crash can be blamed on the ineptitude of VUT. Moreover, whether the VUT failed in a high-risk case or a medium-risk case can be used to rate the performance of the VUT. These kinds of knowledge were not available in the original accelerated evaluation framework in [32].

2.5 Extension to Other Reactive Scenarios

The complete evaluation framework is presented above. However, it is still expensive to execute in practice. It requires thousands of test cases to reach the unbiased crash rate estimate, while in real-world acceptance testing, we will only have days/weeks to test each VUT. Therefore, the accelerated evaluation could only be implemented in a "best-effort" fashion, where the number of test cases is constrained, and the testing will be terminated before the convergence is reached. In this section, we will extend the simplified version of the accelerated evaluation framework to two other scenarios, unprotected left turn (ULT) and cut-in.

2.5.1 Scenario Model

Unprotected Left Turn (ULT) Scenario

We focus on a typical ULT scenario, left turn across path/ opposite direction (LTAP/OD), which is the 3rd most fatal pre-crash scenario involving multiple vehicles [25]. The configuration is shown in Figure 2.12. The POV makes an unprotected left turn in front of the VUT, while the VUT is traveling straight through the intersection. The POV intentionally steals the ROW from the VUT, and the VUT needs to react appropriately. The following assumptions are made for the ULT scenario:

- The POV goes straight before the left turn. Then, the left-turn trajectory is a 90-degree arc with a constant radius.
- The speed of the POV is fixed during the turning phase. Under such speed, the lateral acceleration of the POV is within the friction limit of the tires.

- The VUT travels straight in the scenario.
- The initial condition is defined at t_{LT}, the moment when the POV initiates the left turn, i.e. when it is at the start of the arc.
- The VUT can recognize the turning intention of POV no earlier than t_{LT}.

This research only focuses on one intersection inside the Mcity test track. However, the method is generalizable to different intersection geometries and configurations. From our previous work [109], we model the initial condition of a ULT test case using 3 variables at t_{LT}:

1. Δx: the longitudinal distance between POV and VUT.
2. v_{POV}: the speed of POV.
3. v_{VUT}: the speed of VUT.

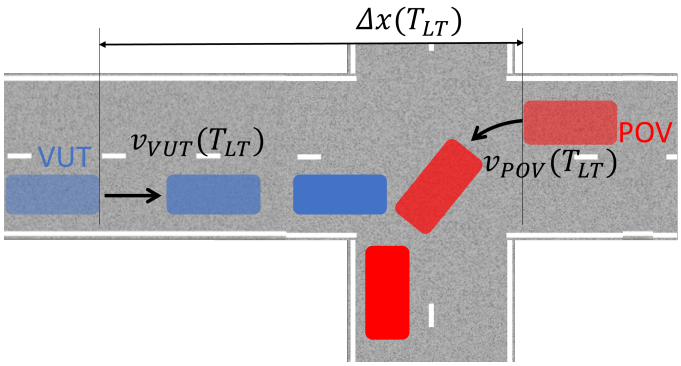


Figure 2.12: Configuration of the ULT scenario.

Cut-in Scenario

In the cut-in scenario, the POV and the VUT drive along a straight road in adjacent lanes with the POV leading. Then, the POV makes a lane change to the VUT's lane, as shown in Figure 2.13. We make the following assumptions on the lane-change maneuver of the POV:

- POV's speed remains constant during the maneuver.
- The heading angle change of the POV is neglected.
- The duration of the lane change is fixed at $T_{LC} = 4.2s$, which is the mean lane change duration according to the Safety Pilot naturalistic driving dataset [33].

- The lateral motion of the lane-change follows a sinusoidal acceleration profile, as suggested in [110].
- The VUT can recognize the cut-in intention of POV no earlier than the starting time of the lane change t_{LC}.

Therefore, the lateral acceleration and displacement profile of the lane change can be written as:

$$a_{LC}(t) = \frac{2\pi W_L}{T_{LC}^2} \sin(\frac{2\pi t}{T_{LC}}) \tag{2.25}$$

$$y_{LC}(t) = -W_L - \frac{W_L}{2\pi} \sin(\frac{2\pi}{T_{LC}} t) + \frac{W_L t}{T_{LC}} \tag{2.26}$$

where W_L is the lane width. For the cut-in scenario, the following key variables are considered, which are measured at t_{LC}:

- Δv: the speed difference; $\Delta v = v_{VUT} - v_{POV}$.
- Δx: the longitudinal distance between the POV and the VUT; $\Delta x = x_{POV} - x_{VUT} - L_0$, where x_{POV} and x_{VUT} are measured at the center of mass of vehicles; L_0 is the nominal length of a vehicle.

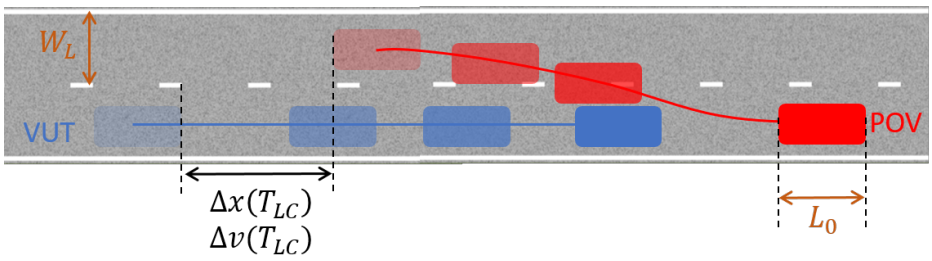


Figure 2.13: Configuration of the cut-in scenario.

2.5.2 Test Case Generation

Unprotected Left Turn (ULT) Scenario

Following the procedure in Section 2.3.5, we first analyze the risk of a given test initial condition. Then, we divide the testing space of the ULT scenario, i.e. $[v_{POV}, v_{VUT}, \Delta x]$, into five RLSs: the sets of infeasible, high-risk, medium-risk, low-risk and trivial cases. We first find the boundary between infeasible and feasible sets using model-based analysis, then lower the capability of the VUT to obtain the boundaries between other RLSs.

First, we define the conflict zone, which is a rectangular region at the center of the intersection. The dimension of the target intersection is shown in Figure 2.14.

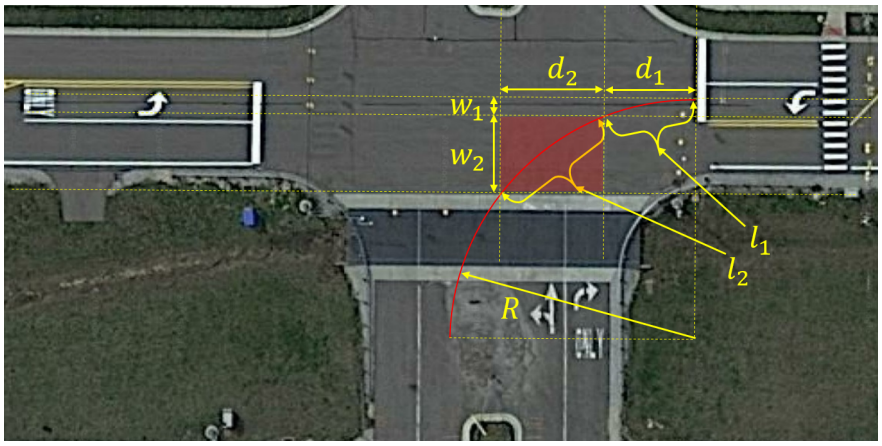


Figure 2.14: The dimension of the intersection and the left-turn maneuver; the conflict zone is shown in red, with width w_2 and length d_2. l_2 is the arc length of the maneuver inside the zone; d_1 and w_1 are the longitudinal and lateral distance from the starting point of the left turn to the edge of the conflict zone, and l_1 is the corresponding arc length. R is the radius of the arc.

When the centers of mass of both vehicles are inside the conflict zone simultaneously, it is deemed a crash and a failed test case for the VUT. To avoid a crash, the VUT has to slow down to not enter the conflict zone until the POV clears it, assuming the POV arrives earlier than the VUT. The braking capability of VUT is assumed as follows: the VUT has a maximum deceleration of a_{min}, which can be achieved after the reaction time t_{react}. During t_{react}, the VUT maintains its speed. For a given initial condition, we can determine whether it is possible to avoid a collision according to the best effort of the VUT. If not, such a case is regarded as infeasible, i.e. a crash is unavoidable. Considering the monotonicity nature of our problem (the smaller Δx is, the harder it is for collision avoidance), we can compute the boundary on Δx for feasible test cases given v_{POV} and v_{VUT}, denoted by Δx_{min}. There are two possible outcomes:

If v_{VUT} is small enough so that VUT can come to a stop before entering the conflict zone, then the minimum distance is calculated as:

$$\Delta x_{min} = v_{VUT} t_{react} - v_{VUT}^2/(2a_{min}) + d_2 + d_1 \tag{2.27}$$

Else, if v_{VUT} is high, and the VUT cannot stop before entering the conflict zone, it may still avoid a crash if it enters the conflict zone after the POV clears it. Then, the minimum distance is calculated as:

$$\Delta x_{min} = v_{VUT}(t_1 + t_2) + 0.5 a_{min}(t_1 + t_2 - t_{react})^2 + d_2 + d_1 \tag{2.28}$$

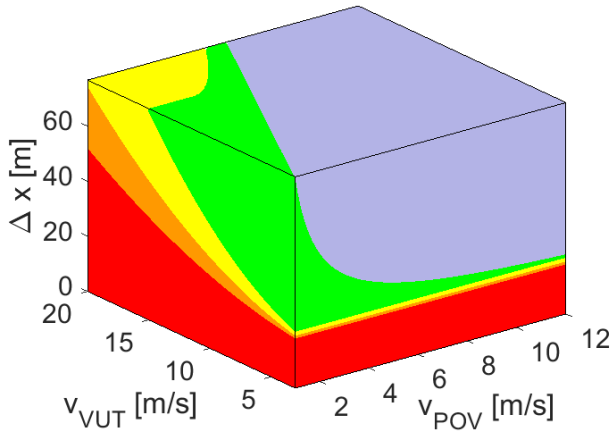


Figure 2.15: The risk level sets for the ULT scenario. Red for the infeasible set, orange for the high risk set, yellow for the medium risk set, green for the low risk set, and purple for the trivial set.

where $t_1 = l_1/v_{POV}$, $t_2 = l_2/v_{POV}$.

The deceleration threshold separating feasible and infeasible cases is set to a_{min} =-0.65g, same as the threshold in the pedestrian crossing scenario. The t_{react} is set to 0.2s, considering the perception, decision-making and actuation delays. Subsequently, the threshold a_{min} to separate other RLSs are the same values as in the pedestrian crossing scenario, i.e. 0.41g for the boundary between high and medium RLSs, 0.23g for medium and low RLSs, and 0 acceleration for low and trivial RLSs. The resulting RLSs for the ULT scenario are shown in Figure 2.15.

Then, we sample test cases from the high, medium and low risk level sets. During a ULT test, the VUT chooses its speed when approaching the intersection. Therefore, the initial conditions to be determined are only v_{POV} and Δx, which lie in a 2-dimensional space. Then, we can repeat the procedure in Section 2.3.6 to generate samples from each RLS stochastically and efficiently according to a given sample ratio R_{IS}. In Figure 2.16, we compare the 300 test cases generated according to the proposed method and according to the naturalistic driving distribution for the ULT scenario from [109]. It shows that cases generated from the naturalistic distribution are mostly trivial or in the low-risk region, resulting in an easy test for the VUT. On the other hand, cases generated from the proposed method can be tuned to spread across risk levels with different ratios, making the test more challenging and efficient. This ratio will decide the acceleration ratio of the evaluation procedure, although knowing the exact acceleration ratio might not be necessary in real-world testing.

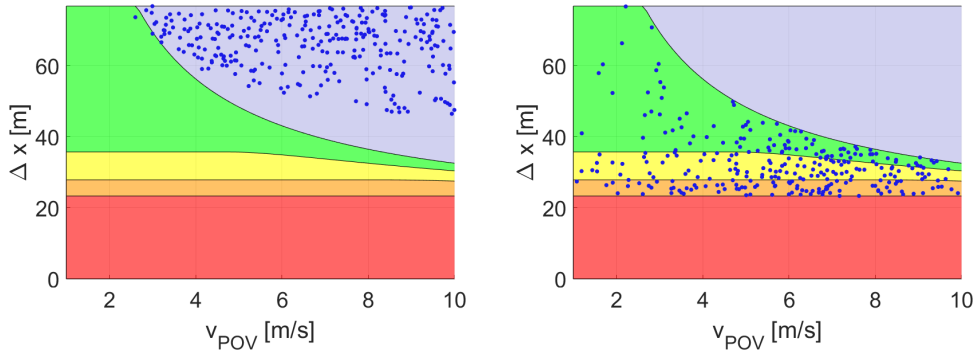


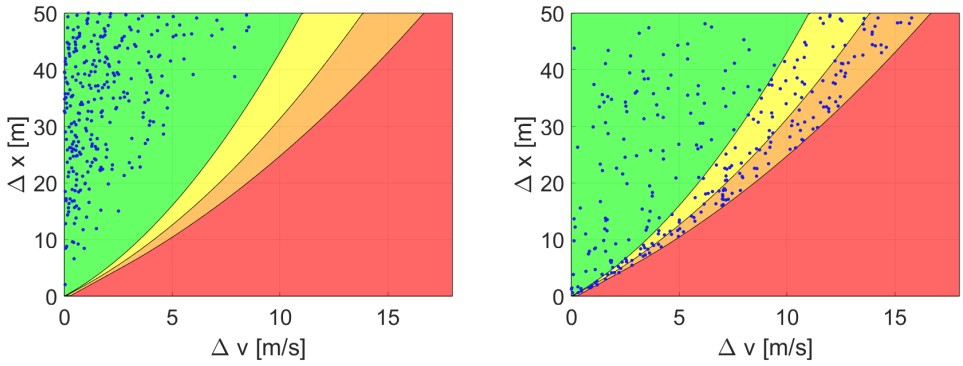
Figure 2.16: Risk level sets and generated test cases for the ULT scenario when $v_{VUT} = 8$ m/s. 300 test cases are generated based on (a) naturalistic distribution and (b) accelerated evaluation distribution with ratio $R_{IS} = [1/3 : 1/3 : 1/3]$. The testing space is partitioned into infeasible (red), high (orange), medium (yellow), low (green) and trivial (blue) risk sets.

Cut-in Scenario

For the cut-in scenario, test parameters are directly sampled from the 2-dimensional space of Δx and Δv. The RLS decomposition of the testing space adopts the same method and threshold parameters as in the ULT scenario. The resulting RLSs are shown in Figure 2.17. There is no trivial set, since all cut-in cases have positive Δv, meaning all cases require deceleration efforts from the VUT to avoid. We also compare the cut-in test cases sampled according to the proposed method and according to the naturalistic driving model for cut-in from [32]. It is shown that the naturalistic distribution rarely generates medium or high-risk test cases, while the proposed method is able to do so, thus accelerating the test procedure in terms of covering the higher-risk region.

2.6 Summary

This chapter presents the accelerated evaluation methodology for reactive scenarios. The main contributions of this chapter include the following: first, a statistical model of pedestrian crossing scenario is built based on 2689 events extracted from open-source video data; second, an accelerated evaluation framework for test case generation is proposed. The test cases are guaranteed to be feasible (i.e. no unavoidable collisions), and have a known level of risk. The usefulness of the evaluation framework is demonstrated in simulations, where the proposed method can achieve unbiased crash rate estimation with only 1/10 of test cases compared to the CMC sampling baseline. In addition, we demonstrate that this framework can be easily extended to two other reactive scenarios, cut-in and unprotected left turn. In summary, the proposed method can efficiently generate



(a) (b)

Figure 2.17: Risk level sets and generated test cases for the cut-in scenario. 300 test cases are generated based on (a) naturalistic distribution and (b) accelerated evaluation distribution with ratio $R_{IS} = [1/3 : 1/3 : 1/3]$. The parameter space is partitioned into infeasible (red), high (orange), medium (yellow) and low (green) risk sets.

test cases with defined risk levels in a variety of reactive scenarios.

CHAPTER 3

Safety Evaluation for Interactive Scenarios

3.1 Motivation and Background

In this chapter, we propose a method for conducting behavior competence testing for interactive scenarios. In interactive scenarios, the POV (mostly representing a human-driven vehicle) and the VUT both can be interaction-aware agents. An interaction-aware agent needs to not only react to the motion of other agents, but also be aware of the impact of itself on others. Therefore, to design a comprehensive evaluation framework for interactive scenarios, extra factors should be considered compared to the reactive scenarios.

First, the diversity in human driver behaviors is not present in reactive scenarios due to the short duration of the challenge phase. In an interactive scenario, with the extended period of the challenge phase, different human drivers may exhibit different behaviors even with the same initial condition, including coasting, accelerating, yielding, etc. This poses challenges to the behavioral prediction and decision-making modules of the VUT. Therefore, the testing space of the evaluation framework should have the modeling capability to capture diverse interactive behaviors.

Second, in interactive scenarios, there is no universal definition of "risk level" as in reactive scenarios. Different VUTs will have vastly different failure modes, which cannot be anticipated prior to the testing. In Chapter 2, the risk of a test case is characterized by the avoidability of a potential collision, which depends on the initial condition. It can be computed with model-based analysis prior to the testing. For example, in the cut-in scenario, the risk is higher if the POV starts the lane change with closer distance and higher relative velocity. Therefore, the failure modes of a given VUT can be predicted. On the other hand, for interactive scenarios, the risk depends on a combination of initial conditions, POV's behavior, and VUT's recognition and response to it, which cannot be analyzed a priori. For example, in a highway merging scenario, the POV is on the main road while the VUT tries to merge from the ramp. A yielding POV may create an easy test case for an aggressive VUT, who can easily merge ahead with hard acceleration; but it can be a failure case for a timid VUT, who tries to slow down as well and creates a dead-lock situation

between the POV and the VUT. In contrast, An aggressively-passing POV can be easily handled by a timid VUT, but not necessarily by an aggressive VUT. The risk is highly dependent on the VUT itself. Therefore, the goal of the test case generation method is to test adaptively for each VUT to discover its failure modes.

3.2 Problem Formulation

In this chapter, the research goal is to systematically evaluate the safety performance of a given VUT in interactive scenarios with limited test cases. The problem can be decomposed into two tasks. First, a testing space will be constructed, which determines all the possible test cases (with interactions) to be evaluated. Second, a mechanism to select test cases from the testing space will be developed. For the first task, the testing space can be characterized by two sets of attributes: the first set defines the initial condition of the scenario, same as in Chapter 2; the second set describes the interactive and behavioral properties of the POV, which then formulates the POV library (introduced in Section 3.4, 3.5 and 3.6). In the second task, the test case generation scheme aims to evaluate the safety performance of a black-box VUT by adaptively discovering its failure modes through efficient sampling schemes (introduced in Section 3.7). We will demonstrate the performance of the proposed method in two typical interactive scenarios, highway merging and roundabout entering (Section 3.8). The overall concept of the proposed interaction-aware evaluation framework is shown in Figure 3.1.

3.3 Review on Interaction-aware Driver Models

Modeling the interaction between human drivers has been a crucial problem in multiple key areas of self-driving research, including behavior prediction, motion planning, V&V, etc. Existing approaches can be categorized into three groups [111]:

1. Rule-based models, e.g., intelligent driver model (IDM) [112], MOBIL model [113].
2. Learning-based models, e.g., variational auto-encoder (VAE) [114], generative adversarial network (GAN) [115].
3. Game-theoretic models [39, 111, 116–123].

Among them, the game-theoretic model blends the interpretability, data-efficiency of rule-based models, and the flexibility of learning-based models, and is adopted as the basis of our approach. Game-theoretic models represent driving as a game. Human drivers are assumed to be rational players that behave (near) optimally according to some utility functions. Nash [116] or Stackelberg

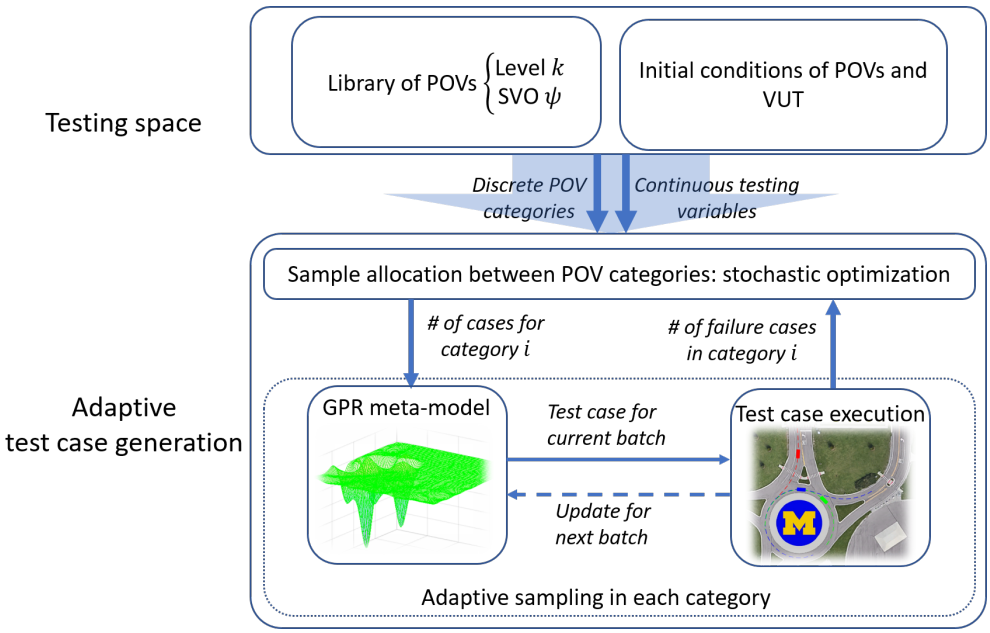


Figure 3.1: The pipeline of the interaction-aware evaluation framework.

[117–119] equilibrium models have been applied to model human driving behaviors. However, they rely on the assumption that each player has an infinite level of rationality, which may not be realistic considering that human drivers have to make quick decisions in a complex environment. Other researchers assumed bounded rationality of human drivers. To model this non-ideal nature of driver behaviors, existing studies have applied level-k game theory [39, 120], quantal response [121] or cumulative prospect theory [111], etc. Among them, level-k game theory has been shown to outperform the equilibrium model in predicting human decision-making behaviors [124]. On the other hand, [116, 119, 122, 123] considered the different social value preferences of human drivers in a game-theoretic setting, which help explain the altruistic or competitive behaviors observed in driving data.

3.4 Basic Methodologies for Primary Other Vehicle (POV) Library Construction

The POV library needs to incorporate interactive driver models that capture the diverse driving styles and behaviors of human drivers for the target scenario. To approximate the decision-making procedure of human drivers, we assume that a POV is a bounded-rational game-theoretic agent, which takes the (near) optimal action with respect to its utility function and assumptions on the opponents.

3.4.1 Markov Game formulation

The problem of solving the optimal policy for one rational agent can be modeled as a Markov Decision Process (MDP), which is defined by $\mathcal{M} = (\mathcal{X}, \mathcal{U}, \mathcal{P}, r, \gamma)$, with the state space $\mathcal{X} \subseteq \mathbb{R}^n$, the action space $\mathcal{U} \subseteq \mathbb{R}^m$, the transition dynamics of the environment $\mathcal{P} : \mathcal{X} \times \mathcal{U} \to \mathcal{X}$, the reward function $r : \mathcal{X} \times \mathcal{U} \to \mathbb{R}$, and the discount factor $\gamma \in (0, 1)$.

When there are multiple rational agents interacting with each other, they can be modeled as a Markov game (MG), which is a generalization of MDP [125], defined by the tuple $\mathcal{G} = (\mathcal{N}, \mathcal{X}, \{\mathcal{U}^i\}_{i\in\mathcal{N}}, \mathcal{P}, \{r^i\}_{i\in\mathcal{N}}, \gamma)$. Other than what is defined in the MDP formulation, $\mathcal{N} = \{1, 2...N_g\}$ denotes the collection of indices of N_g agents; $\mathcal{U}^i$ and r^i denotes the action space and reward function of the i^{th} agent respectively. A policy for agent i is a state-action mapping, i.e. $\pi^i : \mathcal{X} \to \mathcal{U}^i$. The goal of the i^{th} agent is to find a policy π^{i*} that maximizes its expected cumulative reward from any initial state x:

$$\begin{aligned} \pi^{i*}(x) &= \arg\max_{\pi^i} V(x, \pi^i, \pi^{-i}) \\ &= \arg\max_{\pi^i} \mathbb{E}\left[\sum_{t\geq 0} \gamma^t r^i(x_t, u_t) \middle| u^i = \pi^i(x_t), u^{-i} = \pi^{-i}(x_t), x_0 = x \right] \end{aligned} \tag{3.1}$$

In Eq. (3.1), x_t, u_t represent the state and action at time t respectively; $-i$ represents the indices of all agents in $\mathcal{N}$ except agent i.

It is desirable for the POV library to cover a wide range of possible driving behaviors. With the MG formulation, we are able to achieve such modeling capability by either making the POV agent have different assumptions on the opponents' policy π^{-i}, or using different reward functions r_i. Specifically, in this research, we adopt the idea of level-k game theory (for modeling different opponents) and social value orientation (for designing different rewards) to describe the diversified POVs. For simplicity, it is assumed that each POV is involved in a two-player Markov game, i.e. $N_g = 2$, with one opponent vehicle (OV), which could be the VUT or another POV.

3.4.2 Level-k Game Formulation

The level-k game theory model [126] is based on the idea that an intelligent agent (such as a human driver) has a finite level of reasoning depth. For a two-player game with agents A and B, instead of reaching an equilibrium with the opponent assuming that they are infinitely rational, each agent assumes that her/himself is "one level smarter" than the opponent. The model first assumes that a level-0 policy π_0 is known a priori, which could be a naive policy that behaves in a non-interactive way and has no utility function. Then, a level-k agent ($k > 0$) follows a utility-maximizing policy assuming that the opponent is a level-$(k-1)$ agent. Using the level-0 policy as the starting point,

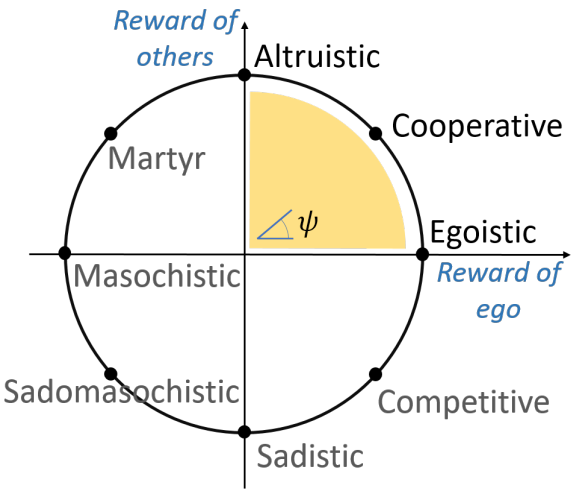


Figure 3.2: The SVO ring: we will focus on $\psi \in [0, \pi/2)$.

the optimal policy for a level-k agent can be generated recursively. Specifically, the optimal policy of a level-k agent A, denoted as π_k^{A*}, can be calculated from:

$$\pi_k^{A*}(x) = \arg\max_{\pi^A} V(x, \pi^A, \pi_{k-1}^{B*}) \tag{3.2}$$

Where π_{k-1}^{B*} denotes the policy of a level-$(k-1)$ agent B. Since π_{k-1}^{B*} is already known and fixed when computing π_k^{A*}, the two-player MG degenerates to an MDP that is easier to solve. On the other hand, since agents at different levels have different assumptions about their opponents, they represent agents with different thinking styles and complexity levels, contributing to the diversity of the POV library. The effectiveness of the level-k game formulation in explaining human driving behaviors has been validated in [127] using real traffic data. According to a study in economics [128], human decision-makers are usually as high as level-2 thinkers. Therefore, we only consider agents that are up to level-2 in this research.

3.4.3 Social Value Orientation

To systematically capture a set of diverse reward functions for the POV, we incorporate the social value orientation (SVO) in the design of the POV library. SVO is a concept from the social psychology literature, which characterizes the degree of selfishness of an agent [129]. It quantifies the preference of an agent regarding optimizing the utility for itself versus for others, which could be represented as an angle ψ on a 2-dimension plane as shown in Figure 3.2. Here, different ψ represents a range of personalities including egoism, altruism, competitiveness, etc. In the original game-theoretic setting, an agent is egoistic and will solely optimize for its own utility function, i.e.

$\psi = 0$. When combining variable SVO with a game-theoretic driver model, as shown in [116], it could improve the accuracy of trajectory prediction, i.e., explain human driving behaviors better. Moreover, agents with different SVO can represent a continuous spectrum of human drivers, which complement the level-k framework where drivers have discrete types and enrich the POV library. In this work, the SVO is combined with the level-k game theory to model POV behaviors.

3.4.4 Combining Level-k with Social Value Orientation

Based on the level-k game theory and SVO, we create a library with the following types of POV agents: the level-0 POV, the level-1 POV, and the level-2 POV with varying SVO. Due to the non-competitive nature of driving tasks, we only consider the SVO angle in the 1^{st} quadrant, i.e. $0 \leq \psi < \pi/2$. The reasons that we do not consider SVO for lower-level POVs are that: a level-0 POV is non-interactive, thus SVO cannot be defined; a level-1 POV assumes its opponent is level-0, which has no utility function, thus SVO is not defined either.

To construct the POV library, we first design the policy for a level-0 POV as a baseline. It is a non-interactive policy with a fixed speed profile longitudinally and laterally, which captures the behavior of inattentive drivers. Next, to generate the policy for a level-k POV $(k > 0)$, a level-$(k-1)$ OV is needed in advance. The level-0 OV policy can be predetermined in the same way as the level-0 POV policy. Therefore, the procedure starts with computing level-0 policies for both POV and OV, and then level-k $(k > 0)$ POV and OV are generated sequentially by assuming a level-$(k-1)$ opponent is known respectively. Although the targets are level-k POVs, level-k OVs are needed as the stepping stones to obtain higher-level POVs.

In the following sections, we will present the details of POV library construction using two important interactive scenarios as examples: highway merging and roundabout entering.

3.5 POV Library for the Highway Merging Scenario

3.5.1 Scenario Model

The configuration of the highway merging scenario is illustrated in Figure 3.3. The VUT attempts to merge onto the highway from the ramp, while the POV is driving on the main road. For the POV, the only OV is the VUT. We make the following assumptions for the scenario:

1. The POV and the VUT see and interact with each other throughout the horizon of the scenario.

2. The POV is not able to change lanes to yield to the VUT; the VUT can only merge at the merge point M, which is the origin of the lane-fixed coordinates for both the ramp and the

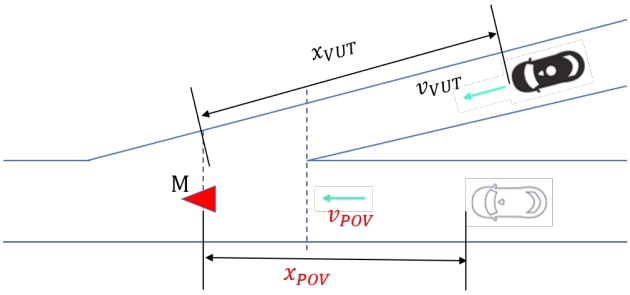


Figure 3.3: The configuration of the highway merging scenario.

main road.

3. There is only one POV on the main road and there is no vehicle in front of the VUT on the ramp.

4. The scenario ends when the VUT reaches point M, and the end time is denoted as t_1.

We model both vehicles as double integrators and they only move longitudinally in their own lane. The discrete-time state-space model of the two-vehicle system can be written as:

$$x(k+1) = f(x(k), u(k)) = A_d x(k) + B_d u(k) \tag{3.3}$$

where

$$A_d = \begin{bmatrix} 1 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \Delta t \\ 0 & 0 & 0 & 1 \end{bmatrix}, B_d = \begin{bmatrix} \frac{1}{2}\Delta t^2 & 0 \\ \Delta t & 0 \\ 0 & \frac{1}{2}\Delta t^2 \\ 0 & \Delta t \end{bmatrix},$$

$$x = [x_{POV}, v_{POV}, x_{VUT}, v_{VUT}]^T, u = [a_{POV}, a_{VUT}]^T.$$

where x_{POV}, x_{VUT} are the longitudinal position, and v_{POV}, v_{VUT} are the longitudinal speed of POV and VUT in their lanes. The input for each vehicle is the longitudinal acceleration, which ranges between $[a_{min}, a_{max}]$. The initial condition is characterized by $[x^0_{POV}, v^0_{POV}, x^0_{VUT}, v^0_{VUT}]^T$. Without loss of generality, we assume x^0_{VUT} is fixed. Moreover, v^0_{VUT} is observed rather than determined by the test conductor. Therefore, the initial condition to sample from is $x_0 = [x^0_{POV}, v^0_{POV}]^T$.

3.5.2 Level-0 Policy

For the highway merging scenario, a level-0 POV is assumed to keep a constant speed, regardless of the VUT. A level-0 VUT will accelerate with constant acceleration (1 m/s^2) until the assumed

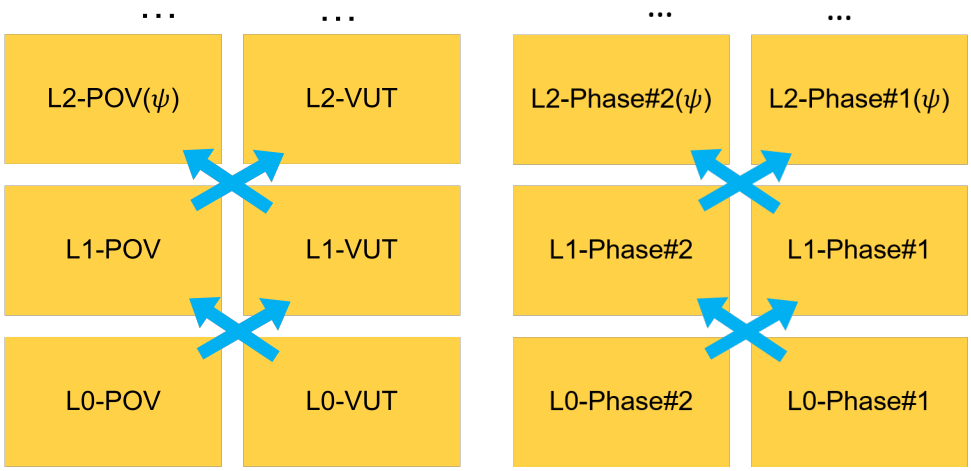


Figure 3.4: The procedure of computing a sequence of level-k agents for (a) the highway merging scenario and (b) the roundabout entering scenarios. The level-2 POVs have an extra parameter ψ characterizing their SVO angle.

highway speed (28 m/s).

3.5.3 POV Behavior Generation Using Reinforcement Learning

To compute the driving policy for a level-k agent ($k > 0$), we use single-agent reinforcement learning (RL) to solve the MDP. To train a level-k POV, we model it as an agent operating in an environment consisting of level-$(k-1)$ VUT. The same idea applies to the training of a level-k VUT. To incorporate the factor of SVO, we consider the SVO angle ψ as an extra state of the model when a level-2 POV is trained to generate a continuum of level-2 POVs. The overall procedure is shown in Figure 3.4a.

For a level-k VUT, the state space of the MDP includes all continuous physical states of the POV and the VUT, denoted as X ($\mathcal{X} = X$). For POVs, the SVO angle is additionally considered in the states space, which is held constant in each episode, i.e. $\mathcal{X} = X \times [0, \pi/2)$. For the highway merging scenario, the physical state space X is 4-dimensional, as shown in Section 3.5.1. The transition dynamics are illustrated in Eq. (3.3), with the opponent's action governed by the level-(k-1) policy. The actions are discrete acceleration choices within a_{min} and a_{max} for both POV and VUT, i.e. $u = a_{POV}/a_{VUT} \in U = \{-4, -3, ..., 0, +1, +2\}$. Each episode terminates when the VUT reaches the merge point $x_{VUT}(t_1) = 0$. The timestep for simulation is $\delta t = 0.1s$, while the timestep for the MDP is $\Delta t = 0.5s$.

The reward function reflects the goal of driving for each agent. We assume that the reward

function can be represented as a linear combination of K reward feature terms:

$$r(x,u) = W^T \Phi(x,u) = \sum_{i=1}^{K} w_i \phi_i(x,u) \tag{3.4}$$

where w_i is the weight of each term, $\phi_i(x,u)$ represents each feature, which represents a different attribute for driving. The rewards can be classified into three categories:

1. Ego reward for POV: $r_{POVe} = W_{POVe}^T \Phi_{POVe}$.
2. Ego reward for VUT: $r_{VUTe} = W_{VUTe}^T \Phi_{VUTe}$.
3. Safety reward for both: $r_{safe} = W_{safe}^T \Phi_{safe}$.

The detailed definitions of the rewards are as follows:

$\Phi_{POVe} = [\phi_{acc}, \phi_{v_{HW}}]^T$, where ϕ_{acc} penalizes acceleration action; $\phi_{v_{HW}}$ penalizes speed exceeding the highway speed limits (either v_{HWmin} or v_{HWmax}). The parameter values are shown in Table 3.1.

$\Phi_{VUTe} = [\phi_{acc}, \phi_{v_{min}}, \phi_{v_{end}}]^T$, where ϕ_{acc} is the same as in Φ_{POVe}; $\phi_{v_{min}}$ penalizes speed lower than a minimum speed v_{min} during the episode; $\phi_{v_{end}}$ penalizes final merging speed of the VUT that is faster or slower than the highway speed limit.

$\Phi_{safe} = [\phi_{TTC}, \phi_{\Delta x}, \phi_{crash}]^T$ are the safety terms evaluated at the end time t_1. We define:

$$\begin{aligned} \Delta x_1 &= x_{POV}(t_1) - x_{VUT}(t_1) \\ \Delta v_1 &= v_{POV}(t_1) - v_{VUT}(t_1) \\ TTC &= \begin{cases} \frac{\Delta x_1}{-\Delta v_1} & \text{when } \Delta x_1 \Delta v_1 < 0 \\ \infty & \text{otherwise} \end{cases} \end{aligned}$$

where ϕ_{TTC} gives penalty when $TTC < TTC_{min}$; $\phi_{\Delta x}$ rewards large $|\Delta x|$, and gives penalty when $|\Delta x| < \Delta x_{critical}$; ϕ_{crash} gives heavy penalty when $|\Delta x| < \Delta x_{crash}$.

Then, the final reward function for a VUT is:

$$r_{VUT} = r_{safe} + r_{VUTe} \tag{3.5}$$

For a POV with SVO angle ψ, the reward function is:

$$r_{POV} = r_{safe} + r_{POVe} \cos(\psi) + r_{VUTe} \sin(\psi) \tag{3.6}$$

where ψ modulates the rewards between POV and VUT. For a level-1 POV, $\psi \equiv 0$.

Parameter	Value	Parameter	Value
v_{HWmax}	35.0 m/s	v_{HWmin}	24.6 m/s
v_{min}	12.0 m/s	TTC_{min}	7.0 s
Δx_{crash}	6 m	$\Delta x_{critical}$	15 m

Table 3.1: Parameters for reward design: highway merging.

To learn the optimal policy for a level-k POV/VUT, we apply the Q-learning method [130]. The action-value function Q is defined as:

$$Q(x, u|\pi) = \mathbb{E}_\pi \left[\sum_{t=0}^{t_1} \gamma^t r_t | x_0 = x, u_0 = u \right] \tag{3.7}$$

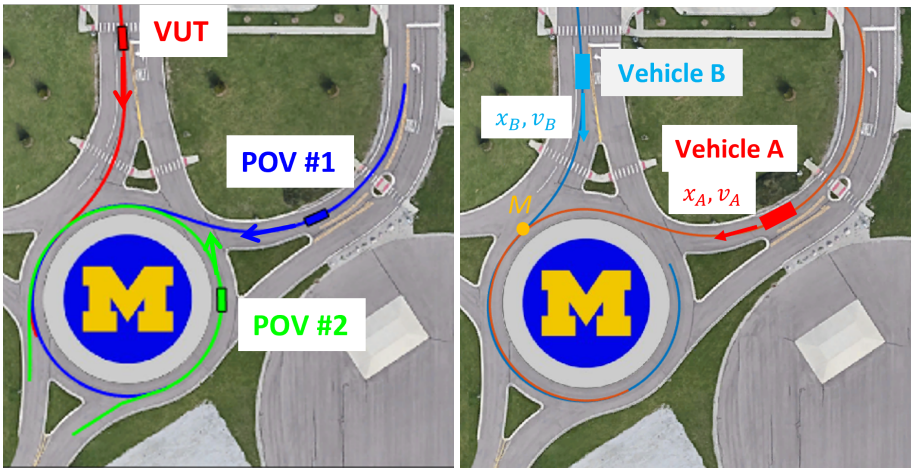
Q-learning uses temporal difference to estimate the optimal Q function, i.e. $Q^*(x, u|\pi^*)$, and learn the optimal policy π^*. For details, please refer to [130]. In this work, the reinforcement learning algorithm we use is Double deep Q-network (DDQN) [131]. DDQN is based on the Deep Q-network (DQN) [132] method. It addresses the problem of overestimating the future return of DQN by decoupling the action evaluation and action selection into max operations in two different Q-networks.

3.6 POV Library for the Roundabout Entering Scenario

Roundabouts are becoming popular in the US due to their significant benefits compared with stop signs and traffic signals [133]. Roundabout driving has been actively studied in the literature [111, 134, 135]. The roundabout entering scenario is a complex interactive scenario due to the multiple entrances and exits, which result in a variety of possible interaction patterns between the subject vehicle and the surrounding vehicles from different branches. This complexity makes designing decision-making and path-planning algorithms for HAVs at roundabouts challenging, and it is even harder for the evaluation of such algorithms. Specifically, vehicles coming from different entrances will predict and influence each other's future motions. Therefore, the interaction between a POV and a VUT, and possibly between multiple POVs need to be modeled by the evaluation framework.

3.6.1 Two-layer Framework for POVs in the Roundabout Scenario

According to common US traffic rules for roundabout entering, e.g., [136], the entering vehicle should yield to all other vehicles that are already in the roundabout or approaching from upstream. For example, in a 4-way roundabout shown in Figure 3.5a, the red vehicle should yield not only



(a) (b)

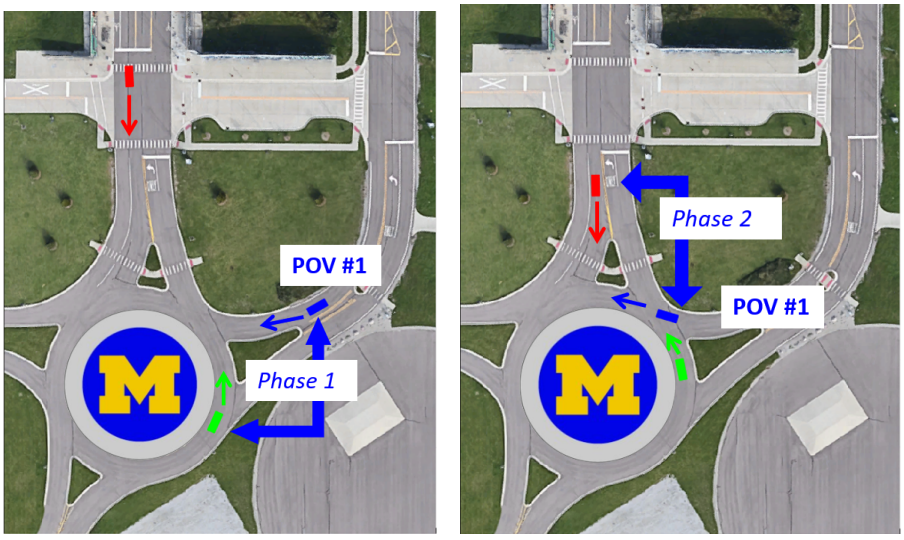
Figure 3.5: (a) The configuration of the roundabout entering scenario and the reference path of the two POVs and one VUT. (b) Definition of variables for each pair of vehicles.

to the closest vehicle to its left, i.e. the green vehicle, but also to the blue vehicle. On the other hand, if the green or blue vehicle offers to yield the right-of-way to the red vehicle, it should also proceed responsively and cautiously. Therefore, since only one POV is present in the scenario formulation in Section 3.5, it cannot represent the diverse situations that an HAV could encounter in a real-world roundabout. In this section, we consider a roundabout scenario with two POVs and one VUT to generate more diverse test cases to challenge the VUT. The proposed framework can scale up to include more POVs.

The geographical layout of the example roundabout for this work is based on a 4-way roundabout inside Mcity, as shown in Figure 3.5a. Each vehicle is assumed to drive along a predetermined reference path. Frenet frame [137] is used to represent the motion of each vehicle along the path.

Though three vehicles are present, we only consider pairwise interaction at any time for simplicity. For each pair, the vehicles are modeled as a double integrator as they move along their reference paths, which are shown in Figure 3.5b. The joint state of the two-vehicle system is $[x_A, v_A, x_B, v_B]^T$. x_A, x_B are the longitudinal position and v_A, v_B are the longitudinal speed of vehicles A and B respectively. Here, all states are defined in the respective Frenet frame. The origin for both frames is the conflict point M, the intersection point of the two reference paths. The input for each vehicle is the longitudinal acceleration, denoted as a_A and a_B.

When multiple POVs are present in the scenario, a key problem is modeling the interaction between POVs. Previous work on decision-making at roundabout [134, 138] assumes that each pair of interactive vehicles are in a fixed two-player game with each other throughout the scenario.



(a) (b)

Figure 3.6: (a) Phase-1 interaction for POV #1; the opponent is the green vehicle. (b) Phase-2 interaction for POV #1; the opponent is the red vehicle.

This formulation not only makes the algorithm not scalable to more vehicles, but also deviates from human driving behaviors. In reality, a vehicle at the roundabout may shift its focus during the scenario. Before entering the roundabout, it will first look for incoming traffic from upstream of the roundabout; after entering, it will ensure safety by looking at downstream vehicles that are about to enter. Therefore, we divide the POV policy into two phases, in which the POV will be involved in different games with different agents in different phases.

- Phase-1: the POV is in phase-1 when it has not entered the roundabout ($x_{POV} < 0$). Since it needs to yield to vehicles from upstream according to traffic rules, it is involved in a two-player game with the closest vehicle upstream, as shown by the blue (POV) and green (OV) vehicles in Figure 3.6a. It assumes that the OV is either non-interactive with the POV (when OV is outside the roundabout) or in its phase-2 (when OV is inside the roundabout). This phase ends when the POV enters the roundabout ($x_{POV} \geq 0$).

- Phase-2: the POV is in phase-2 when it is inside the roundabout ($x_{POV} \geq 0$). It is involved in a two-player game with the vehicle outside the roundabout downstream in the nearest branch, as shown by the blue (POV) and red (OV) vehicles in Figure 3.6b. The opponent is assumed to be in its phase-1. This phase ends when the OV enters the roundabout ($x_{OV} \geq 0$) or when POV passes OV.

Therefore, a POV is involved in one of the two-player games at any given time according to its location with respect to the roundabout. Each phase corresponds to a set of driving policies. Since

the combination of level-k game theory and SVO is used to model the POVs, a level-k phase-1 POV assumes that its opponent is following a level-$(k-1)$ phase-2 policy; by the same token, a level-k phase-2 POV assumes that its opponent is following a level-$(k-1)$ phase-1 policy. In addition, if the POV is in phase-1 but there is no vehicle upstream inside the roundabout, then the POV follows a level-0 phase-1 policy; if the POV is in phase-2 but there is no vehicle downstream in the roundabout, then the POV follows a level-0 phase-2 policy. If there are multiple vehicles in phase-2 (inside the roundabout), each of the following phase-2 POVs will regulate the distance to the preceding vehicle with a rule-based algorithm, in addition to following the nominal phase-2 policy.

Finally, we can combine this two-phase framework with the aforementioned computing procedure for level-k agents to retrieve the policies for all phases and construct the POV library for the roundabout entering scenario. The procedure is demonstrated in Figure 3.4b. We first assume that the level-0 policies for phase-1 and phase-2 are both known: a level-0 phase-2 POV follows a constant speed; a level-0 phase-1 POV decelerates at 1 m/s^2 if $v_{POV} > v_{max}$, then follows a constant speed. Next, we can generate the level-1 phase-1 policy from the level-0 phase-2 policy, and generate the level-1 phase-2 policy from the level-0 phase-1 policy. In this way, we can compute any level-k phase-1 or phase-2 POV policies from the lower-level policies by induction. Finally, a full POV policy is acquired by combining phase-1 and phase-2 policies with the same level and SVO.

In addition, we need to ensure that the behaviors of multiple POVs are compatible with each other. If two POVs have the same level, or one is level-0 and the other is level-2, then both POVs will have wrong assumptions on the opponents, thus collisions may happen. Therefore, we pick four compatible combinations of the two POVs as the "POV categories" for the testing space, which form the POV library that will be considered for this scenario:

1. POV #1 is level-0; POV #2 is level-1;
2. POV #1 is level-1; POV #2 is level-0;
3. POV #1 is level-1; POV #2 is level-2 with any ψ;
4. POV #1 is level-2 with any ψ; POV #2 is level-1.

3.6.2 POV Behavior Generation

We adopt the same RL approach used for the highway merging scenario, with some modifications to the terms in the reward function Eq. (3.4). There are $K = 6$ features, and their definitions are as follows:

Parameter	Value	Parameter	Value
a_{min}	-4.0 m/s^2	a_{max}	3.0 m/s^2
v_{max}	10.0 m/s	TTC_{min}	4.0 s
$d_{critical}$	8.0 m	d_{crash}	6.0 m

Table 3.2: Parameters for reward design: roundabout entering.

1. $\phi_1 = \phi_{acc} = -a_{POV}(t)^2$: negative reward on acceleration action.
2. $\phi_2 = \phi_{step} = -1$: constant step cost to encourage shorter time to finish.
3. $\phi_3 = \phi_{vmax} = -\mathbf{1}_{v_{POV}(t)>v_{max}}$: penalty on over-speeding inside the roundabout; $\mathbf{1}$ is the indicator function.
4. $\phi_4 = \phi_{Dist} = \mathbf{1}_{dist(t)<d_{critical}}(d_{critical} - dist(t))$: $dist$ is the distance to the OV (in Cartesian frame); penalty for being too close.
5. $\phi_5 = \phi_{crash} = \mathbf{1}_{dist(t)<d_{crash}}$: penalty on collision. d_{crash} is the threshold distance for defining a collision between POV and OV.
6. $\phi_6 = \phi_{TTC} = \mathbf{1}_{TTC(t_1)<TTC_{min}}(TTC(t_1) - TTC_{min})$: penalty for the time-to-collision between POV and OV (in the Frenet frame) being too small at the terminal time t_1.

The values of the reward parameters are shown in Table 3.2.

For level-2 POV with SVO angle ψ, the feature ϕ_1 is modified such that the acceleration of the OV is also considered:

$$\phi_1 = -(\cos(\psi)a_{POV}(t)^2 + \sin(\psi)a_{OV}(t)^2)$$

Since ϕ_2 will take effect on both vehicles equally in each episode, and $\phi_4 \sim \phi_6$ are safety features shared by both vehicles, they will not be regulated by the SVO. ϕ_3 will not be considered for SVO because the speed violation of the OV is independent of the action of a POV.

3.7 Adaptive Test Case Generation

3.7.1 Problem Formulation

In the above sections, we systematically generated a library of interactive POVs for each interactive scenario, which is characterized by the SVO ψ and level-k. The testing space can be constructed then by combining the POV library with the initial condition. In this section, we present the method

for generating test cases from the testing space to effectively identify the failure modes of the VUT. We will use the above roundabout-entering scenario as the running example, but the method is directly applicable to all other interactive scenarios. For the two-POV roundabout scenario, the initial condition x_0 is defined as:

$$x_0 = [x^0_{POV1}, x^0_{POV2}, x^0_{VUT}, v^0_{POV1}, v^0_{POV2}, v^0_{VUT}]^T \tag{3.8}$$

By combining x_0 with the SVO of each POV (ψ_1, ψ_2), the category c of two-POV combination, we form the testing space, denoted as $\mathcal{S}$, where each case s is:

$$s = [x_0^T, \psi_1, \psi_2, c]^T \tag{3.9}$$

A test case generation scheme will pick N test cases $\boldsymbol{s} = [s_1, ...s_N]$ from the testing space to locate low-score cases ("failure modes") of the VUT. The main challenge is that different VUTs may have different behaviors and weaknesses, and thus the failure modes are unknown prior to the testing. Therefore, the proposed scheme should select new cases based on past test results to adaptively search for the weaknesses of each VUT as the test proceeds. The goals of the test case generation scheme are two-fold:

1. Challenge: find test cases where the VUT performs poorly (i.e. identify the weakness).

2. Coverage: explore (possibly disjoint) regions of poor performance as many as possible.

The 2nd goal on coverage differentiates our work from other research on falsification-based testing, in which the goal is just to find the "worst case" by solving a minimization problem (e.g. [35]).

For each test case s and a given VUT, we define the *performance score* $P(s)$ as:

$$P(s) = \mu_1 I_{crash}(\boldsymbol{\tau}) + \mu_2 P_{safety}(\boldsymbol{\tau}) + \mu_3 P_{task}(\boldsymbol{\tau}) \tag{3.10}$$

where $\boldsymbol{\tau}$ is the resulting joint trajectory of POVs and the VUT; I_{crash} is the indicator for collision; P_{safety} is the safety score; P_{task} is the score of task accomplishment; μ_1, μ_2, μ_3 are weighting factors. The failure modes of a VUT are defined as: $\mathcal{S}_f(\lambda) = \{s | s \in \mathcal{S}, P(s) < \lambda\}$, where λ is the performance score threshold for a failure. All cases in $\mathcal{S}_f(\lambda)$ are failure cases. The key objectives can be translated to maximizing the coverage of the failure modes.

3.7.2 Adaptive Testing Method Overview

We will generate N test cases in batches, with batch size n_b. For the testing space $\mathcal{S}$, the last attribute for the POV category $c \in C$ is a categorical variable, while all others are continuous

variables. Subsequently, the testing space can be decomposed into two parts, written as: $\mathcal{S} = S \times C$, where S is the subspace with continuous variables.

Sampling from continuous and categorical attributes need to be treated differently, since samples in one category have little correlation with samples in another category. Therefore, the test case generation scheme can be divided into two stages for each batch, as shown in the lower part of Figure 3.1. In the 1st stage, we allocate the quota of samples into different POV categories, i.e. assign n_c^i cases to category c at batch i. Sample allocation methods will be introduced in Section 3.7.5. In the 2nd stage, we sample new test cases within each category from S using an adaptive sampling method based on the Gaussian process regression (GPR). We will first introduce the proposed method for the 2nd-stage next.

3.7.3 Adaptive Sampling within Single POV Category

We start with the 2nd-stage of test case generation, which operates on the continuous test attributes. First, to assess the quality of test samples in each POV category, we define the criterion FMC to formally characterize the failure mode coverage in each subspace S:

$$\text{FMC}(\boldsymbol{s}_c, \rho, \lambda) = \int_{\bigcup B_{dim}(\rho, \boldsymbol{s}_\lambda)} \mathbf{1}\, dv \tag{3.11}$$

In Eq. (3.11), $\boldsymbol{s}_c$ are the test cases within the c^{th} POV category, dim is the dimensionality of S, and $B_{dim}(\rho, s)$ is a dim-dimensional hyper-ball centered around case s with radius ρ. $\boldsymbol{s}_\lambda = \mathcal{S}_f(\lambda) \bigcap \boldsymbol{s}_c$, which consists of all the failure cases in $\boldsymbol{s}_c$. The FMC evaluates the generalized volume of the union of hyper-balls centered around identified failure cases, which characterizes the coverage. Here, all dimensions of S are normalized between [0,1]. Figure 3.7 is a graphic illustration of the FMC in 1 dimension.

Within each POV category, we conduct adaptive sampling with a Gaussian process regression (GPR) meta-model. The adaptive sampling method alternates between the two steps: updating meta-models with prior testing results and generating samples from the new meta-models [139]. GPR is a non-parametric probabilistic model [140] that is popular for its ability to model complex functions and infer the value around unexplored regions [141]. The key idea is to maintain and update a GPR-based meta-model according to existing samples, and use the meta-model to generate new samples.

Gaussian process (GP) is a stochastic process, for which the joint distribution of every finite collection of random variables follows a multivariate Gaussian distribution. A GP is characterized

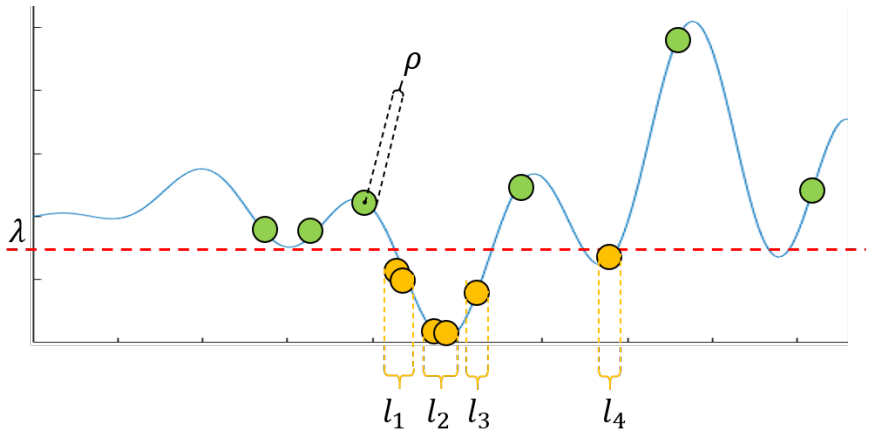


Figure 3.7: Measuring the failure mode coverage (FMC) of test samples on a 1-dimension continuous testing space. The blue curve shows the performance score $P(s)$, the red dashed line represents the score threshold λ, and the regions where the curve is under λ are the failure modes. For this example, FMC$(\boldsymbol{s}, \rho, \lambda) = l_1 + l_2 + l_3 + l_4$.

by its mean function $m(s)$ and covariance function $k(s, s')$ (kernel), as shown in Eq. (3.12).

$$f(s) \sim GP(m(s), k(s, s')) \tag{3.12}$$

In this work, we use GP as the surrogate model of the performance score profile of each VUT, as shown in Eq. (3.13). GP characterizes the prior belief of $P(s)$, which is updated by the new test samples and their test results.

$$P(s) = \delta + f(s), \text{ where } f(s) \sim GP(0, k(s, s'|\theta)),\ \delta \sim \mathbf{N}(\beta, \sigma^{\mathbf{2}}) \tag{3.13}$$

In Eq. (3.13), θ is the kernel parameter, (β, σ, θ) constitute the hyper-parameters of the model. In this work, we use a zero mean function and a square exponential kernel function for the GPR model, as shown in Eq. (3.14), where $\theta = [\theta_1, \theta_2]^T$. Hyper-parameters are optimized using maximum likelihood estimation.

$$k(s, s'|\theta) = \theta_1^2 \exp\left(-\frac{(s - s')^T (s - s')}{2\theta_2^2}\right) \tag{3.14}$$

With a GPR model $\hat{P}(.)$ based on existing testing cases $\boldsymbol{s}$ and their results $\boldsymbol{y}$, for any unobserved query case s^0, the joint distribution of $\hat{P}(s^0|\boldsymbol{s}, \boldsymbol{y})$ and $\boldsymbol{y}$ is also a Gaussian distribution. Therefore,

the conditional mean and variance of $\hat{P}(s^0|\boldsymbol{s},\boldsymbol{y})$ are:

$$\begin{aligned}\mathbb{E}\left(\hat{P}(s^0|\boldsymbol{s},\boldsymbol{y})\right) =& k(s^0,\boldsymbol{s})(k(\boldsymbol{s},\boldsymbol{s})+\sigma^2 I)^{-1}(\boldsymbol{y}-\beta)\\ \mathbb{V}ar\left(\hat{P}(s^0|\boldsymbol{s},\boldsymbol{y})\right) =& k(s^0,s^0)-k(s^0,\boldsymbol{s})(k(\boldsymbol{s},\boldsymbol{s})+\sigma^2 I)^{-1}k(\boldsymbol{s},s^0)\end{aligned} \tag{3.15}$$

The procedure of adaptive sampling is illustrated in Algorithm 1. Details about selecting new samples using the meta-model (lines 6-9) are explained in Section 3.7.4.

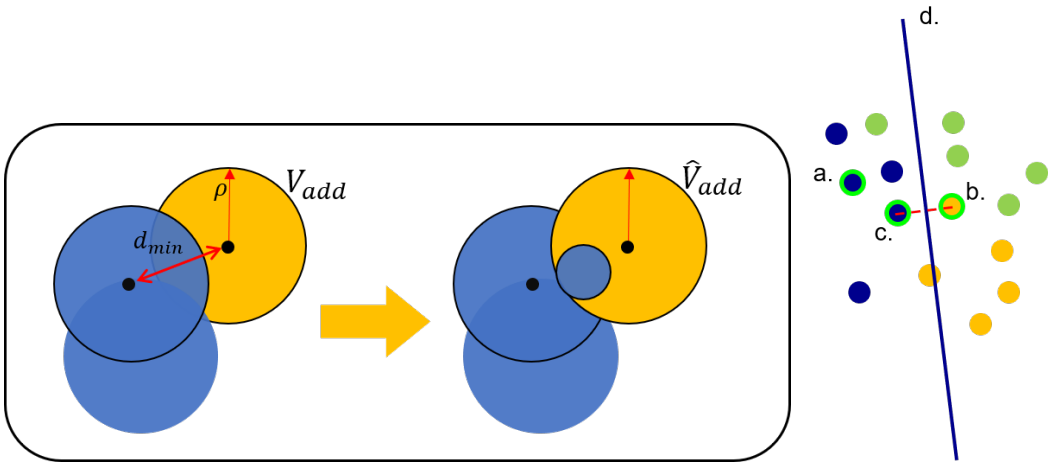
Algorithm 1 Adaptive sampling for one step

Input: batches number i; batch size n_c^i; previous GPR model $\hat{P}_c^{i-1}$; exploration factor ϵ_0.
Output: test cases with POVs of category c, $\boldsymbol{s}_c^i$, and test results $\boldsymbol{y}_c^i$; updated GPR model $\hat{P}_c^i$.

1: **if** $i=1$ **then**
2: Sample initial test batch $\boldsymbol{s}_c^1$ uniformly from S.
3: **else**
4: Randomly sample p queries $\check{\boldsymbol{s}}$ from S based on uniform distribution ($p \gg n_c^i$).
5: $\epsilon = \epsilon_0 \alpha^{i-1}$.
6: Pick $(1-\epsilon)n_c^i$ queries from $\check{\boldsymbol{s}}$ according to $\mathcal{C}_{exploit}(s)$ as exploitation samples, denoted as $\boldsymbol{s}_{exploit}$.
7: Pick $\epsilon n_c^i(1-r_{BMB})$ queries from $\check{\boldsymbol{s}}$ according to $\mathcal{C}_{explore}(s)$ as a part of exploration samples, denoted as $\boldsymbol{s}_{explore\text{-}1}$.
8: Sample $\epsilon n_c^i r_{BMB}$ cases according to Algorithm 2 as the rest of exploration samples, denoted as $\boldsymbol{s}_{explore\text{-}2}$.
9: $\boldsymbol{s}_c^i = [\boldsymbol{s}_{exploit}, \boldsymbol{s}_{explore\text{-}1}, \boldsymbol{s}_{explore\text{-}2}]$.
10: **end if**
11: Execute test cases $\boldsymbol{s}_c^i$, acquire results $\boldsymbol{y}_c^i = P(\boldsymbol{s}_c^i)$.
12: Fit/Update the GPR model: $y = \hat{P}_c^i(s) = \hat{P}(s|\boldsymbol{s}_c^{1:i}, \boldsymbol{y}_c^{1:i})$.

3.7.4 Test Case Selection

To achieve good coverage of the failure modes, we need to balance exploitation and exploration when choosing a new batch of samples. On the one hand, samples with low predicted $\hat{P}(s)$ represent more challenging cases, which are preferred for the goal of challenge. On the other hand, it is desirable to explore regions with high uncertainty to pick more informative samples for a better meta-model, which helps coverage. We will describe the criteria $\mathcal{C}(s)$ that evaluate the potential quality of a query s for both exploration and exploitation, and how to balance between these two goals.



(a) (b)

Figure 3.8: (a) Compute expected FMC improvement through volume approximation. The yellow region on the left shows the real V_{add} brought by the new sample, while on the right shows the approximated $\hat{V}_{add}$. (b) Sample along the performance boundary. Points with different colors belong to different behavior modes. a.,b.,c.,d. correspond to line 3,4,5,6 in Algorithm 2 respectively.

Exploitation: Modified Expected Improvement (MEI) Criterion

Expected improvement (EI) is a popular acquisition function for choosing the most promising samples in Bayesian optimization [142], a closely-related method with the GPR-based adaptive sampling. EI indicates the possible improvement in the optimal value brought by the new query.

$$\begin{aligned} \mathcal{C}(s) &= \mathbb{E}(I(s)) \\ \text{where } I(s) &= \max(P_{min} - \hat{P}(s), 0) \end{aligned} \tag{3.16}$$

In Eq. (3.16), P_{min} is the minimal performance score achieved by existing test samples. To better suit our problem settings and goals, we modify the EI criterion to explicitly consider the possible FMC improvement from the new query:

$$\begin{aligned} \mathcal{C}_{exploit}(s) &= \mathbb{E}(I(s)) \times V_{add} \\ \text{where } I(s) &= \max(\lambda - \hat{P}(s), 0) \end{aligned} \tag{3.17}$$

In Eq. (3.17), V_{add} is the additional volume covered by $B_{dim}(\rho, s)$. Since the actual V_{add} is hard to compute due to its highly non-convex shape, we use an approximation of V_{add} for the above computation, as demonstrated in Figure 3.8a and Eq. (3.18):

$$V_{add} \approx \hat{V}_{add} \propto \rho^{dim} - (\max(\rho - d_{min}(s), 0))^{dim} \tag{3.18}$$

where $d_{min}(s)$ is the distance from s to the nearest identified failure case. A sample with a high MEI means that it has a high chance of being a failure case, and it is far from previous failure cases.

Exploration #1: Standard Deviation Criterion

For exploration, a straightforward query criterion is the posterior standard deviation.

$$\mathcal{C}_{explore}(s) = \hat{\sigma}(s) \tag{3.19}$$

where $\hat{\sigma}(s) = \sqrt{\mathbb{V}ar[\hat{P}(s)]}$. Higher $\mathcal{C}_{explore}(s)$ indicates that the current GPR meta-model has higher uncertainty at s, i.e., higher exploration return.

Exploration #2: Sampling on the Behavior Mode Boundary (BMB)

In the aforementioned sampling schemes, the only information from past test cases that aids the adaptive sampling is the performance score. However, utilizing more contextual information of test cases for selecting new test queries can be beneficial, especially for a high-dimensional testing space. Therefore, we consider the behavior mode of a test case as additional information to guide adaptive sampling. The idea is inspired by [68], where the performance mode boundaries of a system under test are identified in hope of generating informative test cases. The intuition is that a switch between performance modes has the potential to induce confusion and fail the VUT.

We define the behavior mode boundary (BMB) for our test scenario. In the roundabout entering scenario with two POVs, the BMB is defined as the final passing order of the VUT. There are three BMBs: VUT being the first to pass, the second to pass or the last to pass.

The process of generating test cases using BMB is illustrated in Algorithm 2. Here, the BMB is estimated locally using existing test cases of different behavior modes. Then, new test cases are sampled along the estimated BMB. The procedure is visualized in Figure 3.8b. Therefore, the BMB becomes another heuristic that guides exploratory sampling in the testing space.

Balancing Exploration and Exploitation

For each batch, We pick test cases according to the three criteria above. The percentage of cases for exploration and exploitation are determined by the parameter ϵ, which gradually decreases at the rate of α ($\alpha \in (0.9, 1)$), such that the procedure will start with more exploration, and bias towards exploitation as more data are collected and a better meta-model is built. Between the two exploration criteria, we allocate a fixed ratio r_{BMB} of all exploration cases to be sampled from the BMBs, while the others are chosen based on the standard deviation criterion (e.g. $r_{BMB} = 1/4$).

Algorithm 2 Sampling near the behavior mode boundary

Input: number of cases to be sampled along BMBs n_{BMB}; all previous test cases $\boldsymbol{s}$, and their corresponding behavior modes $BM^{\boldsymbol{s}}$.

Output: $\boldsymbol{s}_{BMB}$: test cases sampled at the behavior mode boundaries.

$s_{BMB} = [\,]$
for $k \leftarrow 1$ to n_{BMB} **do**
    Randomly pick two behavior modes, $BM_1 \,\&\, BM_2$
    Randomly pick a point from mode BM_1: $\hat{s}_1'$.
    Find the closest neighbor of $\hat{s}_1'$ in mode BM_2: $\hat{s}_2$.
    Find the closest neighbor of $\hat{s}_2$ in BM_1: $\hat{s}_1$
    Draw a sample $\hat{s}$ randomly on the bisection subspace of $\hat{s}_1$ and $\hat{s}_2$: $\boldsymbol{s}_{BMB} = [\boldsymbol{s}_{BMB}, \hat{s}]$
end for

It is important to note that the proposed method has different goals compared to the Bayesian optimization methods though they have similar formulations and usages of the GPR model and the EI criterion. The Bayesian optimization focuses on solving an optimization problem, i.e. finding one global optimal point on the performance surface, while the proposed adaptive sampling method focuses on identifying failure modes, i.e. finding more "valleys" of the performance surface. Therefore, the exploration criteria are added compared to standard Bayesian optimization methods.

3.7.5 Sample Allocation between POV Categories

In this section, we discuss the 1st stage of the adaptive testing framework, i.e. how to optimally distribute test samples into different categories of POVs. Since the potential of finding failure cases in each category is unknown a priori, the sample allocation method needs to balance exploration (finding out which category is the most promising) and exploitation (sample more from the most promising category). The key objective can take two different forms: (1) to maximize the total number of identified failure cases from all categories; (2) to find more failure cases in total, while preferring failure cases that are scattered in more categories. Since both forms of the objective have their significance and applicability, we propose two sample allocation methods to handle them separately.

Balancing Failure Case Number and Diversity

To handle the trade-off between the diversity and richness of the failure cases, we propose to find the best allocation policy by solving a stochastic optimization problem.

First, we define the failure case reward for each category as a function of the number of failure

cases $\bar{n}_c^i$ at batch i and category $c \in C$, denoted as β_c^i:

$$\beta_c^i = h(\bar{n}_c^i) \tag{3.20}$$

$h(.)$ is a concave and monotonically increasing function defined over $[0, +\infty]$. In this research, we choose $h(x) = x^\tau, \tau \in (0, 1)$. Such an $h(.)$ will encourage more failure cases in each category. On the other hand, the reward for each extra failure case of the same category will diminish ($dh/d\bar{n}_c^i$ decreases as $\bar{n}_c^i$ increases), which encourages finding failure cases in other categories rather than concentrated in a single category, i.e., exploration is encouraged.

We define the optimization problem below. Since $\bar{n}_c^i$ is unknown beforehand, we will estimate it according to the results from previous iterations, which can be written as $\bar{n}_c^i = \eta_c^i n_b q_c^i$. Here, η_c^i is the variable being optimized, the ratio of samples assigned to category c at batch i; n_b is the batch size, and $q_c^i \in [0, 1]$ is a random variable describing the belief on the failure case probability (FCP) of c^{th} category in batch i. Specifically, the test-case sampling from each category can be approximated as independent Bernoulli trials, with the "probability of successfully finding a failure case in each trial" being q_c^i. The goal is to maximize the overall expected failure case reward R_{fc}, as stated in Eq. (3.21).

$$\begin{aligned} \max_{\eta_c^i} \quad & \mathbb{E}_{q_c^i}(R_{fc}) = \sum_{c=1}^{|C|} \mathbb{E}_{q_c^i}(\beta_c^i) \\ \text{s.t.} \quad & \sum_{c=1}^{|C|} \eta_c^i = 1 \\ & \eta_c^i \geq 0, \quad c = 1....|C| \end{aligned} \tag{3.21}$$

$|C|$ is the cardinality of the set of categories C. q_c^i can be modeled by a Beta distribution: $q_c^i \sim \text{Beta}(\alpha_c^i, \beta_c^i)$, where α_c^i, β_c^i are the parameters that control the shape of the Beta distribution [143]. Without loss of generality, we assume $q_c^1 \sim \text{Beta}(1, 1)$, i.e. q_c^1 has a uniform prior. Assuming that for batch i and category c, a_c^i failure cases and b_c^i non-failure cases were found, and we denote the observation D_c^i as $D_c^i = [a_c^i, b_c^i]$. Then, we can conduct a Bayesian update on q_c^i according to the existing testing results:

$$\begin{aligned} Pr(q_c^i | D_c^i) &\propto Pr(D_c^i | q_c^i) Pr(q_c^i) \\ Pr(q_c^{i+1}) &= Pr(q_c^i | D_c^i), \end{aligned} \tag{3.22}$$

where the 1st line is the Bayes theorem; the 2nd line states that the the prior of q_c^{i+1} is defined as the posterior of q_c^i. Since Beta distribution is the conjugate prior of the binomial distribution [143], the posterior distribution of q_c^i will have the same form, i.e. a Beta distribution. Therefore, we can

compute the distribution of q_c^{i+1} with the following updating rules on its parameters:

$$\begin{aligned} q_c^{i+1} &\sim \text{Beta}(\alpha_c^{i+1}, \beta_c^{i+1}), \text{ where} \\ \alpha_c^{i+1} &= a_c^i + \hat{\gamma}(\alpha_c^i - 1) \\ \beta_c^{i+1} &= b_c^i + \hat{\gamma}(\beta_c^i - 1) \end{aligned} \tag{3.23}$$

When the constant $\hat{\gamma} = 1$, Eq. (3.23) is equivalent to Eq. (3.22). However, in the adaptive sampling procedure, we acquire more knowledge about the failure modes with more test cases, and subsequently, the FCP q_c^i will likely not stay stationary with respect to i. Therefore, we assume $\hat{\gamma} \in (0, 1)$ and use it as a discounting factor to attenuate earlier information.

In each iteration, we solve the optimization problem Eq. (3.21) to acquire an optimal allocation of samples in each category, i.e. η_c^i. This strategy will invest more samples in better-performing categories, while maintaining some samples in all categories.

Greedy Optimization for Failure Case Number

When the only goal is to find as many failure cases as possible, we model the reward of each sample as a binary variable, i.e. 1 if the sample is a failure case, and 0 if not. Then, this problem is akin to the stochastic multi-arm bandit (MAB) problem [144].

In a MAB problem, one needs to allocate limited resources to competing choices to maximize the expected reward (minimize the total expected regret). Assume we have N rounds to play and have K choices (arms) to choose from for each round, $l_t \in L = \{1, ...K\}$. Each choice has a stochastic reward, whose distribution is unknown a priori, denoted as ν_l, $l \in L$. We receives a reward $X_t \sim \nu_{l_t}$ for the t^{th} round. The goal is to minimize the expected regret, defined as:

$$\mathbb{E}(R_N) = N\mu^* - \sum_{t=1}^{N} \mathbb{E}(X_t) \tag{3.24}$$

where $\mu^* = \max_{l \in L} \mathbb{E}_{X \sim \nu_l}(X)$ is the expected reward for the best arm.

There have been many classical algorithms for solving the MAB, including upper confidence bound (UCB) [145], Thompson sampling (TS) [146], etc. Both methods can achieve sub-linear regret with respect to the number of samples [144]. In standard MAB formulation, the reward for each choice follows a stationary distribution. However, in our problem setting, the goodness of each category (i.e. the q_c^i) will change over time due to more informed sampling, as explained in the above section. Therefore, directly applying the above methods will not yield ideal results.

In this paper, we propose a modified version of the UCB algorithm for sample allocation, named "UCB-dynamic". At each time-step t, We pick a new sample from category c_t^{UCB} based on the

following criterion:

$$c_t^{UCB} = \arg\max_{c \in C} \hat{Q}_t(c) + U_t(c)$$
$$\text{where } U_t(c) = \sqrt{\frac{2 \log t}{N_t(c)}}, \hat{Q}_t(c) = \mathbb{E}(q_c^i). \tag{3.25}$$

Here, i is the batch number that the t^{th} sample belongs to; $\hat{Q}_t(c)$ and $U_t(c)$ are the estimated reward and upper confidence bound on that reward estimation respectively for category c; $N_t(c)$ is the number of total samples in category c as of time t. The algorithm starts by exploring the under-discovered categories ($U_t(c)$ dominated). Then it gradually turns to exploit the category with the best estimated reward ($\hat{Q}_t(c)$ dominated). There are several modifications we made compared to the standard UCB algorithm:

1. In a standard UCB, $\hat{Q}_t(c)$ equals the sample mean of the reward, i.e. the cumulative ratio of failure cases for each category. In our setting, since the reward distribution for each category is not stationary, we set $\hat{Q}_t(c) = q_c^i$ to ensure that the estimated reward adapts to the dynamics of the FCP.

2. Since we generate and execute test samples in batches, $\hat{Q}_t(c)$ will stay constant throughout each batch, whereas $\hat{Q}_t(c)$ is updated at every new sample in the standard UCB.

It is worth noting that the POV category is just one example of categorical variables in the testing space. The proposed methods can be directly applied to other testing space designs with categorical attributes like weather, lighting conditions, vehicle/VRU types, etc.

3.8 Simulation Results

In this section, we first evaluate the performance of the proposed sample allocation methods using fictitious experiments. Then, we show some example test cases generated for both highway merging and roundabout entering scenarios to demonstrate the behavior-generation capability of the proposed POV library scheme. Finally, we conduct interaction-aware testing at the roundabout entering scenario in simulation to validate the effectiveness of the proposed test case generation scheme.

3.8.1 Comparison of Sample Allocation Methods

We evaluate the two sample allocation methods proposed in Section 3.7.5 using four fictitious experiments. During the test case generation procedure, the sample allocation (1st stage) is followed by adaptive sampling within each category (2nd stage), which will introduce uncertainty due to

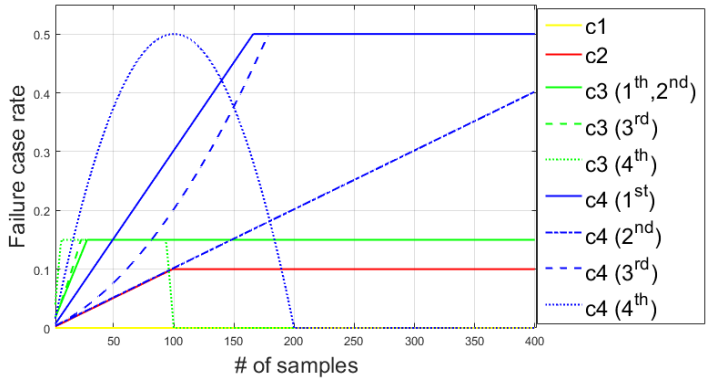


Figure 3.9: The FCP dynamics for each category for the 4 simulated experiments. The experiment indices are shown in parentheses.

the randomness in sampling. To reduce the impact of the 2nd-stage process and to compare the performance of the sample allocation methods in isolation, we abstract the 2nd stage into Bernoulli trials with varying parameters for this comparison. Here is the simulation setup:

- Each test sample is an independent trial with a binary outcome, either 0 (not a failure case) or 1 (failure case). The probability of a case being a failure case is $p_c(N_c) \in [0, 1)$, which depends on the property of the category c, and the number of samples so far in that category, denoted as N_c.

- There are four categories ($C = \{\text{c1,c2,c3,c4}\}$), each with a different FCP dynamics $p_c(N_c), c \in C$. Each category has a fixed maximum FCP.

- We conduct four experiments. In all of them, no failure cases could be found in c1 ($p_{c1} = 0$); p_{c2} is set to be the same linear function; p_{c3}, p_{c4} are linear in the first two experiments, and quadratic in the last two experiments, each with different parameters. The presented FCP dynamics are meant to mimic the different phenomena in adaptive sampling, including increased FCP due to more informed sampling, or decreased FCP due to depleted failure modes, etc. The dynamics of the FCP for the comparison experiments are shown in Figure 3.9.

- In each run, cases are generated in 20 batches, each batch with 60 cases. This is the same setting as the final experiments for interaction-aware testing in Section 3.8.4. 100 repeated runs are conducted for each method in one experiment.

We compare the proposed methods (stochastic optimization and UCB-dynamic) with other popular algorithms for the MAB problem, including the allocation method from the previous paper [147]; the standard UCB algorithm [145]; the Thompson sampling method with the same modification on

Method	Mean # of Failure Cases			
	1st exp.	2nd exp.	3rd exp.	4th exp.
Stochastic optimization	366.5	151.3	302.9	121.2
UCB-dynamic	420.6	**209.8**	**372.8**	**125.2**
Heuristics in [147]	288.0	135.5	262.8	111.2
UCB	**421.5**	182.3	356.6	111.0
TS-dynamic	394.3	147.8	280.1	116.0
Uniform allocation	172.4	108.7	158.2	105.1

Table 3.3: Comparison of the number of failure cases for different sample allocation methods.

reward update as UCB-dynamic (named as TS-dynamic); and the most basic baseline, the uniform allocation. The comparison metric is the average number of failure cases in the 100 repeated runs. The results are shown in Table 3.3.

It is shown that the proposed UCB-dynamic method outperforms the other methods. It can find the most failure cases across all the experiments, except a close-2nd place in the 1st experiment. Though the goodness of each category changes in various ways in different experiments, the proposed method can always allocate samples wisely to all categories. Compared with the standard UCB, UCB-dynamic shows significant improvement in 3 out of 4 experiments. The TS-dynamic method, though received the same treatment as UCB-dynamic, has uniformly inferior performance. The stochastic optimization method sacrifices the count of failure cases to diversity, but it still outperforms the method from our previous work [147].

3.8.2 Baseline Algorithm for the Vehicle Under Test (VUT)

Highway Merging Scenario

For the highway merging scenario, we design a rule-based algorithm for the merging vehicle (the VUT). This is a flawed algorithm by design, such that it has failure modes to be discovered. Its decision-making procedure has 3 modes:

1. The VUT starts by following the speed profile of a level-0 VUT policy π^0_{VUT}. Go to mode 2 when the distance to the merge point M is x^{rb}_1.

2. The VUT predicts Δx with the POV when arriving at M, assuming the POV keeps a constant speed, and VUT follows π^0_{VUT}. If Δx is too close, switch to coast; else, keep following π^0_{VUT}. Go to mode 3 when the distance to point M is x^{rb}_2 ($x^{rb}_2 < x^{rb}_1$).

3. The VUT predicts Δx with the POV when arriving at M, assuming POV keeps a constant

speed, and VUT follows π^0_{VUT}. If the predicted Δx is too close, the VUT switches to PID-control to change acceleration reactively; if not, it keeps following π^0_{VUT}.

By adjusting the parameters, we can have VUT designs that have different failure modes.

Roundabout Entering Scenario

For the roundabout scenario, we design a rule-based speed planning algorithm for the VUT, which is also flawed by design. Its decision-making has 3 modes:

1. The VUT starts by coasting at a constant speed. Go to mode 2 when it is x^{rb1} close to point M in Figure 3.5b.

2. The speed of the VUT is regulated by a PID controller to a target speed v_{tar}. The VUT predicts the projected final gap with both POVs, and makes an online decision between three options: yield (to both), slip between (two POVs), or pass in front (of both). If "yield" or "slip in", then decelerate (set $v_{tar} = 0$); Else, $v_{tar} = v_0$, v_0 is a default desired speed. Go to mode 3 when the distance from VUT to point M is less than x^{rb2} ($x^{rb2} < x^{rb1}$).

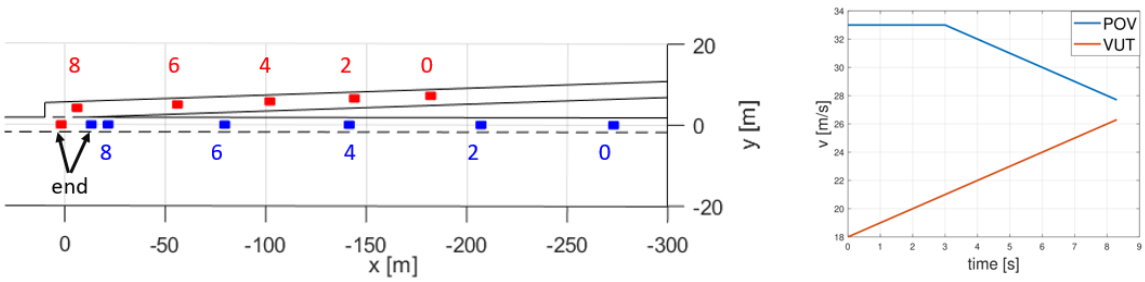
3. Execute the last decision from mode 2. If "yield", wait for both POVs to pass, and then perform car-following with the last POV; if "slip in", wait for the 1st POV to pass, then perform car-following with the first POV; if "pass", then maintain the PID control in mode 2 with $v_{tar} = v_0$.

This same VUT algorithm will be the subject VUT of the experiments in the following sections.

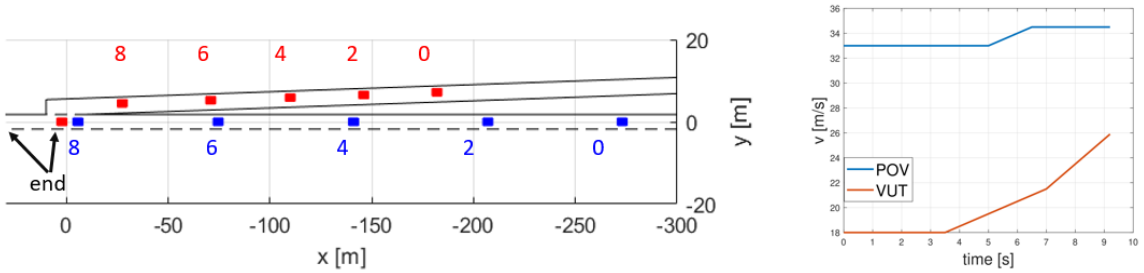
3.8.3 Example Interactive Test Cases

Highway Merging Scenario

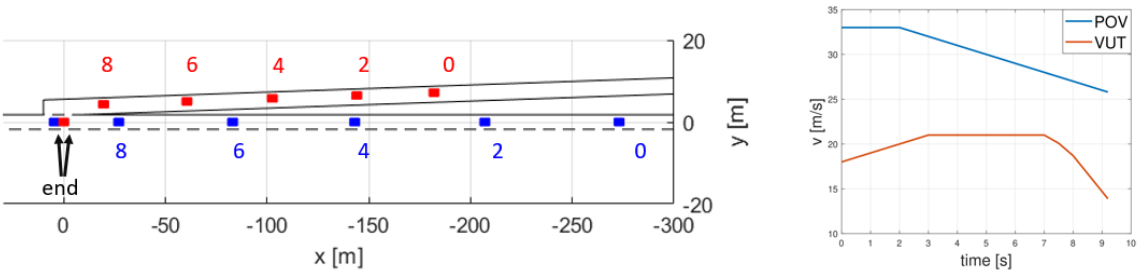
We first present several exemplar test cases with different interactions between the POV and the VUT in the highway merging scenario. The road geometry of the highway merging scenario is based on an entrance ramp on US 23 North near exit 41. The VUT started at $x^0_{VUT} = -182$[m]. In Figure 3.10, we present three test cases with different POVs and VUTs. In all cases, the initial conditions are the same. In the 1st case, as shown in Figure 3.10a, the level-0 VUT accelerates non-cooperatively, while the POV yields by reducing its speed to let the VUT merge first. In the 2nd scenario (Figure 3.10b), the level-1 VUT yields by starting its accelerating phase later, while the level-2 POV with a cooperative SVO accelerates to leave more room for the VUT to merge behind. In the 3rd scenario, (Figure 3.10c), the same level-2 POV yields to let the VUT enter first. However, the rule-based VUT fails to understand the POV's intention. It starts to accelerate,



(a) Level-1 POV & Level-0 VUT



(b) Level-2 POV & Level-1 VUT; $\psi = 0.60$



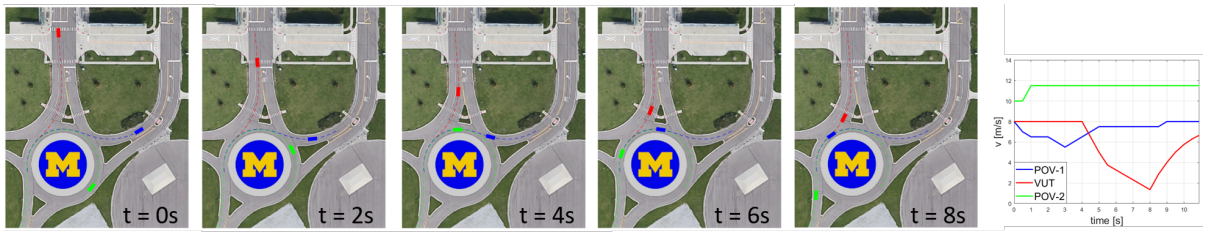
(c) Level-2 POV & rule-based VUT; $\psi = 0.60$

Figure 3.10: Highway merging test cases with initial condition: x^0_{POV} = -273 m, v^0_{POV} = 33 m/s, v^0_{VUT} = 18 m/s; blue for VUT, red for POV; the numbers show time lapses in seconds.

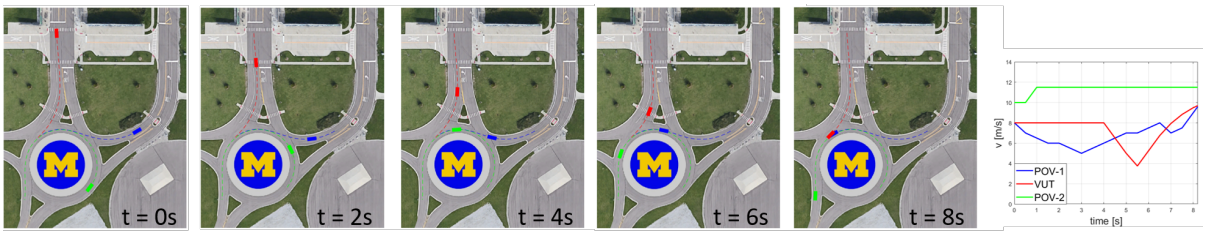
then coasts and even decelerates heavily before it crashes with the VUT. This last case shows a "stalemate" situation, when both agents try to yield to each other and create an inefficient and dangerous scene. These three test cases capture different interactions, which makes the evaluation scenarios diverse and rich.

Roundabout Entering Scenario

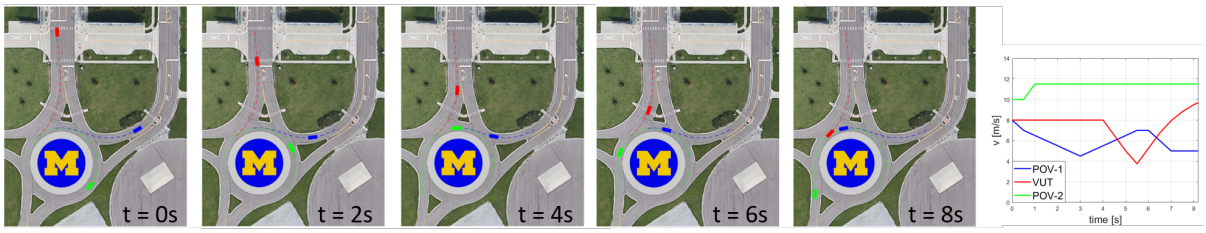
In the roundabout scenario, the diversity of interaction patterns is reflected by the different passing orders of the involved vehicle, i.e. the behavior modes. The order is jointly determined by the timing of entering the roundabout and the yielding/passing decision for each pair of vehicles during their right-of-way negotiations. In this section, we present a variety of simulated test cases



(a) 1st case: POV #1 with $\psi = 0$. The final order: POV #2, POV #1, VUT.



(b) 2nd case: POV #1 with $\psi = 0.2\pi$. The final order: POV #2, VUT, POV #1 (crash).



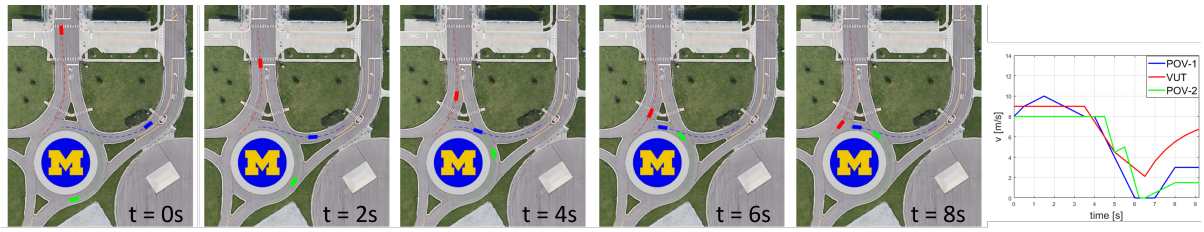
(c) 3rd case: POV #1 with $\psi = 0.35\pi$. The final order: POV #2, VUT, POV #1.

Figure 3.11: Roundabout entering test cases with initial condition: Level-2 POV #1, Level-1 POV #2, x^0_{POV1} = -30.5 m, x^0_{POV2} = -4.5 m, v^0_{POV1} = 8.0 m/s, v^0_{POV2} = 10.0 m/s, v^0_{VUT} = 8.0 m/s; blue for POV #1, green for POV #2, red for VUT.

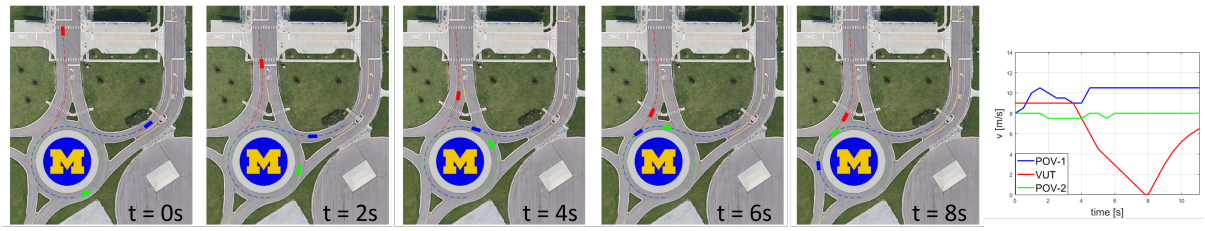
generated by the proposed POV library that demonstrates such diversity.

In Figure 3.11, we present three test cases with the same initial condition but with different POV properties. In the 1st case, as shown in Figure 3.11a, the level-1 POV #2 accelerates to pass POV #1, then passes the VUT. The POV #1 first decelerates to yield POV #2, then keeps a steady speed to pass ahead of the VUT. The VUT slows down to yield to both vehicles, and enters the roundabout behind POV #1. The final order is: POV #2, POV #1, VUT.

In the 2nd case (Figure 3.11b), all the initial conditions and attributes are the same as the 1st case, except that the SVO of the level-2 POV #1 increases from 0 to 0.2π, i.e. POV #1 is more cooperative. Major difference starts when the VUT interacts with POV #1 after $t = 2s$: the VUT switches from "yield to both" to "slip in", while POV #1 still chooses to drive ahead of the VUT. Then a collision happens between POV #1 and VUT. A slight change in the test parameters incurs completely different interactive behaviors and reveals a failure mode of this VUT.



(a) 4th case: $x^0_{POV2} = -18.0$ m. The final order: VUT, POV #1, POV #2.



(b) 5th case: $x^0_{POV2} = -12.0$ m. The final order: POV #1, POV #2, VUT.

Figure 3.12: Testing cases with parameters: Level-1 POV #1, Level-2 POV #2, $x^0_{POV1} = -34.0m$, , $v^0_{POV1} = 8.0m/s$, $v^0_{POV2} = 8.0m/s$, $v^0_{VUT} = 9.0m/s$; blue for POV #1, green for POV #2, red for VUT.

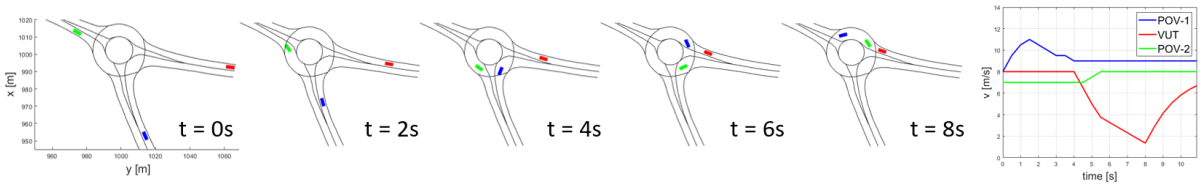
In the 3rd case (Figure 3.11c), all are unchanged except the SVO of POV #1 increases to 0.35π. Although the VUT still attempts to slip in ahead of the POV #1, the POV #1 chooses to yield and there is no collision this time. The final order is: POV #2, VUT, POV #1.

In Figure 3.12, we present two test cases with different initial conditions. In the 4th case (Figure 3.12a), the level-1 POV #1 first passes the POV #2, then yields to the approaching VUT. The VUT first reduces speed, and then accelerates to pass first after observing the yielding of POV #1.

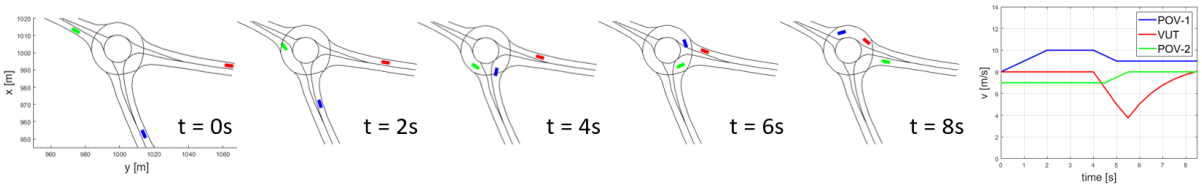
In the 5th case (Figure 3.12b), the initial position of the POV #2 is closer than in the previous case. The POV #1 first accelerates harder to pass the POV #2, then it accelerates again to pass the VUT. The POV #2 keeps a steady speed and also passes the VUT. The VUT stops before the roundabout to yield both POVs before it proceeds. By altering only the initial conditions, we also generate different interaction patterns and passing orders of vehicles.

In addition, to demonstrate that the proposed POV library is scalable to different roundabout layouts, we present two additional test cases at another 3-branch roundabout from the INTERACTION dataset [148] in Figure 3.13. Each POV follows a reference path defined along the lane center, while following the same driving policy computed in Section 3.6 to plan the longitudinal speed. The VUT follows the same rule-based algorithm.

In the 6th case (Figure 3.13a), all vehicles enter the roundabout from different branches and exit at the same branch (top left). The POV #1 first accelerates to pass the POV #2, then keeps constant speed to pass the VUT; the POV #2 accelerates to pass the VUT; the VUT yields to both POVs.



(a) 6th case: both POVs exit at the same branch; Level-1 POV #1, Level-2 POV #2 with $\psi = 0$. The final order: POV #1, POV #2, VUT.



(b) 7th case: POV #2 exit one branch earlier; Level-2 POV #1 with $\psi = 0.2\pi$, Level-1 POV #2. The final order: POV #1, VUT.

Figure 3.13: Testing cases in a different roundabout site, with parameters: x^0_{POV1} = -42.0 m, x^0_{POV2} = -25.0 m, v^0_{POV1} = 8.0 m/s, v^0_{POV2} = 7.0 m/s, v^0_{VUT} = 8.0 m/s; blue for POV #1, green for POV #2, red for VUT.

The final order is: POV #1, POV #2, VUT.

In the last case, (Figure 3.13b), the initial states for all vehicles are the same as the 4th case, while the POV #2 exits the roundabout one intersection earlier, thus there is no potential conflict between POV #2 and VUT. The POV #1 first accelerates to pass the POV #2, then passes the VUT; The POV #2 yields to the POV #1, then exit the roundabout; the VUT yields to only POV #1. The final order is: POV #1, VUT.

The presented seven cases show that we can generate different interaction patterns, and thus challenge the VUT in different ways. Moreover, the approach can be extended to roundabout scenarios with different road geometry and route configurations. Therefore, the richness and flexibility of the proposed testing space are demonstrated.

3.8.4 Results of the Interaction-aware Testing

Then, we present the results of interactive-aware testing for a baseline VUT in MATLAB simulations for the roundabout entering scenario. The VUT follows the rule-based algorithm presented in 3.8.2. A test case at the two-POV roundabout scenario is defined by Eq. (3.8) and Eq. (3.9). We fix the initial position for the VUT, $x^0_{VUT} = -60m$ and assume the speed of the VUT v^0_{VUT} is given to us. The POV category c is also fixed in each experiment. Only a subset of the testing space will be the sampling space for each experiment, where each sample is denoted as s^*. For the following experiments, the threshold for a failure case is set to $\lambda = -500$, which indicates that a

collision had happened.

Results for Sampling within One POV Category

We first show the results of testing experiments with a fixed POV category. We compare the proposed GPR-based adaptive sampling method to other test case generation methods, including uniform sampling, simulated annealing [35], and subset simulation [33]. The FMC criterion is computed for all methods to compare their capability of discovering failure modes.

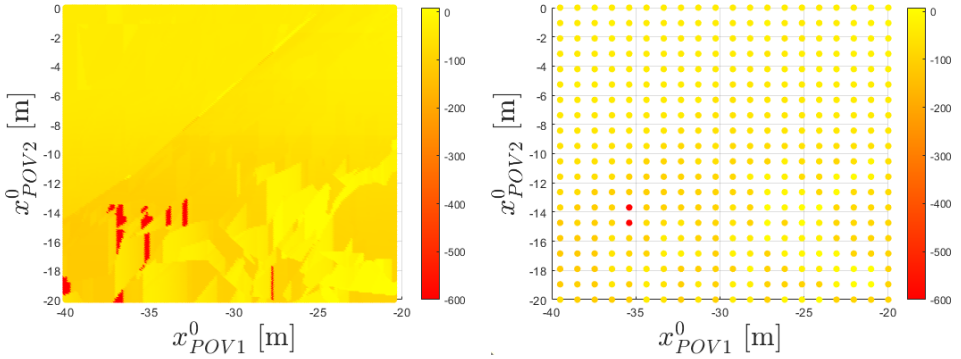
Figure 3.14 shows the testing result comparison when the sampling space is set to 2-dimension. Only the initial positions of both POVs are sampled, i.e. $s^* = [x^0_{POV1}, x^0_{POV2}]$. All methods are compared against the ground truth, which is generated with 40000 samples using uniform sampling. As shown in 3.14a, the ground truth has several disjoint regions with red colors, i.e. the failure modes. For each method from Figure 3.14b to 3.14e, the total number of test cases is $N = 400$. Specifically, for the proposed method, the batch size is $n_b = 20$, and it runs for 20 batches.

Uniform sampling locates one failure region with very few failure cases within 400 cases; simulated annealing identifies many failure cases, but all concentrated around two failure modes at the lower-left corner, which results in low FMC value; subset simulation performs better than simulated annealing, but failure cases are also not distributed in multiple failure modes. The proposed method stands out by identifying failure cases at more diverse failure modes and achieving the highest FMC value. Specifically, failure cases in Figure 3.14e are not too concentrated thanks to the MEI exploitation criterion, which explicitly discourages failure cases that are too close. With only 1% of the sample sizes of the ground truth, the proposed method discovers most of the failure modes qualitatively.

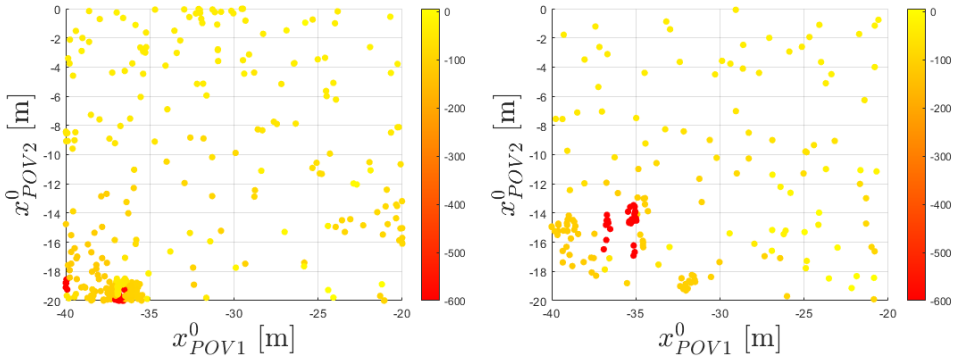
Then, we present the comparison of quantitative results when the sampling space has a higher dimension in table 3.4. Two POV categories are evaluated:

- Category #1: level-0 POV1 and level-1 POV2, with $s^* = [x^0_{POV1}, x^0_{POV2}, v^0_{POV1}, v^0_{POV2}]$ (4-dimension);
- Category #4: level-2 POV1 and level-1 POV2, with $s^* = [x^0_{POV1}, x^0_{POV2}, v^0_{POV1}, v^0_{POV2}, \psi_1]$ (5-dimension).

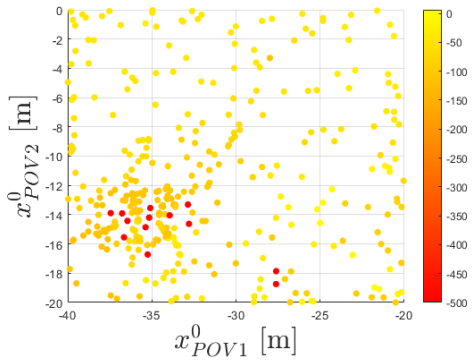
The high dimensionality of the sampling space makes the test case generation results sensitive to the initialization of the proposed and comparison methods. Therefore, we run 100 repeated test runs for each method (except uniform sampling) and compare their average performance. For uniform sampling, to ensure that each dimension is discretized in the same resolution, we allow for more test samples (625 cases for category #1, 1024 for category #4). Each experiment with all



(a) Ground truth: FMC $= 0.1224$ (b) Uniform sampling: FMC $= 0.0128$



(c) Simulated annealing: FMC $= 0.0252$ (d) Subset simulation: FMC $= 0.0421$



(e) Proposed method: FMC $= 0.0659$

Figure 3.14: Testing experiment results with level-1 POV #1 and level-0 POV #2. $v^0_{POV1} = v^0_{POV2} = 9m/s, v^0_{VUT} = 10m/s$. The samples are color coded by the performance score (truncated at 0 and -600), where red means lower score.

Method	$\mathbf{FMC}(\boldsymbol{s}^*, 0.05, -500)$	
	L-0 POV#1, L-1 POV#2	L-2 POV#1,L-1 POV#2
Uniform sampling	0.92×10^{-4}	1.97×10^{-5}
Simulated annealing	0.79×10^{-4}	0.68×10^{-5}
Subset simulation	1.54×10^{-4}	2.21×10^{-5}
Adaptive sampling in [147]	4.07×10^{-4}	3.55×10^{-5}
Proposed w/o BMB	4.38×10^{-4}	3.59×10^{-5}
Proposed method	$\mathbf{4.86 \times 10^{-4}}$	$\mathbf{4.09 \times 10^{-5}}$

Table 3.4: Results comparison for adaptive test case generation in one category.

other methods has 400 test cases. The adaptive sampling method from our previous work [147] is also added for comparison. Even with the advantage on the case number, the uniform sampling method only achieves slightly better results than simulated annealing. Subset simulation performs better than the above two methods, but fall short compared to the three variant of the GPR-based adaptive sampling methods. By modifying the sample selection techniques, the proposed method achieves significant FMC improvement compared to the scheme in [147] in both experiments. Moreover, By adding the step of BMB identification, the proposed method also achieves better results than the variant without it consistently.

Results for Sampling from All Categories

Finally, we simulate the adaptive test case generation procedure with all four POV categories. The goal is to identify more failure cases given a fixed number ($N = 1200$) of cases. Figure 3.15 shows the change of sample allocation across different POV categories, and the ratio of failure cases for each category. The results are presented for both of the proposed sample allocation methods. Here, the sample allocation in batch i is determined by the failure case ratios in batch $(i-1)$ and in earlier batches.

When using the stochastic optimization method, the sample sizes start evenly. Then, since more failure cases have been found within category #1, #3, #4, and none for category #2 POVs, the sample sizes increase for the other three categories while reducing gradually for category #2. This helps to focus on the more promising categories, while maintaining some exploration in the under-performing ones. Finally, we find 271 failure cases in total: 47 failure cases in category #1, 203 in category #3, and 21 in category #4.

When applying the UCB-dynamic method, category #3 is quickly identified as the most promising one, and is then exploited extensively. Finally, we find 518 failure cases in total: 516 in category #3, 2 in category #4. More failure cases are found compared to the previous method, while the diversity of samples is not as good. The two presented methods have their own strengths, which can

be utilized by testers with different goals.

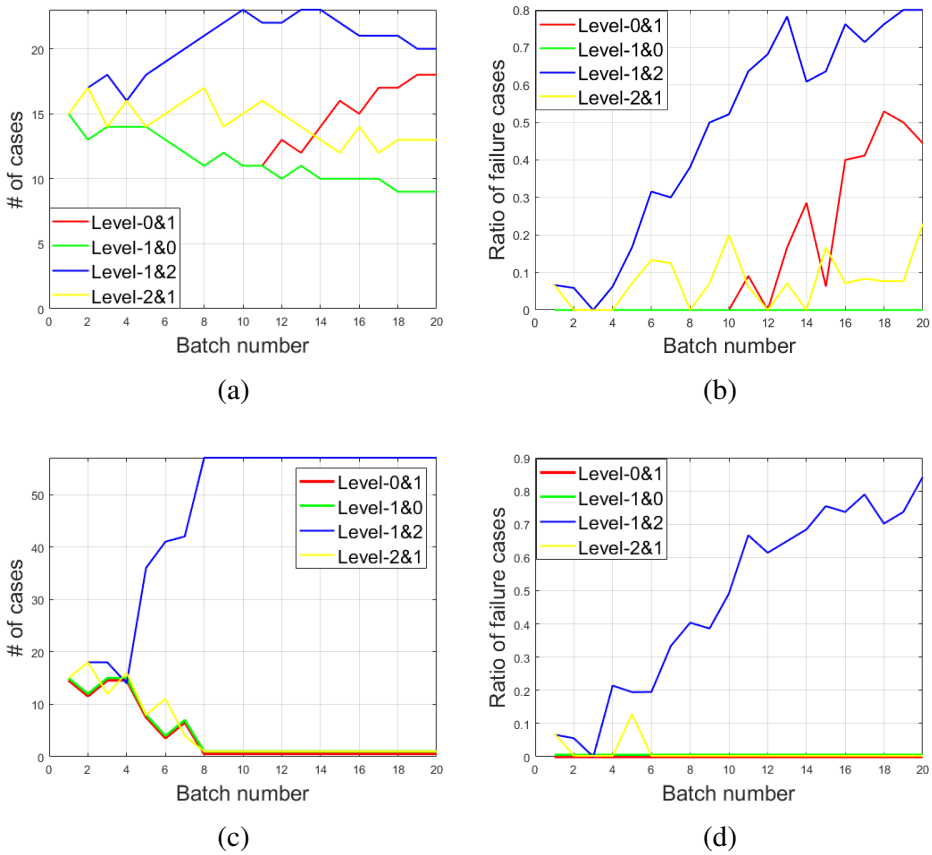


Figure 3.15: Testing experiment results for the roundabout entering scenario, with the full POV library; batch size $n_b = 60$, run for 20 batches. The sample allocation and the failure case ratio across POV categories in all batches are demonstrated for the stochastic optimization method in (a), (b); for the UCB-dynamic method in (c),(d).

3.9 Summary

In this chapter, we develop a framework for evaluating black-box HAVs in interactive scenarios. We combine two game-theoretic formulations, level-k game theory and SVO, and generate a library of interactive POVs using reinforcement learning. The idea is applied to two important scenarios with strong inter-vehicle interactions, highway merging and roundabout entering. For the roundabout-entering scenario specifically, we formulate a realistic two-phase planning algorithm for each POV, making the POV models scalable to the presence of multiple POVs. On the other hand, we design a two-stage adaptive test case generation scheme: a sample allocation procedure

that distributes samples into different scenario categories to ensure case diversity and challengingness; and an adaptive sampling procedure for a single category that aims at failure mode coverage. For sample allocation, we propose two methods based on different objectives. For adaptive sampling, multiple exploration and exploitation criteria are designed and implemented. In addition, We propose the metric FMC to measure the test sample quality. Finally, we demonstrate the usefulness of the proposed framework by running testing in simulation with a rule-based VUT. Our POV library is able to capture a wide variety of interactive behaviors for the two above scenarios, and it can be extended to different locations with different road configurations. The test case generation scheme can customize test samples to discover the failure modes of the VUT efficiently. It outperforms other sampling methods according to the FMC metric. The primary use case of this framework is the acceptance testing. However, HAV companies can also utilize it in their development tests to explore the weaknesses of their algorithms in interactive scenarios. For future work, we plan to extend the current framework to other interactive scenarios including crowded parking lots, unsignalized intersections, etc.

CHAPTER 4

Interaction-aware Corner Case Generation

4.1 Motivation and Background

Corner case testing has been a crucial part of the evaluation paradigm. According to [8], corner cases refer to rare conditions that challenge the capabilities of the VUT while still being within its capability. The term corner cases are used interchangeably with the notion of "edge cases" [149]. They aim to assess the VUT's competence under the worst-case situations. Most existing methods focus on challenging the perception, motion planning or the control capability of the VUT, using situations including adversarial lighting conditions [50], aggressive PORU behaviors [54,81], near-crash initial conditions [58, 59], etc. However, for interactive scenarios, another aspect that should be examined is VUT's interaction capability, namely to recognize the intent of the PORU and make reasonable behavioral decisions. For example, in a highway merging scenario with one human-driven vehicle on the main road and one VUT on the entrance ramp, an inexperienced human driver might not be able to make up his/her mind to pass ahead or yield behind, and may behave in an indecisive or ambiguous way. Another example is a staggered or distracted pedestrian walking near an intersection. A capable HAV should be able to handle these situations. Hence, it is important that we generate corner test cases to cover the possible interactions between the HAV and human drivers/road users in the real world. Despite its significance, this problem has not been discussed in the existing literature to our knowledge.

This chapter presents a novel method for corner case generation for interactive scenarios. In our setting, the desirable corner cases should have the following attributes. First, the PORU should not egregiously try to collide into the VUT like in [81], since interaction modeling requires basic rationality of agents. Second, the corner case should have interpretable difficulty levels to improve the test case generation procedure. Third, the corner cases should be intrinsically difficult, such that they are generalizable to different VUTs. Therefore, our key idea is to generate adversarial PORU behaviors that are ambiguous with intentions. These ambiguous behaviors will form at least a major type of corner case for interactive scenarios.

4.1.1 Related Work

Types of Corner Case Evaluation for HAVs

Corner evaluation refers to testing methods that focus on the capability limit of the VUT. As reviewed in Section 1.2.2, many notions of corner cases exist, which correspond to different evaluation schemes. Adversarial falsification methods (e.g. [35, 64]) view corner cases as the failure modes of a given VUT, which are not generalizable or transferable to other VUTs. The failure cases generated in Chapter 3 can be seen as corner cases from this perspective. Several other studies focus on generating test cases that are universally difficult regardless of the VUT, with objective metrics for test difficulty. The difficulty is characterized by the size of the solution space for a VUT [57, 58], by the distance to the boundary of the control invariant set of the VUT [59], or by the product of exposure frequency and the failure probability [34]. However, the above ideas are not suitable for interactive scenarios, since the difficulty metrics are mainly based on the physical limits of the VUT. In [150], corner cases are defined by the lack of predictability of surrounding objects, quantified by the error of a nominal motion prediction model. Though their focus was on the perception task of HAVs, their intuition matches ours for the interaction-aware evaluation task.

Interactive and Hierarchical PORU Modeling

Modeling interactive human driver/pedestrian behaviors has been extensively studied due to its significance to HAV prediction and decision-making. Many game-theoretic PORU modeling methods have been reviewed in Section 3.3. On the other hand, hierarchical decision-making models assume the observed human actions are governed by their high-level latent states, either directly [151] or indirectly through different utility functions [152]. The latent states can be their goals [153], intents [154, 155], personalities [156] or other behavioral properties. To achieve accurate prediction and successful interaction with human agents, the robots need to effectively estimate the latent states of human agents using online observation. In [157], a planning framework was developed for an indoor robot that navigates around people, which estimates human intents and communicates its intent both implicitly (with motion) and explicitly. In [154, 156], Shannon entropy was used to categorize the expected information gain of the robot action with respect to the human agent's latent states, thus guiding the robot to take information-gathering actions to reduce the uncertainty in human behavior prediction. This idea has inspired our work: for evaluation purposes, we could approach the interactive prediction problem from an opposite direction, i.e, to reduce the amount of information revealed in the action of the PORU, which then confuses the VUT.

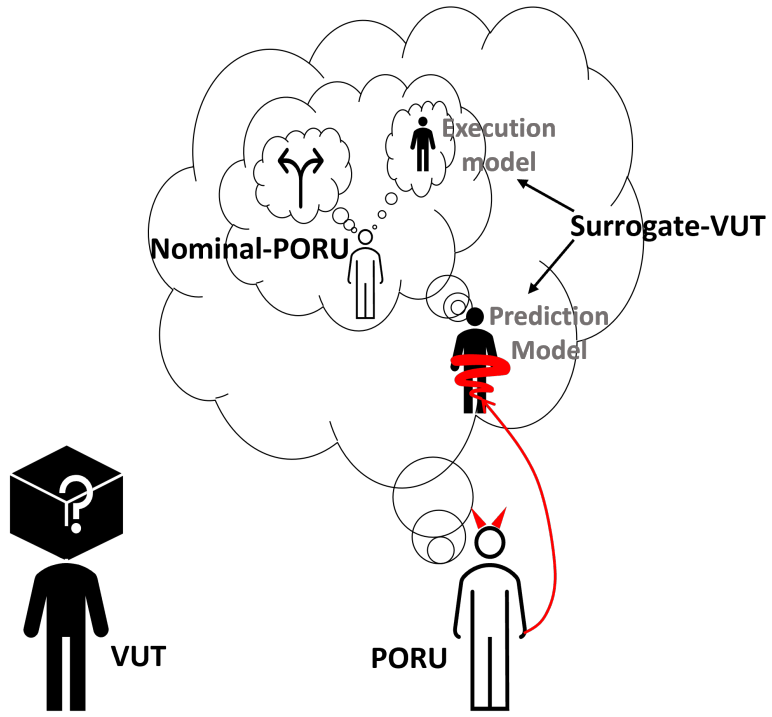


Figure 4.1: The core concept of corner case generation for interactive scenarios: to create PORU behaviors that confuse the surrogate-VUT prediction model with the intention of the PORU.

4.1.2 Contributions

In this chapter, we propose a corner case generation method for HAV evaluation in interactive scenarios. The key contributions are as follows:

- We proposed a novel definition of corner cases for interactive scenarios based on the ambiguity of the PORU, which enables the process to be generalizable (i.e., not targeting a specific VUT).
- We proposed an MPC-based behavior planning method for the adversarial PORU that actively generates ambiguous behaviors.
- We implemented the PORU algorithm in two interactive scenarios, highway merging and pedestrian crossing.
- We proposed a corner case testing framework. We demonstrated the efficacy of the framework through simulation tests for several off-the-shelf VUT algorithms.

4.1.3 Organization

The rest of the chapter is organized as follows: Section 4.2 describes the basic formulations of the interactive scenarios, and presents the definition of ambiguity-based corner cases. Section 4.3 presents the MPC formulation for both nominal and adversarial settings, and describes the method for computing the ambiguity. Section 4.4 presents the implementation of the proposed PORU algorithm in two interactive scenarios, highway merging and pedestrian crossing. Section 4.5 presents the simulation results. Finally, Section 4.6 summarizes the chapter and discusses future research directions.

4.2 Problem Formulation

In this chapter, we study how to generate corner cases in interactive scenarios. we assume that there is one VUT and one PORU. The VUT is a black box, namely its algorithmic details are fully unknown to the PORU, and only the actions and states are observable. The overall idea is illustrated in Figure 4.1. As a starting point, we propose a nominal PORU, denoted as nominal-PORU, which has a known set of intentions, and moves safely and efficiently interacting with the VUT. Then, we assume the existence of a surrogate model for the VUT that governs its interaction with the PORU, denoted as surrogate-VUT. The nominal-PORU assumes that the surrogate-VUT follows a known execution model; conversely, the nominal-PORU serves as the mental model of a PORU for the prediction model in the surrogate-VUT. Finally, the final (adversarial) PORU aims to challenge the above surrogate-VUT prediction model. To generate corner cases, the final PORU will systematically deviate its behaviors from the nominal-PORU, so that it is ambiguous with its true intention, and thus makes itself unpredictable by the above prediction model.

4.2.1 Nominal Decision Model for the Primary Other Road User (PORU)

The nominal-PORU is modeled as a rational agent that optimizes a cost function. Moreover, it is assumed to follow a 3-stage hierarchical decision-making procedure:

1. The high-level intention, $\psi_1 \in \Psi_1$. This represents the motion targets of the PORU. E.g., the target maneuvers of a car on the highway (lane following, lane changing, merge out, etc); the target location of a pedestrian near the crosswalk (across the street, along the sidewalk, etc). In our evaluation setting, we assume that the high-level intention is fixed throughout a scenario, i.e. the PORU will not change its mind on "where to go".

2. The low-level intention, $\psi_2 \in \Psi_2$. This represents different passing orders with another traffic agent, i.e. to pass first or yield. The PORU can alter its low-level intention online

according to the cost of each option.

3. The control input $u \in \mathcal{U}$. The action of the PORU depends on its high-level and low-level intentions, and may include input including acceleration, yaw rate, etc.

In summary, the nominal-PORU takes optimal action with respect to a cost function that depends on the high-level and low-level intentions, denoted as $J_n(\psi_1, \psi_2)$.

4.2.2 The Surrogate VUT Model

The model of the surrogate-VUT consists of two parts, the prediction model and the execution model.

The prediction model of the surrogate-VUT predicts the future motion of the nominal-PORU. It knows the intention set Ψ_1 and Ψ_2, and the corresponding cost functions. At each time-step, the surrogate-VUT first estimates the intention of the nominal-PORU, then subsequently predicts its control input.

The execution model of the surrogate-VUT is assumed to follow a known policy given the current state. With this assumption, the problem of solving an interactive driving policy for the nominal-PORU is simplified from a multi-agent game-theoretic problem to a single-agent planning problem. In other words, the nominal-PORU can optimize its policy by assuming full knowledge of the actions of surrogate-PORU execution model.

4.2.3 Definition of Ambiguity

Since the real intentions of the PORU are not observable, the VUT could only estimate the probabilistic distribution of the intention (i.e. belief) according to the observed states and actions. For an intention ψ of the PORU (either high-level or low-level), we define ambiguity as the Shannon entropy [158] of the belief on ψ by the surrogate-VUT in Eq. (4.1).

$$\mathbf{H}(P(\psi)) = \mathbb{E}[-\log(P(\psi))] \tag{4.1}$$

In this way, high ambiguity means the intention of the PORU has higher uncertainty in the predictor's eyes, which leads to unpredictable future actions of the PORU, thus becoming corner cases in the sense of interactive prediction and decision-making. Ambiguity is the key concept for generating adversarial PORU behaviors, and the computation details are illustrated in the next section.

4.3 Solution Method

4.3.1 MPC Formulation for the Nominal-PORU

Model predictive control (MPC) is a popular control algorithm that iteratively solves an optimal control problem for a rolling horizon [102]. We assume that the nominal-PORU plans its future motion following the MPC framework. Given the joint state of the 2-agent-system at time t, denoted as $x(t)$, the MPC for the nominal-PORU can be formulated as the following optimization problem:

$$\begin{aligned} \min_{\mathbf{u}} \quad & J_n(x_0, \mathbf{u}|\psi_1, \psi_2) \\ \text{s.t.} \quad & x_{i+1} = f(x_i, u_i), \forall i \in \{0, 1...N_h - 1\} \\ & x_0 = x(t), \mathbf{u} = \{u_0, u_1...u_{N_h-1}\} \\ & u_i \in \mathcal{U}, x_{i+1} \in \mathcal{X}, \forall i \in \{0, 1...N_h - 1\} \end{aligned} \tag{4.2}$$

Here J_n denotes the cost function of the nominal-PORU. The cost function depends on the chosen intention (ψ_1 and ψ_2). The discrete-time dynamics is described by $x_{i+1} = f(x_i, u_i)$. N_h denotes the time horizon of the MPC. $\mathcal{X} \subset \mathbb{R}^{n_x}$, $\mathcal{U} \subset \mathbb{R}^{n_u}$ are the state and input sets respectively. After solving (4.2), we obtain the optimal open-loop input trajectory $\mathbf{u}^*$. We will only execute the input at the 1st time-step, i.e. u_0^*, then update the dynamics and re-solve the MPC at the next time-step.

4.3.2 Prediction Model of Surrogate-VUT

The prediction model of the surrogate-VUT will estimate the intentions of the nominal-PORU according to its current state and action. The intention estimation procedure is modeled as a Bayesian inference procedure, which is updated at every time-step.

First, an observation model needs to be defined. To reflect the uncertainty of the human decision-making process, it assumes that PORU's action is only "noisily rational" with respect to the cost function. We apply the principle of maximum entropy [159], which has been widely used in the human-robot interaction community. It states that the likelihood of observing an action is exponentially proportional to the estimated utility of that action, as expressed in Eq. (4.3).

$$P(u|x, \psi_1, \psi_2) = \frac{\exp(Q(x, u|\psi_1, \psi_2))}{\Sigma_{\bar{u}\in\mathcal{U}} \exp(Q(x, \bar{u}|\psi_1, \psi_2))} \tag{4.3}$$

where

$$Q(x, u|\psi_1, \psi_2) = -\min_{\mathbf{u}} J_n(x, \mathbf{u}|\psi_1, \psi_2, u_0 = u) \tag{4.4}$$

Here, $Q(x, u|\psi_1, \psi_2)$ is the estimated utility of action u at state x with intention ψ_1, ψ_2, similar to the Q-value function in the Q-learning algorithm [160]. It is derived from the cost function of the

nominal MPC in (4.2). Eq. (4.3) bridges the cost function and the action observation likelihood. Thereby, the surrogate-VUT is able to update its belief on the high-level and low-level intentions of the PORU through continuous observation, as discussed in the following sections.

4.3.3 Low-level Intention Estimation

Assuming the high-level intention of the PORU is known, the low-level intention can be estimated from the Bayesian theorem.

$$P(\psi_2|x,u,\psi_1) \propto P(u|x,\psi_2,\psi_1)P(\psi_2|x,\psi_1) \tag{4.5}$$

In Eq. (4.5), the 1st term of the right-hand side is the action likelihood, as shown in Eq. (4.3); the 2nd term is the prior belief on the PORU's low-level intention. The prior can be obtained from two possible sources:

1. From the posterior belief at the last time-step,

$$P_1(\psi_2|x_t,\psi_1) = P(\psi_2|x_{t-1},u_{t-1},\psi_1) \tag{4.6}$$

2. From the estimated utility of each low-level intention at the current state without observing the action. Following the same idea as in Eq. (4.3), we assume that the prior can be represented as:

$$P_2(\psi_2|x,\psi_1) \propto \max_{u\in U}(\exp(Q(x,u|\psi_1,\psi_2)) \tag{4.7}$$

   It means that the prior probability is exponentially proportional to the best-case utility of that intention at the current state.

The actual prior probability is computed as an affine combination of Eq. (4.6) and Eq. (4.7).

$$P(\psi_2|x,\psi_1) = \mu P_1(\psi_2|x,\psi_1) + (1-\mu)P_2(\psi_2|x,\psi_1) \tag{4.8}$$

Using Eq. (4.5)-(4.8), the VUT updates the posterior estimate on the PORU's low-level intention.

4.3.4 High-level Intention Estimation

Based on the low-level intention, the belief on the high-level intention of the PORU can be updated according to the latest observation.

$$\begin{aligned} P(\psi_1|x,u) &= \sum_{\psi_2 \in \Psi_2} P(\psi_1, \psi_2|x,u) \\ &\propto \sum_{\psi_2 \in \Psi_2} P(u|x,\psi_2,\psi_1) P(\psi_1,\psi_2|x) \\ &= \sum_{\psi_2 \in \Psi_2} P(u|x,\psi_2,\psi_1) P(\psi_2|x,\psi_1) P(\psi_1|x) \end{aligned} \tag{4.9}$$

In the last line, the first two terms are defined in Eq. (4.3) and (4.8), respectively. The 3rd term is the prior belief on the high-level intention. It is computed in a similar way as the prior belief of low-level intention by combining two sources.

$$P(\psi_1|x) = \mu P_1(\psi_1|x) + (1-\mu) P_2(\psi_1|x) \tag{4.10}$$

where

$$\begin{aligned} P_1(\psi_1|x_t) &= P(\psi_1|x_{t-1}, u_{t_1}) \\ P_2(\psi_1|x) &\propto \max_{u \in U, \psi_2 \in \Psi_2} (\exp(Q(x,u|\psi_1,\psi_2)). \end{aligned} \tag{4.11}$$

Here, P_1 is the posterior belief on the high-level intention at the last time-step; P_2 is computed based on the best-case utility of all inputs and all low-level intentions at the current state. Thereby, the VUT can update the belief on the PORU's high-level intention.

4.3.5 Estimation of Q-value Function

To estimate the PORU's intention online, The Q-value function (Eq. (4.4)) needs to be computed at every time-step for every input candidate. The most straightforward way is to solve the MPC with the 1st input fixed as the input candidate. However, this approach will introduce high computation costs. To estimate the Q-value function more efficiently, we propose to approximately compute the Q-value function, which only requires solving the MPC in (4.2) once every time-step. Assume we are at state $\bar{x}$ and want to evaluate the Q-value function of input $\bar{u}$. The procedure is as follows.

1. We solve the MPC in (4.2) to obtain the optimal input sequence $\mathbf{u}^* = \{u_0^*, ...u_{N_h-1}^*\}$ and state trajectory $\mathbf{x}^* = \{x_0^*, ...x_{N_h}^*\}$, where $x_0^* = \bar{x}$.

2. We replace u_0^* with $\bar{u}$. For the rest of the inputs, additional proportional control is used to

track the nominal state trajectory $\mathbf{x}^*$.

$$\begin{cases} \hat{u}_0 & = \bar{u} \\ \hat{u}_i & = u_i^* + K_p(x_i^* - x_i), \quad i = 1, ...N_h - 1 \end{cases} \tag{4.12}$$

3. The cost of the trajectory generated with $\hat{\mathbf{u}}$ is evaluated, and is used to approximate the actual Q-value function.

$$Q(x, \bar{u}|\psi_1, \psi_2) \approx -J_n(x, \hat{\mathbf{u}}|\psi_1, \psi_2) \tag{4.13}$$

where $\hat{\mathbf{u}} = \{\hat{u}_0, ...\hat{u}_{N_h-1}\}$.

4.3.6 Adversarial Interactive PORU Model

With the above formulation as the basis, we finally present the planning algorithm for the adversarial PORU. It shares the same MPC planning pipeline as the nominal-PORU described in (4.2). The only difference is in the cost function, since an adversarial PORU has an extra goal of being ambiguous. To achieve this, the cost function comprises two terms:

$$J_{adv} = J_n + \lambda J_{ambi} \tag{4.14}$$

Here, J_n is the same nominal cost function as in (4.2). J_{ambi} is the ambiguity cost that encourages the PORU to take ambiguous actions. λ is the weight of J_{ambi} in the final cost. λ is named the ambiguity level, since it determines to what extent the PORU focuses on ambiguity-maximizing. When $\lambda = 0$, the PORU plans its motion the same way as the nominal-PORU. When λ increases, the PORU becomes more ambiguous in its actions. J_{ambi} is defined as:

$$J_{ambi}(x, u|\psi_1, \psi_2) = -(\alpha \mathbf{H}(P(\psi_1|x, u)) + (1 - \alpha)\mathbf{H}(P(\psi_2|x, u, \psi_1))). \tag{4.15}$$

Here, J_{ambi} is an affine combination of the ambiguity for the high-level intention (1st term) and the low-level intention (2nd term), where the ambiguity is defined in Eq. (4.1). It means that a PORU action that confuses the surrogate-VUT with both the high-level and low-level intentions is more desirable. To generate adversarial behaviors that serve as corner cases in evaluation, the PORU prefers to take the least informative action regarding its intentions. The overall algorithm is shown in Algorithm 3.

Algorithm 3 Ambiguity-guided PORU behavior generation (one time-step)

Input: Ψ_1, Ψ_2, $\mathcal{X}$, $\mathcal{U}$; current time-step k, state $x(k)$; previous belief on intentions of the surrogate-VUT:
$b(\psi_1|k-1) = P(\psi_1|x(k-1), u(k-1))$
$b(\psi_2|\psi_1, k-1) = P(\psi_2|x(k-1), u(k-1), \psi_1)$.

Output: PORU action $u(k)$; next state $x(k+1)$; current belief on intentions of the surrogate-VUT $b(\psi_1|k)$, $b(\psi_2|\psi_1, k)$.

for each intention $\psi_1 \in \Psi_1$ **do**
  for each intention $\psi_2 \in \Psi_2$ **do**
    Solve the nominal MPC defined in (4.2).
    for each input candidate $u \in \mathcal{U}$ **do**
      Estimate Q-value function $Q(x(k), u|\psi_1, \psi_2)$. (cf. Section 4.3.5).
    end for
    Update posterior distribution for low-level intentions if u applied: $P(\psi_2|x(k), u, \psi_1)$. (cf. Section 4.3.3)
  end for
  Update posterior distribution for high-level intentions if u applied: $P(\psi_1|x(k), u)$. (cf. Section 4.3.4)
end for
Compute ambiguity cost J_{ambi} (cf. Section 4.3.6).
for each intention $\psi_2 \in \Psi_2$ **do**
  Solve the adversarial MPC with cost function defined in Eq. (4.14), obtain first input $u^*_{\psi_2}$ (cf. Section 4.3.6).
end for
Pick the intention ψ_2 with the lowest cost; adopt corresponding input as $u(k)$; $x(k+1) = f(x(k), u(k))$.
Update belief on high-level / low-level intentions: $b(\psi_1|k)$, $b(\psi_2|\psi_1, k)$.

4.4 Implementation for two interactive scenarios

In this section, we implement the above adversarial PORU algorithm in two typical interactive scenarios, highway merging and pedestrian crossing.

4.4.1 Highway Merging Scenario

Scenario Setup

In the highway merging scenario, the PORU is a primary other vehicle (POV), which is driving on the main road with the right-of-way. The VUT attempts to merge onto the highway while keeping a safe distance from the POV. The configuration is the same as Figure 3.3 in Chapter 3. The VUT has a target merge point M, which is the origin of the lane-fixed coordinates for both the ramp and the main road. The scenario ends when the VUT reaches point M, and the end time is denoted as

t_1. The POV has one high-level intention, which is point M. For low-level intention, the POV can either merge ahead or behind the POV, i.e. $\Psi_2 = \{\text{pass}, \text{yield}\}$.

We model both vehicles as double integrators and they only move longitudinally in their lane. The joint state of the two-vehicle system is $[x_{POV}, v_{POV}, x_{VUT}, v_{VUT}]^T$, where x_{POV}, x_{VUT} are the longitudinal positions of POV and VUT, and v_{POV}, v_{VUT} are their longitudinal speeds. The input for each vehicle is the longitudinal acceleration, denoted as a_{POV} and a_{VUT}. $u = a_{POV}$ is solved by the MPC scheme, while the disturbance input a_{VUT} is controlled by the black-box VUT. For the surrogate VUT model, a_{VUT} is computed non-reactively from a known model: it accelerates with constant acceleration (1 m/s^2) until the highway target speed (28 m/s) and then drives at that constant speed. It is the same model as the level-0 policy in Section 3.5.

To avoid the deadlock between the POV and the VUT, the closed-loop planning for the POV will stop when the predicted remaining time of the scenario is less than the MPC time horizon. After that, the POV will follow the latest MPC solution for the rest of the scenario in an open-loop way. Therefore, even in a corner case, the PORU will stop to be adversarial eventually, and the open-loop behavior will leave time for a potent VUT to avoid collisions.

Cost Function Design

The goal of the nominal POV is to leave the VUT enough gap with minimal effort and minimal deviation from its desired speed. Its cost function J_n can be described as follows.

$$J_n = W_{merge}^T [J_{\Delta v}\ J_u\ J_{gap}]. \tag{4.16}$$

Where W_{merge} is the cost weight. The three cost terms are explained below.

1. $J_{\Delta v}$: deviation from the initial speed.

$$J_{\Delta v} = \sum_{i=1}^{N_h} (v_{POV,i} - v_{POV}^0)^2 \tag{4.17}$$

2. J_u: input cost at every stage:

$$J_u = \sum_{i=0}^{N_h - 1} u_i^2 \tag{4.18}$$

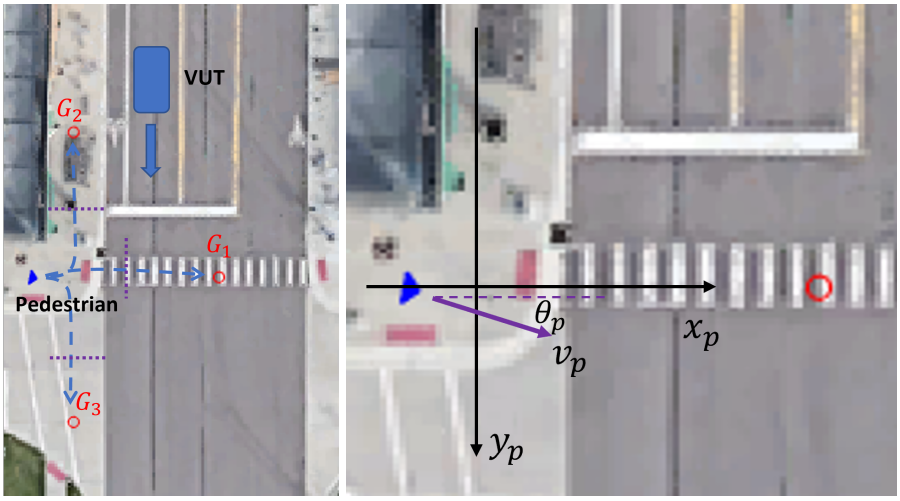
3. J_{gap}: cost on the final gap between the POV and VUT. This term characterizes the predicted distance gap between them when the VUT reaches the merge point. It is dependent on the

low-level intention.

$$J_{gap} = \begin{cases} \frac{1}{1+\exp(\beta(x_{POV}(t_x)))}, & \text{if } \psi_2 = \text{pass} \\ \frac{1}{1+\exp(\beta(-x_{POV}(t_x)))}, & \text{if } \psi_2 = \text{yield} \end{cases} \tag{4.19}$$

t_x represents the estimated time that the VUT arrives at the merge point. If the intention is to pass, J_{gap} is low if the POV is expected to lead the VUT by a large margin at t_x, and vice versa when the intention is to yield.

4.4.2 Pedestrian Crossing Scenario



(a) (b)

Figure 4.2: (a) The configuration of the pedestrian crossing scenario, where G_1, G_2 and G_3 are the three goal locations, the blue arrows are sketches of the nominal trajectory for each goal; the purple lines show the "point of no return" for each goal. (b) Definition of the variables and the frame for the pedestrian scenario.

The 2nd target scenario is the pedestrian crossing scenario. The scenario configuration is shown in Figure 4.2a. The PORU refers to the pedestrian, which is walking towards an unsignalized intersection, for which a VUT is approaching from the top. The pedestrian is assumed to have three possible routes: go up along the sidewalk, go down to cross the side street, and go right to cross the main road. Each route is represented by a target location, which form the high-level intention set of the pedestrian, i.e. $\Psi_1 = \{G_1, G_2, G_3\}$. Since only the goal G_1 has potential traffic conflict with the VUT, the low-level intention set of G_1 is $\Psi_2(\psi_1 = G_1) = \{\text{pass}, \text{yield}\}$. For target locations G_2 and G_3, the low-level intention is irrelevant. Each scenario will have a fixed duration, which is set to 8s in the following discussion.

Scenario Model

The dynamics of the pedestrian can be represented as a nonlinear state-space model, as shown in Eq. (4.20).

$$\begin{cases} x_p(k+1) & = x_p(k) + v_p(k)\cos\theta_p(k)\Delta t \\ y_p(k+1) & = y_p(k) + v_p(k)\sin\theta_p(k)\Delta t \\ v_p(k+1) & = v_p(k) + a_p(k)\Delta t \\ \theta_p(k+1) & = \theta_p(k) + \omega_p(k)\Delta t \end{cases} \tag{4.20}$$

The model has four states: $x = [x_p, y_p, v_p, \theta_p]^T$. x_p and y_p are the coordinates of the pedestrian in a global Cartesian frame, v_p is its speed, θ_p is its heading angle. The model has two control inputs: $u = [a_p, w_p]^T$, where a_p is the acceleration, w_p is the yaw rate. The definition of the state variables is shown in Figure 4.2b. The surrogate-VUT is assumed to drive at a constant speed.

During the evolution of the scenario, the ambiguous behavior of the pedestrian will last until it reaches the "point of no return" for each intention, which is defined as thresholds on its x / y location. After that, the pedestrian will follow the current intentions with $\lambda = 0$. In addition, similar to the highway merging scenario above, the closed-loop planning for the pedestrian will be replaced with open-loop planning when the predicted remaining time of the scenario is less than the MPC time horizon.

Cost Function Design

The goal of the nominal pedestrian is to reach its target location in a fast and safe manner. Its cost function J_n can be described as follows.

$$J_n = W_{ped}^T [J_{\Delta x}\ J_u\ J_{gap}\ J_{corridor}]. \tag{4.21}$$

Where W_{ped} is the cost weight. The four cost terms are explained below.

1. $J_{\Delta x}$: deviation from the goal state.

$$J_{\Delta x} = \sum_{i=1}^{N_h} \Delta x_i^T Q \Delta x_i \tag{4.22}$$

   where $\Delta x_i = x_i - x_{goal}$, x_{goal} is predefined for each goal location in Ψ.

2. J_u: input cost at every stage:

$$J_u = \sum_{i=0}^{N_h - 1} u_i^T R u_i \tag{4.23}$$

3. J_{gap}: cost on the gap between the pedestrian and VUT. This term characterizes the time gap between them regarding the predicted moment that each of them reaches the crosswalk. It is only active of $\psi_1 = G_1$. It is dependent on the low-level intention.

$$J_{gap} = \begin{cases} \frac{1}{1+\exp(\beta(TTA_{VUT} - TTA_{Ped}))}, \text{ if } \psi_2 = \text{pass} \\ \frac{1}{1+\exp(\beta(TTA_{Ped} - TTA_{VUT}))}, \text{ if } \psi_2 = \text{yield} \end{cases} \tag{4.24}$$

TTA represents the estimated time-to-arrival to the crosswalk for the PORU or the VUT, according to the current input trajectory. If the choice is to pass, J_{gap} is low when the pedestrian is expected to enter the crosswalk much sooner than the VUT, and vice versa when the choice is to yield.

4. $J_{corridor}$: a soft constraint that avoid the pedestrian to walk outside a valid corridor defined for the chosen high-level intention.

4.5 Simulation Results

The simulation results include both qualitative case studies and quantitative analysis. The PORU algorithm and the simulation environment for both scenarios are implemented in MATLAB. The nonlinear MPC is set up and solved with MATLAB model predictive control toolbox [161]. The simulation parameters are listed in Table 4.1.

Highway merging	
Simulation time-step	0.1s
MPC time-step	0.5s
N_h	5
a_{POV} range	$[-4, 3]$ m/s^2
Pedestrian crossing	
Simulation time-step	0.1s
MPC time-step	0.4s
N_h	6
a_p range	$[-2, 2]$ m/s^2
ω_p range	$[-\pi/2, \pi/2]$ rad/s

Table 4.1: Simulation parameters for corner case generation.

4.5.1 VUT Algorithms

The VUT algorithms used in the simulations are described below. For both scenarios, the VUT only moves in the longitudinal direction. The input is the acceleration, which is bounded to $[-4, 2]$ m/s^2. The VUT algorithms we tested are not meant to be flawless; the goal is to demonstrate our capability of generating corner cases against different VUTs.

Highway Merging Scenario

We tested two state-of-the-art VUT algorithms. The first algorithm is based on the deterministic sampling method from [162]. The VUT plans its motion in a receding-horizon fashion at every time-step: it first samples the target state in a 5-second horizon from 208 target candidates, then connects the current state with the targets using quintic polynomials to form jerk-optimal trajectory candidates [137]. Next, trajectories are screened with constraints, and the best trajectory is picked based on a cost function. The second algorithm is the Generalized Intelligent Driver Model (GIDM) [163], which is an extension of the widely-used Intelligent Driver Model (IDM) [112] and was specifically developed for the highway merging scenario. It projects the main-lane vehicle (POV) onto the ramp to determine the passing order.

Pedestrian Crossing Scenario

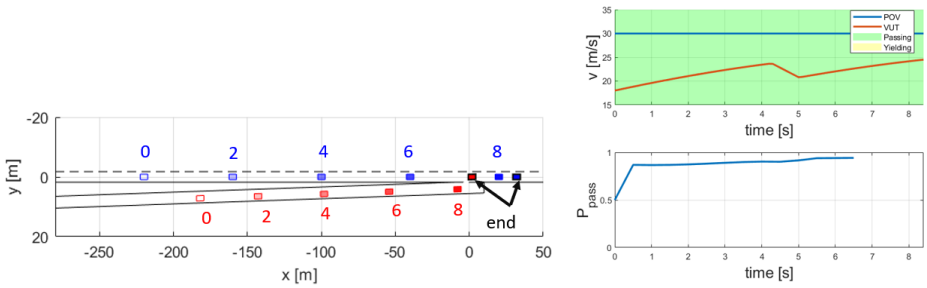
We tested VUT algorithms that are based on the IDM model [112]. The IDM is modified to interact with oncoming pedestrians. Specifically, the pedestrian is treated as the target "preceding vehicle" and the VUT will try to yield to the pedestrian if the predicted arrival time to the conflict region (time-to-arrival) of the pedestrian is earlier than the VUT's. Otherwise, the VUT will ignore the pedestrian and keep its target speed. We tested two variants of this algorithm. The first one computes the time-to-arrival of the pedestrian assuming that it keeps current acceleration until it stops or reaches the max walking speed. It is denoted as "CA-VUT". The second one predicts the time-to-arrival assuming a constant-walking-speed pedestrian, denoted as "CV-VUT".

4.5.2 Adversarial Test Cases

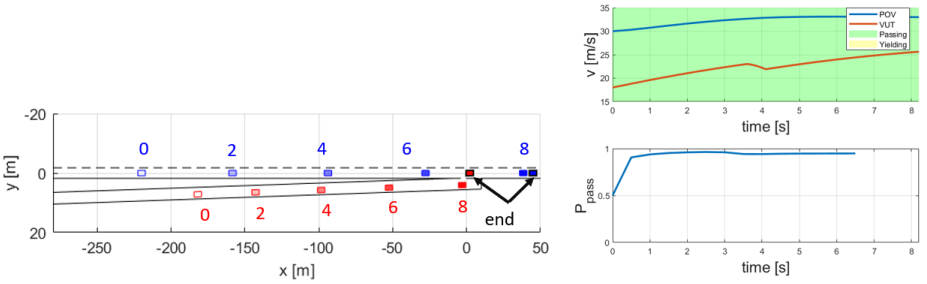
We present selected resulting test cases for the target scenarios to demonstrate the effectiveness of generating interaction-aware corner cases using the proposed ambiguity-guided PORU planner.

Highway Merging Scenario

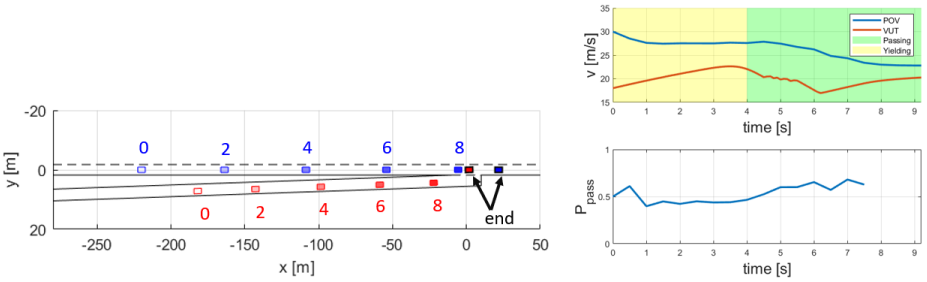
For the highway merging scenario, the road geometry of the simulation is based on an entrance ramp on the US 23 North near exit 41 in Ann Arbor, Michigan, US. We present two groups of test



(a) POV is non-interactive, VUT with GIDM



(b) Proposed POV with $\lambda = 0$, VUT with GIDM

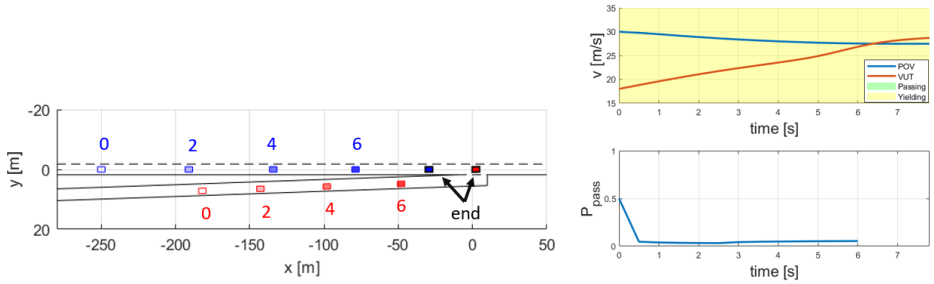


(c) Proposed POV with $\lambda = 400$, VUT with GIDM

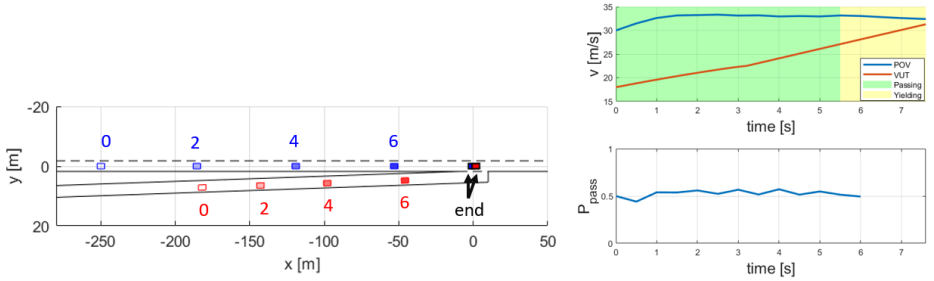
Figure 4.3: The 1st case of highway merging scenario: $x^0_{POV} = -220$m, $v^0_{POV} = 30$m/s, $x^0_{VUT} = -182$m, $v^0_{POV} = 18$m/s.

cases, each with the same initial condition, but different combinations of the POV and VUT.

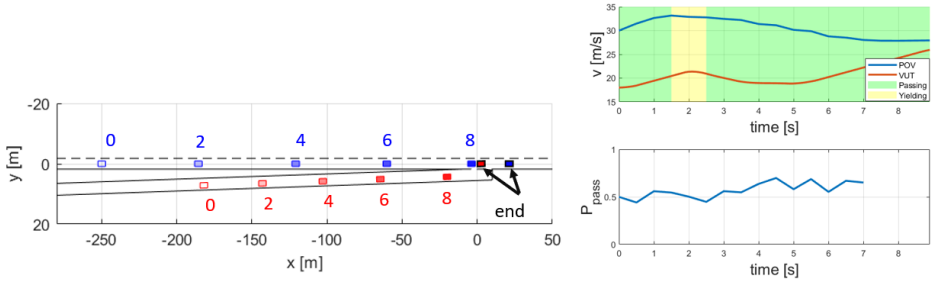
The first group of cases is shown in Figure 4.3, where the VUT is controlled by the GIDM algorithm. The position of the POV (blue) and the VUT (red) are shown on the left figure; each top right figure shows the speed trajectories of both vehicles, with the background color showing the pass/yield decision of the POV; each bottom right figure shows the evolution of the belief on low-level intention $b(\psi_2 = \text{pass}|\psi_1, k)$, denoted here as P_{pass}. The closer P_{pass} is to 0.5, the more ambiguous POV appears. We first show the baseline case of running a non-interactive POV in Figure 4.3a. As the POV kept a constant speed, its predicted intention converged to "pass" very early, making the case unambiguous and easy to deal with. The VUT resolved the situation with a brief stage of deceleration to let the POV pass.



(a) $\lambda = 0$, VUT with GIDM



(b) $\lambda = 400$, VUT with GIDM



(c) $\lambda = 400$, VUT with Deterministic Sampling

Figure 4.4: The 2nd case of highway merging scenario: $x^0_{POV} = -250$m, $v^0_{POV} = 30$m/s, $x^0_{VUT} = -182$m, $v^0_{POV} = 18$m/s.

In the second case, as shown in Figure 4.3b, The nominal MPC-based POV decided to pass all along, and it accelerated to create more room for the VUT to merge from behind. Therefore, the VUT was required to decelerate less. Moreover, the estimated intention also converged to 'pass' quickly.

In the third case (Figure 4.3c), The POV is adversarial with $\lambda = 400$, so its behavior is ambiguous. Though the POV decided to pass at 4s, it decelerated and reduced the gap to the VUT behind, which pushed the VUT to decelerate more. In this case, the VUT was able to merge onto the highway without collision but the final gap was smaller than in the two previous cases. The adversarial POV achieved this by keeping P_{pass} close to 0.5 until the open-loop stage.

The second group of cases is shown in Figure 4.4, where the starting position of the POV is

shifted 30m behind. In the first case in Figure 4.4a, the nominal POV decided to yield to the VUT by decelerating. The GIDM-based VUT safely merged onto the highway. For this case, P_{pass} approached zero quickly, indicating that the POV signaled its yielding attention clearly.

In the next case in Figure 4.4b, the POV is adversarial with $\lambda = 400$. Though the POV started much behind, the POV chose to pass until 5.5s and it accelerated to catch up with the VUT. The VUT failed to respond to the passing decision of the POV and a collision did occur. The value P_{pass} was kept close to 0.5 for most of the time, representing the ambiguous nature of the POV's behavior. A failure mode of the GIDM-based VUT is identified.

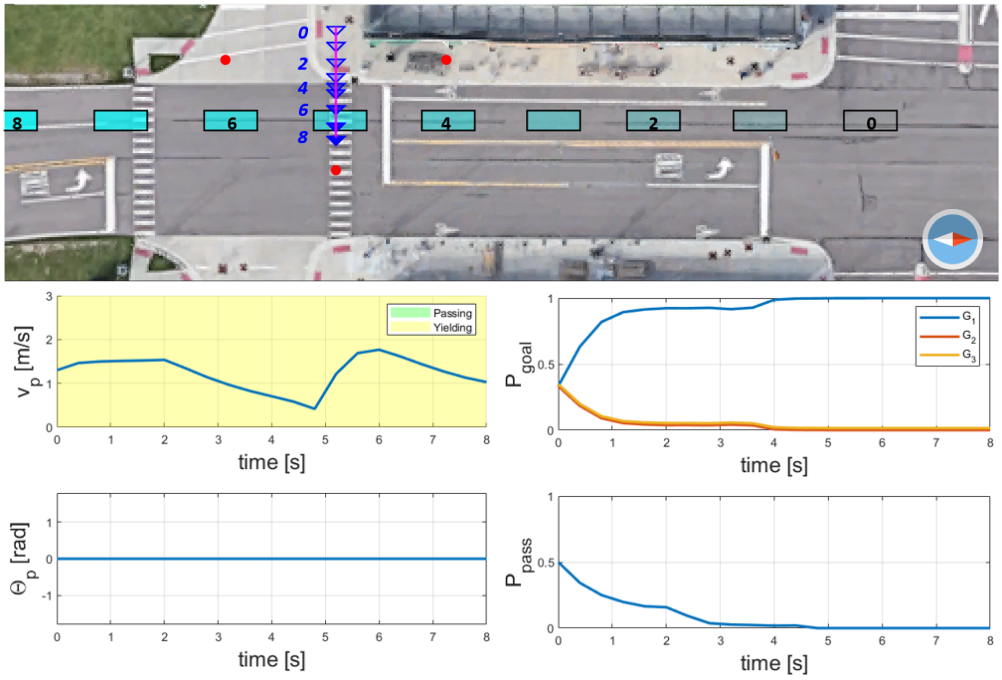
In the last case in Figure 4.4c, the same adversarial POV is put against a VUT controlled by the deterministic sampling algorithm. This VUT was able to respond to the passing intention of the POV promptly and a collision was avoided. The VUT still presented a challenge by keeping the passing probability close to 0.5 the whole time. Specifically, the POV decelerated between 4-7 s even though its intention was to pass, which indicated its adversarial nature.

Pedestrian Crossing Scenario

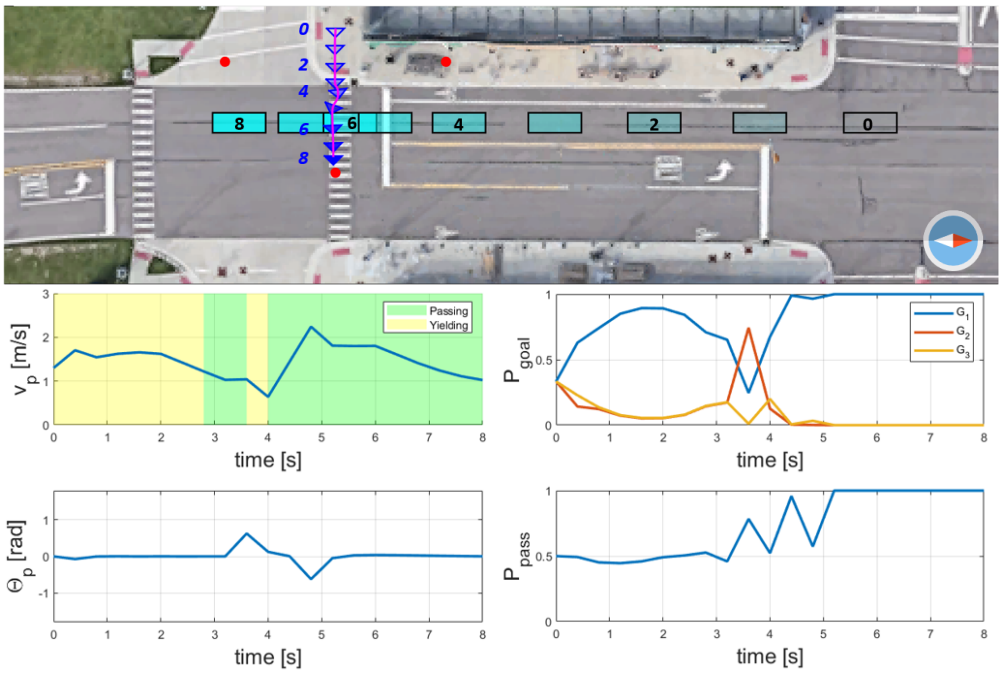
For the pedestrian crossing scenario, the road layout of the simulation is based on a junction inside Mcity. We present three groups of example test cases, each with the same initial conditions. In both scenarios, the VUT comes from the north with an initial velocity $v^0_{VUT} = 10$m/s, controlled by the "CA-VUT" algorithm. The pedestrian has an initial position of $[-3, 0]$, heading east ($\theta_p = 0$). The varying conditions are the initial velocity of the pedestrian and the initial position of the VUT.

The first group of test cases is shown in Figure 4.5. In the figure for each case, the top figure shows the pose of the pedestrian and the VUT at different time-steps. The bottom left figure shows the speed and heading angle of the pedestrian, with the background color showing the pass/yield decision of the pedestrian. The bottom right figure shows the evolution of the belief on high-level intention $b(\psi_1)$ (P_{goal}) and on low-level intention $b(\psi_2 = \text{pass}|\psi_1 = G_1)$ (P_{pass}). In the first case (Figure 4.5a), the nominal pedestrian went directly to the goal G_1, making P_{goal} converge to G_1 quickly. On the other hand, the pedestrian chose to yield to the oncoming VUT, and P_{pass} converged to 0 quickly. This shows a case where the pedestrian has a clear and unambiguous intention and is not challenging to the VUT.

In the second case (Figure 4.5b), The goal location is still G_1, but the pedestrian is adversarial ($\lambda = 400$). The pedestrian had lateral motions between 3-5 s, which induced uncertainty with respect to its goal location, and caused $b(\psi_1 = G_2)$ to increase and $b(\psi_1 = G_1)$ to decrease briefly. On the other hand, the pedestrian shifted from "yielding" to "passing" before 3s, but it kept slowing down and even switched back to "yielding" briefly before committing to pass. This behavior kept P_{pass} close to 0.5 for longer. Thereby, the VUT could not decelerate promptly and collided with the pedestrian.

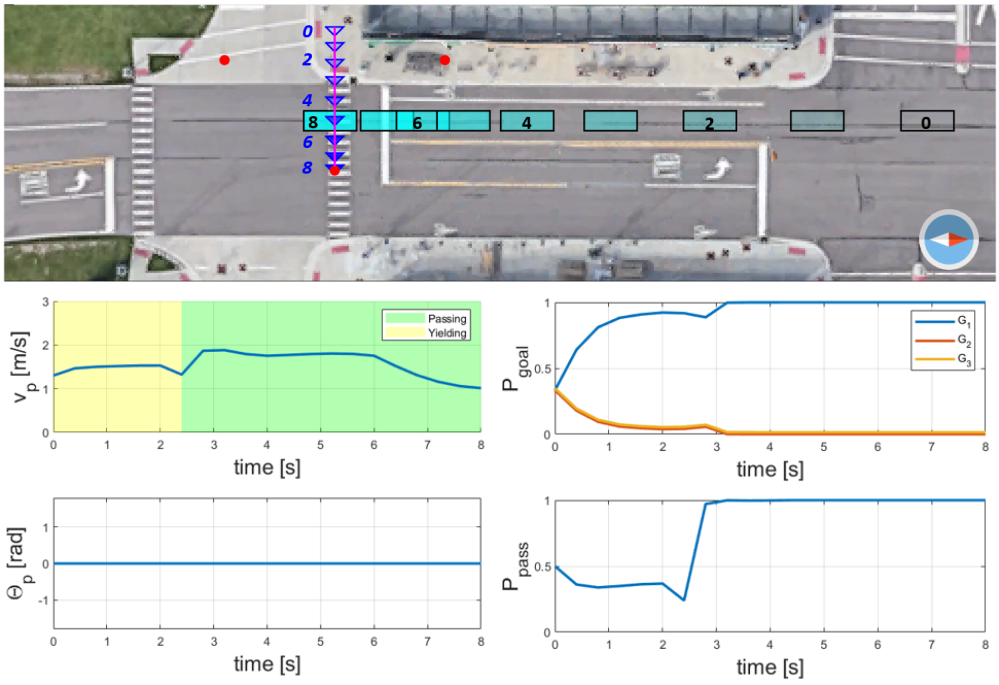


(a) $\lambda = 0$

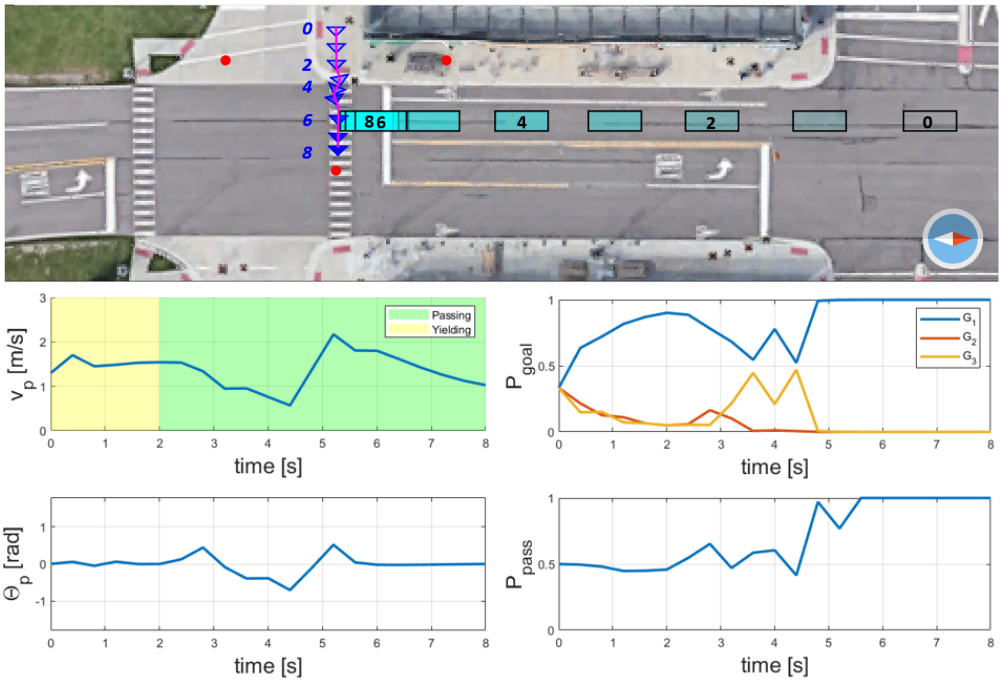


(b) $\lambda = 400$

Figure 4.5: The 1st group of cases of the pedestrian crossing scenario: $x^0_{VUT} = -46.0$m, $v^0_p = 1.3$m/s. The goal is G_1. In the top figure, numbers represent timestamps in seconds; the compass icon shows the orientation.

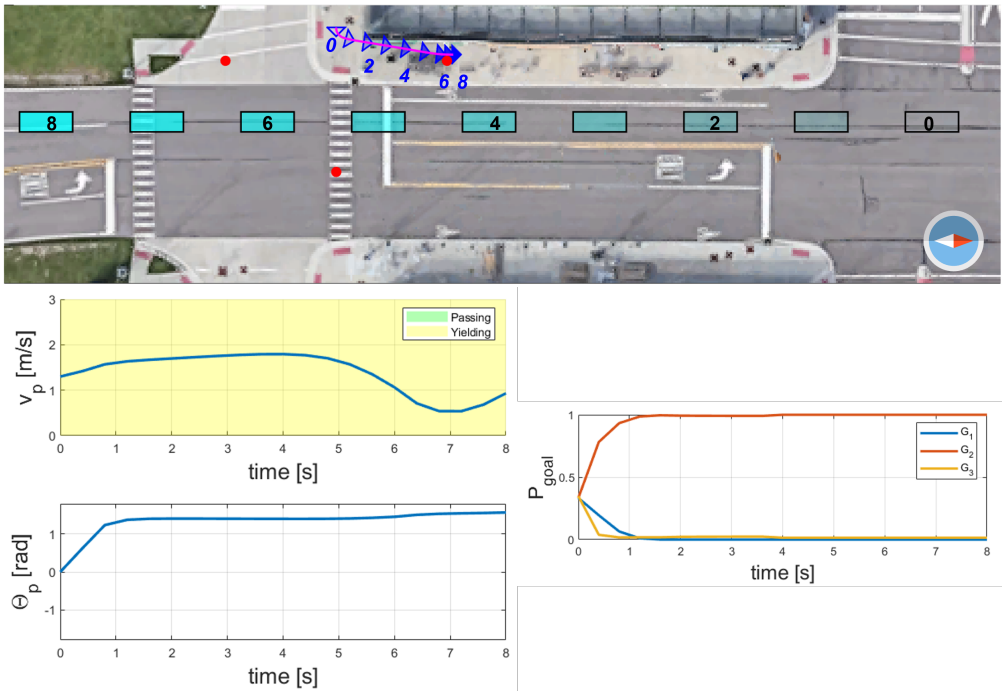


(a) $\lambda = 0$

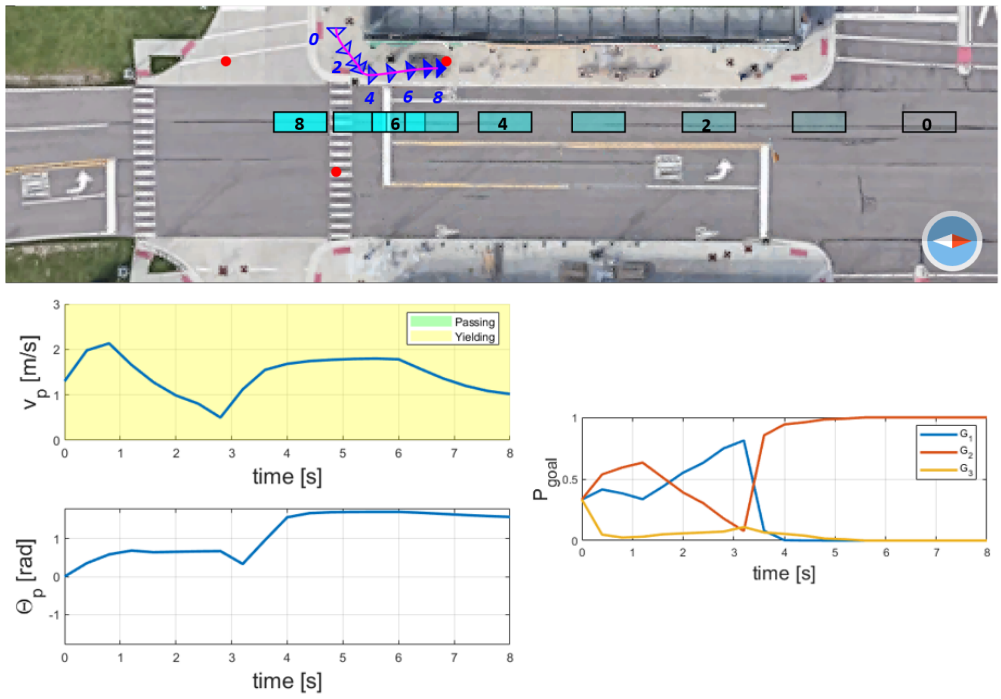


(b) $\lambda = 400$

Figure 4.6: The 2nd group of cases at the pedestrian crossing: $x^0_{VUT} = -51.4$m $v^0_p = 1.3$m/s. The goal is G_1.



(a) $\lambda = 0$



(b) $\lambda = 400$

Figure 4.7: The 3rd group of cases at the pedestrian crossing: $x^0_{VUT} = -51.4$m $v^0_p = 1.3$m/s. The goal is G_2.

In the second group of test cases (Figure 4.6), the initial range of the VUT is farther. In the first case (Figure 4.6a), the non-adversarial pedestrian walked directly to its goal G_1 with no lateral deviation, rendering a fast-converging belief on its high-level intention. For longitudinal motion, it switched from yield to pass at 2.5s and started accelerating immediately. The VUT successfully yielded to the pedestrian. P_{pass} started near 0.5, but converged to 1 soon after the pedestrian decided to pass.

In the second case in Figure 4.6b, the adversarial pedestrian had more lateral motion and kept the uncertainty on P_{goal} high until 4.5s. For the longitudinal motion, though the pedestrian decided to pass at 2.0s, it kept decelerating for more than 2s afterwards, then it accelerated to pass. P_{pass} was kept close to 0.5 for longer than the 1st case. It entered the junction at 4.9s, 1.0s later than the previous case. The VUT reacted late and barely collided with the pedestrian. Therefore, another corner case is identified.

In the third group of cases in Figure 4.7, the goal of the pedestrian is switched to G_2. In Figure 4.7a, the nominal pedestrian walked directly to G_2 with smooth actions. The intention became obvious as early as 1s. In Figure 4.7b, the adversarial pedestrian pretended to aim for G_1 at first. It first walked towards the edge of the sidewalk, and loitered there before it turned north to its actual goal around 4s. Though there was no conflict in the intended path of the pedestrian and the VUT, this behavior confused the VUT about its high-level intention, as shown in the figure of P_{goal}. Subsequently, the VUT braked for the pedestrian. This example showcases a different kind of corner case for the VUT, where both unnecessary timidness and recklessness are undesirable.

In addition, it is worth mentioning the difference between the above test cases with the high-risk test cases in Chapter 2, even though both are designed for challenging the VUT at unsignalized pedestrian crossings. In Chapter 2, the pedestrian is assumed to become observable right when they enter the junction area. Due to the short horizon, the interaction between the VUT and the pedestrian is not considered, thus it is viewed as a reactive scenario. It is only the short-term reflex of the emergency braking on the VUT that is examined. In this chapter, the pedestrian can be observed from much earlier and throughout the build-up of the conflict. Therefore, the corner cases in this chapter are always physically avoidable due to the observability of the pedestrian, but it requires sufficient interaction capability on the VUT to achieve so. Therefore, different aspects of the VUT's capability are addressed in these two chapters.

4.5.3 Implementation of a Corner Case Testing Scheme

In this section, we propose a practical corner case testing scheme based on the aforementioned ambiguity-based PORU planner. The goal is to test the capability limit of the VUT with a fixed number of randomly-generated corner cases.

We propose to generate cases with three difficulty levels: easy, medium and hard. Each difficulty is categorized by a unique ambiguity level value (λ) of the PORU: for easy cases, the interactive PORU has $\lambda = 0$. The medium cases are created with an intermediate λ value and hard cases with a high λ value. The values may need to be tuned in simulations for different driving scenarios.

For each interactive scenario, we first determine the testing space, which consists of key initial conditions of the PORU or the VUT. Then, assuming the number of test cases from each difficulty level is given, we generate uniform samples for each level. The samples are generated using the Halton sequence, a popular quasi-random sequence [164]. It is designed to fill the sampling space with lower discrepancy than sampling with uniform distribution.

To evaluate the performance of the VUT in each test case, we define a performance score similar to Section 3.7. The score is determined by the minimal separation between the PORU and the VUT, and the task accomplishment. A lower value represents poorer performance, and a collision is given an extremely low score (e.g., <-500). For details on the performance score please refer to Section 3.7.

Results for Highway Merging Scenario

We first show the results for the highway merging scenario. We assign medium cases with $\lambda = 85$ and hard cases with $\lambda = 400$. The testing space is composed of two variables: the initial position and speed of the POV: ($[x^0_{POV}, v^0_{POV}]$). The initial conditions of the VUT are held fixed: $x^0_{VUT} = -182$m, $v^0_{POV} = 18$m/s.

We first compare the performance of a GIDM-based VUT under different difficulty levels of PORU. Additionally, we add a non-interactive PORU into comparison, which drives at a constant speed. Each test run includes the same 100 test cases generated by a Halton sequence. We compare the number of poor cases and failure cases for each PORU. Poor cases refer to cases with a performance score below -100, while failure cases with a score below -500 (collision). The results are shown in Table 4.2. Non-adversarial (easy) PORU results in neither poor nor failure cases, proving that they are easy to deal with. Medium PORU creates several poor and failure cases, while the hard PORU creates the most. It is demonstrated that for this VUT, a higher ambiguity level translates to more failures.

Then, for the corner-case testing scheme, we combine the three difficulty levels to form the test library. Assuming 200 cases to be tested, we allocate them as 20 easy cases, 60 medium cases and 120 hard cases, since the hard cases are our focus and easy cases serve as pilot experiments. We test the above two VUTs with the same set of test cases, and the results are shown in Figure 4.8. The VUT with GIDM has 8 collisions, where 7 of them are hard cases and one is a medium case. The VUT based on deterministic sampling has no collision, with 28 poor-performing cases, all are

PORU type	Poor Cases	Failure Cases
Non-interactive	2/100	0/100
None Ambiguous (Easy)	0/100	0/100
Mildly Ambiguous (Medium)	4/100	1/100
Ambiguous (Hard)	32/100	7/100

Table 4.2: Test results comparison for different PORUs for the highway merging scenario.

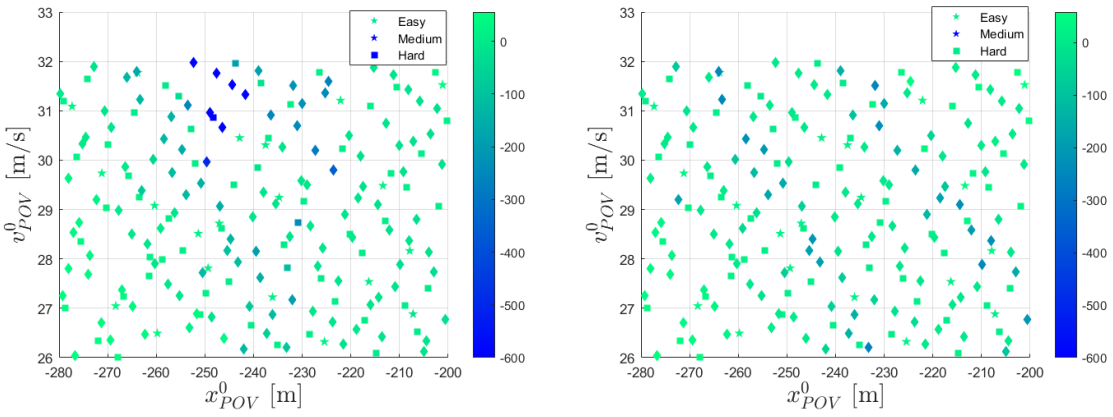


Figure 4.8: Corner case testing results at highway merging scenario. (a) VUT with GIDM algorithm; (b) VUT with deterministic sampling algorithm.

hard cases. It is shown that the proposed corner case generation method can create challenging test cases to evaluate different VUT algorithms. Moreover, the deterministic sampling method is superior to GIDM in this corner-case test.

Results for Pedestrian Crossing Scenario

For the pedestrian crossing scenario, we assign medium cases with $\lambda = 100$ and hard cases with $\lambda = 400$. The testing space is composed of two variables: the initial velocity of the pedestrian and the initial position of the VUT when the pedestrian is launched: $([v_p^0, x_{VUT}^0])$. The other initial

VUT type	Failure Cases
CA-VUT (nominal conflict region)	15/200
CV-VUT (nominal conflict region)	0/200
CV-VUT (west boundary -0.6m)	1/200
CV-VUT (west boundary -0.9m)	9/200

Table 4.3: Test results comparison for different VUTs for the pedestrian crossing scenario.

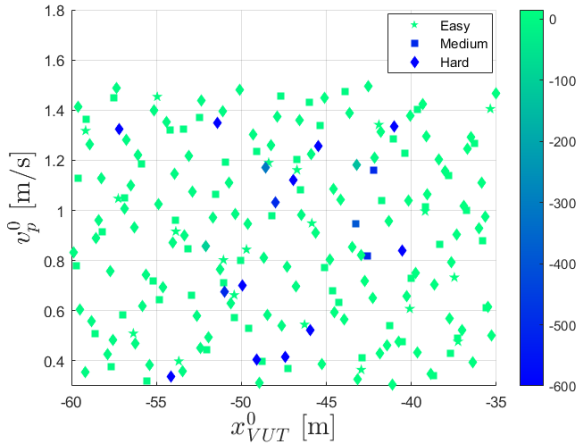


Figure 4.9: Corner case testing results for the CA-VUT at the pedestrian crossing scenario: 15/200 failures, in which 13/120 are hard cases, 2/60 are medium cases, 0/20 are easy cases.

conditions are the same as in Section 4.5.2.

The test library is created similarly as in the highway merging scenario. Assuming 200 cases in total, the allocation to easy, medium and hard cases is 20, 60 and 120 respectively. Since only pedestrians with goal G_1 are in potential conflict with the VUT, we assume all hard and medium cases have the goal as G_1. For easy cases, 60% of the cases target G_1, while the rest of the cases choose the goal randomly between G_2 and G_3. The reason for adding cases with other goal locations is to keep the uncertainty on the intended goal of the pedestrian, thereby avoiding the VUT to always assume that the pedestrian will go to G_1.

We compare the testing results of the CA-VUT and the CV-VUT on the same set of test cases. Moreover, for CV-VUT, we vary the assumed west boundary of the conflict region for computing the time-to-arrival for the pedestrian. The results are shown in Table 4.3. While the CA-VUT encounters 15 collisions, the nominal CV-VUT has none. On the other hand, by shrinking the left boundary of the conflict region, the CV-VUT has more collisions. Specifically, by reducing the assumed conflict region, the predicted pedestrian arrival time becomes later, which translates to a more aggressive VUT that chooses to pass under more circumstances.

Therefore, it is demonstrated that the CV-VUT has superior performance than the CA-VUT. Moreover, a conservative CV-VUT also has fewer failures than the more aggressive variants. In Figure 4.9, we show the performance scores of the CA-VUT under all test cases. It is again demonstrated that hard cases have a higher collision rate than medium cases, and no failure is found for easy cases.

4.5.4 Discussions

The results from case studies in 4.5.2 demonstrate that by increasing the ambiguity level, the behavior of the PORU becomes more erratic and ambiguous in both scenarios. Therefore, introducing ambiguity into the PORU planning algorithm creates corner cases in the qualitative sense. On the other hand, results from simulation tests in 4.5.3 demonstrate that by increasing the ambiguity level, higher rates of poor cases and failure cases are observed regardless of VUTs or scenarios. It shows that increasing the ambiguity level of the PORU creates more challenging cases quantitatively. Therefore, we can conclude that the proposed ambiguity concept is effective in generating corner cases for interactive scenarios. Moreover, it is shown that the ambiguity level provides us with a tuning knob for varying the difficulty of interactive test cases, which is similar to the risk level sets in Chapter 2. It improves the interpretability of corner cases, and helps generate test cases in a principled way.

4.6 Summary

This chapter proposes a systematic and generalizable method of modeling and creating corner cases for human-HAV interaction based on the notion of ambiguity. We model the PORU as a cost-minimizing agent controlled by MPC with hierarchical intentions. Then, ambiguity is defined as the entropy of the belief on PORU intentions by a surrogate prediction model, which is updated online according to a Bayesian inference procedure. To generate interactive behaviors that lead to corner cases, the MPC of the adversarial PORU is modified with an extra ambiguity cost, which encourages the PORU to take ambiguity-maximizing actions, while taking nominal behavior planning goals into consideration. Then, we implement this PORU algorithm in two interactive scenarios, highway merging and pedestrian crossing. In both scenarios, corner cases are successfully generated by increasing the ambiguity level. Finally, we propose a practical corner case testing procedure for both scenarios, using the ambiguity level to represent difficulties. The testing scheme presents objectively challenging test cases, and evaluates the safety performance of different VUTs, which meets the original goal of this study. As one of the first studies to consider corner cases in interactive scenarios, the proposed method could advance the field of HAV safety evaluation, and help us better understand the capability of HAVs in a realistic driving environment. In the future, we plan the improve the computation time of the current PORU algorithm by replacing the online MPC optimization with an offline-trained neural network using imitation learning, so that it is real-time implementable for complex interactive scenarios. Moreover, we plan to extend the framework to interactive scenarios with multiple PORUs, and create a multi-agent simulation environment to test the interaction capability of the HAV in a more realistic setting.

CHAPTER 5

Execution of the Behavior Competence Testing

As discussed in Section 1.3.3, once the test cases are generated, the last step is to execute them in a simulated or real-world environment accurately and reliably. This procedure is defined for the behavior competence test and shared across different testing paradigms. This chapter focuses on the execution procedure of behavior competence tests, specifically for reactive scenarios. We follow the pipeline in Figure 1.2, which consists of the following two tasks when applied to reactive scenarios:

1. Closed-loop motion synchronization in the preparation phase: synchronize the motion of the POV with the VUT, so that target test parameters are reached accurately.

2. Open-loop maneuver execution in the challenge phase: the POV executes the test maneuver to challenge the VUT.

The 1st task requires the POV to swiftly adjust its speed according to the motion of the VUT regardless of the speed profile of the VUT. This is the key to the successful execution of a test, thus it will be the focus of this chapter. We will first introduce the speed planning algorithm for motion synchronization for both cut-in and ULT scenarios. Then, we present the vehicle longitudinal dynamics modeling and control for the experimental vehicle platform to enable accurate speed tracking. Finally, we show the implementation results in both a simulated environment and the real Mcity test track. Regarding the 2nd task, the main requirement is to accurately track the predetermined maneuvers, including lane change, left turn, etc. This will be solved with an existing vehicle lateral controller from [165], and will not be addressed in detail.

5.1 Motion Synchronization for the POV

5.1.1 Problem Formulation

The POV needs to synchronize its speed profile with the VUT such that it can initiate the maneuver at the specified condition. Compared with existing research on vehicle speed planning and control,

where the focus is either the passenger comfort [166] or fuel/energy saving [167], we requires the POV to adapt its speed agilely and accurately according to the movement of the VUT in real-time. This problem is non-trivial due to the uncertain nature of VUT's behaviors and the nonlinearities in the powertrain dynamics of the POV.

We make the following assumptions. First, the POV knows the position and velocity of the VUT through communications, and in our case by using the 4G cellular network. Second, the target nominal speed or reference speed of the VUT is known to the POV, denoted as v^r_{VUT}.

Then, depending on the nature of the conflict point, we can categorize reactive scenarios into two types: scenarios with a floating conflict point and scenarios with a fixed conflict point. For the former, the conflict of POV and VUT can take place at any location along a road segment, e.g. cut-in scenario, opposite-direction-passing scenario, etc. The POV can initiate the maneuver whenever the set distance/time margin and speed value are met. For the latter type, the conflict must happen at a fixed location. Therefore, the POV needs to arrive at a certain location with the set distance/time margin as well as speed value before it initiates the maneuver. Examples are unprotected left turn, right turn, or other intersection scenarios with crossing paths. In the following sections, we introduce motion synchronization methods for both types separately, with one scenario for each as an example.

5.1.2 Motion Synchronization for the Cut-in Scenario

The cut-in scenario is a representative example of scenarios with a floating conflict point. The goal of motion synchronization is to reach the desired initial condition $[\Delta x^r, \Delta v^r]$ accurately before starting the lane change maneuver. Since all target cut-in scenarios will have the POV moving slower than VUT at the start of cut-in ($\Delta v(t_{LC}) > 0$), we can achieve motion synchronization with a simple design: the POV first accelerates and increases its range ahead from VUT, then reach the set speed, and wait for VUT to catch up. The detailed steps are the following:

1. When the test begins, the POV has an initial target speed of $v^r_{VUT} + v_{margin}$, until $\Delta x(t) = \Delta x^r + D_{margin}$, where v_{margin} and D_{margin} are parameters with small positive values.

2. POV tracks the target speed $v_{VUT}(t) - \Delta v^r$.

3. When the target distance is achieved, i.e. $\Delta x(t) = \Delta x^r$, the lane-change maneuver is initiated.

This process can be extended to other scenarios with a floating conflict point.

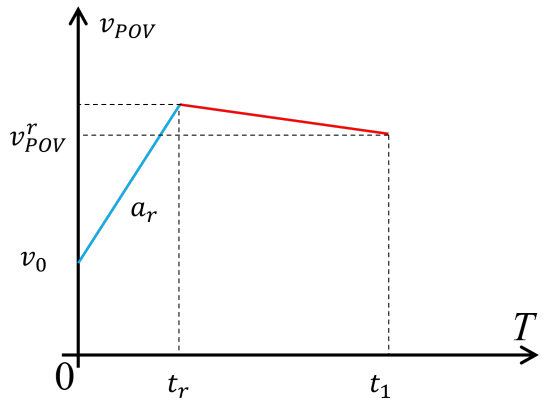


Figure 5.1: Proposed speed profile for PSP method

5.1.3 Motion Synchronization for the Unprotected Left Turn Scenario

We use the ULT scenario as the illustrative example for scenarios with a fixed conflict point. In the ULT scenario, the POV needs to reach the start of the intersection at the target speed v^r_{POV} when the VUT is Δx^r distance away, and execute the left turn with speed v^r_{POV} to pass the intersection ahead of the VUT. We presented two methods for this task.

Synchronization with a Parameterized Speed Profile (PSP)

A straightforward way is to assume a parameterized speed profile. At each timestep, we denote the current speed of POV as v_0, and the distance-to-go as s_0. Based on the predicted VUT motion, the time-to-go is denoted as t_1. The POV speed profile needs to simultaneously fulfill the above three constraints. If we propose a linear speed profile, then it only has 2 degree-of-freedom (slope and duration). Therefore, we propose a speed profile that consists of two phases: an accelerating phase followed by a decelerating phase, as shown in Figure 5.1. Then we have 4 degrees of freedom, which is enough to meet all the constraints and there is one spare parameter to tune for better performance. The speed profile can be solved with (5.1).

$$s_0 = v_0 t_r + a_r t_r^2/2 + (v_0 + a_r t_r + v^r_{POV})(t_1 - t_r)/2 \tag{5.1}$$

Here, t_r is the duration of the 1st phase, a_r is the acceleration during the 1st phase, i.e. the target acceleration for current time-step. By fixing t_r, we can solve the proposed acceleration. Different t_r will result in different shapes of the proposed speed profile.

At each time-step, we will update the speed profile based on the latest measurement of the kinematics of POV and VUT. As for the estimation of t_1, we predict the future speed of VUT based on an explicit rule: If VUT is slower than v^r_{VUT}, it will keep current acceleration until it

reaches v^r_{VUT}, then stay at constant speed; otherwise, it will keep the current speed. Based on the predicted VUT's speed profile and the distance-to-go, the time-to-go t_1 is estimated.

One issue that arises is the singular point of a_r as the VUT approaches the target point, i.e., when t_1 becomes infinitesimal. The POV thus needs infinitely large acceleration to hit the target exactly. To address this issue, we adopt a heuristic rule: when t_1 drops below a certain threshold T_{op}, the POV will try to match v^r_{POV} only. We adopt $T_{op} = 2s$ in our simulations and field tests.

Synchronization with Optimal Control

Alternatively, we can formulate the motion synchronization as an optimal control problem (OCP) with a finite horizon, and solve it using the Pontryagin maximum principle (PMP). The longitudinal dynamics of the POV can be modeled as a double-integrator. Then, the state-space model is

$$\dot{x} = Ax + Bu \tag{5.2}$$

where $x = [s_{POV}, v_{POV}]^T, u = a_{POV}$,

$$A = \begin{bmatrix} 0 & 1 \\ 0 & 0 \end{bmatrix}, \quad B = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

We assume that the initial condition x_0 and the end condition x_1 are both fixed and known. The time duration t_1 is based on the predicted VUT motion. To achieve a smooth and efficient speed profile, the cost function is quadratic with respect to the acceleration input, i.e. $J = \int_0^{t_1} u^2(t)dt$. Then, the Hamiltonian of this OCP is:

$$H(t) = p_0(\frac{1}{2}u(t)^2) + p^T(t)(Ax(t) + Bu(t)), p_0 \in \{0, 1\} \tag{5.3}$$

It can be shown that $p_0 = 1$. Then, according to the PMP, we have

$$\frac{\partial H}{\partial u} = u + B^T p = 0 \rightarrow u^*(t) = -B^T p(t) \tag{5.4}$$

The adjoint equation for costate p is:

$$\dot{p} = -\Big[\frac{\partial H}{\partial x}\Big]^T = -A^T p \tag{5.5}$$

Thus, the optimal control input can be solved analytically:

$$u^*(t) = -B^T e^{-A^T t} p(0) \tag{5.6}$$

For this particular system, we have:

$$u^*(t) = [t \quad -1]\, p(0) \tag{5.7}$$

Therefore, the optimal speed trajectory will have a constant acceleration derivative (jerk), i.e. $u^*(t) = k_1 t + k_2$. Given the initial and final conditions, we can compute the optimal acceleration profile by solving k_1 and k_2 analytically. At each time step, the speed profile is updated based on the latest measurement of the kinematics of POV and VUT.

To estimate the end time t_1, we predict the future average speed of VUT $\bar{v}_{VUT}$ to be a weighted sum of the nominal speed v^r_{VUT} and the current speed of the VUT. Then, t_1 is estimated by dividing the distance to-go of the VUT over $\bar{v}_{VUT}$.

5.2 Vehicle Longitudinal Dynamics and Control

5.2.1 Vehicle Longitudinal Model

The experimental vehicle is a 2015 Lincoln MKZ hybrid with an E-CVT transmission. The longitudinal dynamics of the system explain the relationship between the throttle / braking input (ϕ_t, ϕ_b), the speed v and the resulting acceleration a. It can be represented as the following [166]:

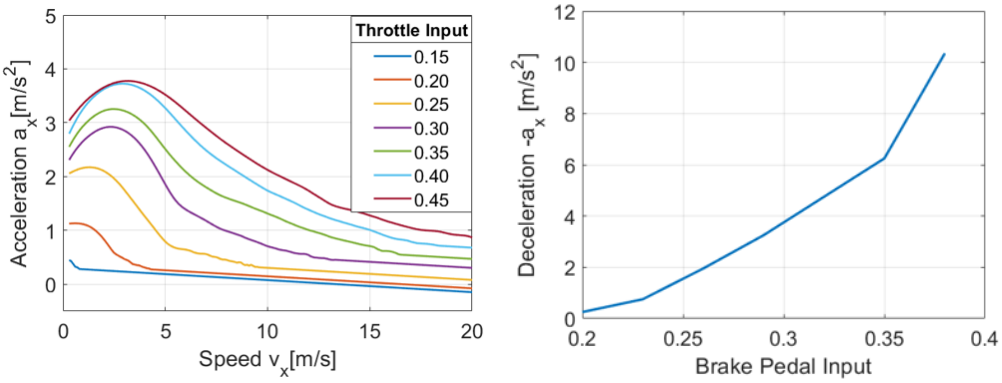
$$a = \Psi(v, \phi_t, \phi_b) - a_r(v) - g \sin\theta_r \tag{5.8}$$

Here, a_r is the lumped road load term (aerodynamic drag and rolling resistance), which is a function of the longitudinal speed; θ_r is the road grade; $\Psi(\cdot,\cdot,\cdot)$ is the nonlinear map between throttle/brake pedal input to the traction/braking force at tires. Assuming zero wind speed, a_r can be written as the following:

$$a_r(v) = F_{road} + F_{air} = Mg\cos\theta_r \mu_z + \frac{1}{2}\rho_{air} C_d A_f v^2 \tag{5.9}$$

where M is the mass of the vehicle, μ_z is the road rolling resistance coefficient, ρ_{air} is the air density, A_f and C_d are the frontal area and the drag coefficient of the vehicle respectively. The parameters are extracted according to the vehicle's properties (such as vehicle mass, etc), as well as from Chapter 4 of [168], which are presented in Table 5.1.

Since $\Psi(\cdot,\cdot,\cdot)$ is vehicle-dependent, we calibrate it for the experimental vehicle in a field experiment. The calibration took place in the Mcity on a straight road with constant and known inclination. Fixed throttle or brake command was sent to the vehicle through the by-wire control interface, and the speed profile of the vehicle was recorded. This was repeated for multiple runs



(a) $\Psi(v, \phi_t, \phi_b = 0)$ (b) $\Psi(v, \phi_t = 0, \phi_b)$

Figure 5.2: Throttle and brake map.

with different throttle/brake input values. The resulting throttle/brake map is shown in Figure 5.2. With these maps, we are able to translate the target acceleration to the throttle and brake command to the vehicle.

However, since $\Psi(\cdot, \cdot, \cdot)$ only accounts for the quasi-static relationship between pedal input and force at tires, a separate dynamic model is needed for the lag and delay in the powertrain of the vehicle, which has been observed during the experiment. We describe the longitudinal dynamics using a linear dynamic model with delay. The input is the acceleration command, the output is the actual acceleration and speed of the vehicle. The lag and delay are identified experimentally and modeled as:

$$\dot{x}(t) = A_c x(t) + B_c u(t - d_t), A_c = \begin{bmatrix} 0 & 1 \\ 0 & -1/\tau \end{bmatrix}, B_c = \begin{bmatrix} 0 \\ 1/\tau \end{bmatrix} \tag{5.10}$$

where the state is $x = [v, u_{eff}]^T$, u_{eff} is the effective input, and $u = a$ is the acceleration command, d_t is the time delay, and τ is the time constant of the first-order lag. Compared to [166], the delay is considered explicitly in the model. Then, assuming the delay d_t is a multiple of the time step T_s, we can rewrite the discretized system model into a linear state-space model:

$$x^{aug}(k+1) = A_{aug} x^{aug}(k) + B_{aug} u(k) \tag{5.11}$$

where

$$A_{aug} = \begin{bmatrix} A_d & B_{del} \\ 0 & A_{del} \end{bmatrix}, B_{aug} = \begin{bmatrix} 0_{(n_x + n_d - 1) \times 1} \\ 1 \end{bmatrix}$$

$$A_{del} = \begin{bmatrix} 0 & 1 & 0 & \cdots \\ 0 & 0 & 1 & \cdots \\ \vdots & \vdots & \ddots & \vdots \\ \cdots & 0 & 0 & 1 \\ \cdots & 0 & 0 & 0 \end{bmatrix}_{n_d \times n_d}, B_{del} = \begin{bmatrix} B_d & 0_{2\times(n_d-1)} \end{bmatrix},$$

$$x^{aug}(k) = \begin{bmatrix} x(k) \\ \tilde{u}(k) \end{bmatrix}, \quad \tilde{u}(k) = \begin{bmatrix} u(k-n_d) \\ \cdots \\ u(k-1) \end{bmatrix},$$

$$A_d = e^{A_c T_s}, \quad B_d = \big(\int_{kT_s}^{(k+1)T_s} e^{A_c((k+1)T_s - t)} dt \big) B_c$$

where $n_x = 2$, $n_d = d_t/T_s$.

Parameter	Value
Mass M	1800 kg
Air density ρ_{air}	1.23 kg/m^3
Drag coefficient C_d	0.30
Frontal area A_f	2.18 m^3
Rolling resistance coefficient μ_z	0.015
Time constant T_s	0.04 s
Time Delay d_t	0.40 s
Time constant τ	0.17 s

Table 5.1: Vehicle parameters.

5.2.2 Speed Control with Preview Control Method

To track the aforementioned speed profile accurately, we apply the preview control algorithm as the low-level speed controller. Preview control is based on the linear quadratic optimal control theory, where the future reference signals are ingested into an enlarged state-space model [166]. Compared to model-free PID control, the lag and delay in the system are explicitly represented in the dynamic model. Compared to the popular model predictive control (MPC), preview control does not need to solve an optimization problem numerically in real-time, which drastically reduces the computational load.

We design the preview tracking controller that takes in future desired speed profile: $V_d(k) = [v_d(k+1), v_d(k+2)...v_d(k+N_v)]^T$, where N_v is the horizon of the preview window. We apply the delta input formulation [102], and the transformed state-space model of the system will have

the extended state defined as:

$$\mathcal{X} = \begin{bmatrix} e_v(k) \\ \Delta x^{aug}(k)] \\ \Delta V_d(k) \end{bmatrix} = \begin{bmatrix} v(k) - v_d(k) \\ x^{aug}(k) - x^{aug}(k-1) \\ V_d(k) - V_d(k-1) \end{bmatrix},$$

and input: $\Delta u = u(k) - u(k-1)$.

Subsequently, we formulate an OCP with the cost function defined as:

$$\mathcal{J} = \sum_{k=0}^{\infty} \mathcal{X}^T(k)\mathrm{Q}\mathcal{X}(k) + \Delta u^T(k)\mathrm{R}\Delta u(k) \tag{5.12}$$

In Eq.(5.12), Q and R are the weighting matrices. The problem then becomes a standard linear quadratic regulator (LQR), for which there is a time-invariant state feedback gain K^* by solving a discrete-time algebraic Riccati Equation. Noted that since the control horizon is infinite and the preview horizon is finite, we set the preview speed beyond the horizon N_v to be zero, since their impact on the final control law can be ignored.

Finally, if we decompose the optimal gain as $K^* = [K_e, K_s, K_v]$, the final control law becomes:

$$u_c^*(k) = -\big(K_e \sum_{i=0}^{k} e_v(i) + K_s x(k) + K_v v_d(k+1:k+N_v)\big) \tag{5.13}$$

We refer interested readers to [166] for more details about the derivation. Finally, we compute the throttle and brake pedal command to the vehicle by plugging $a = u_c^*(k)$ into Eq. (5.8).

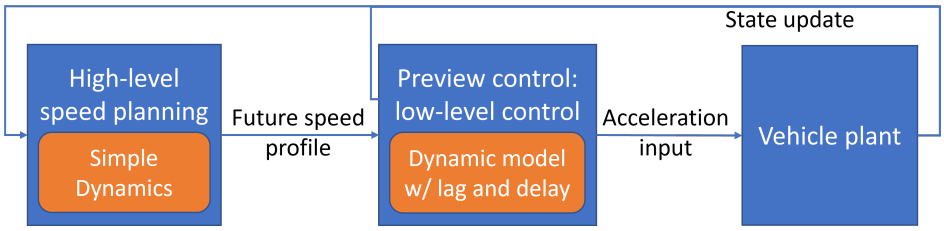
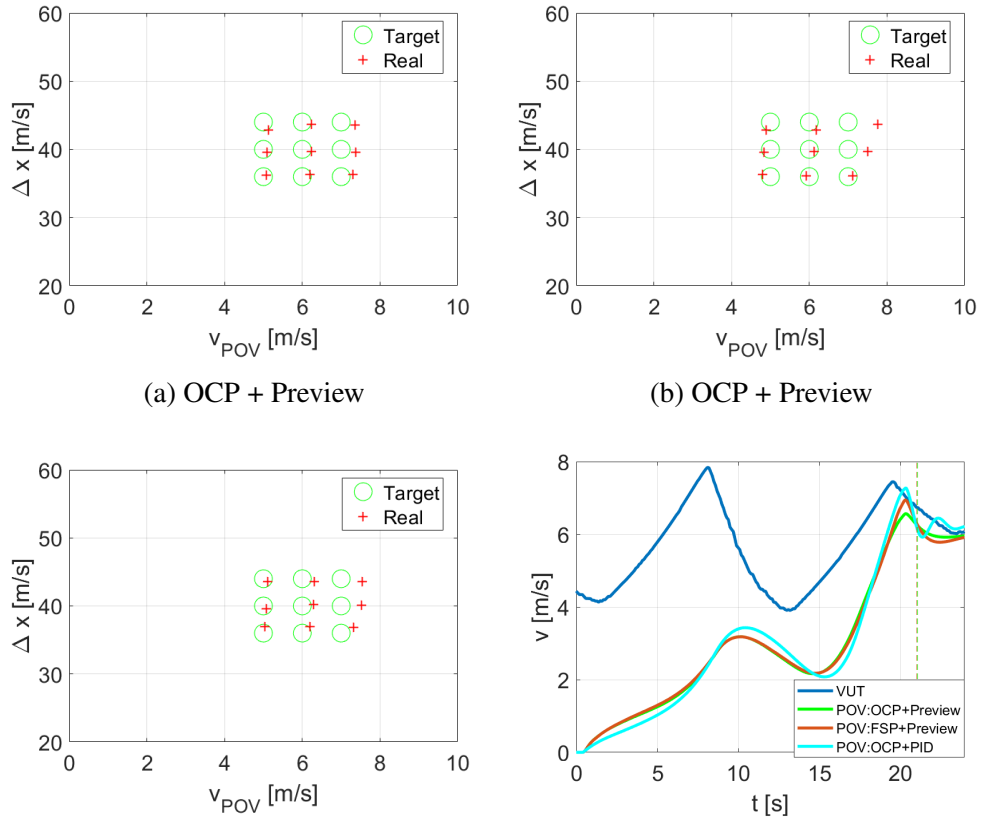


Figure 5.3: The diagram for the motion synchronization algorithm.

In summary, the overall motion synchronization algorithm is illustrated in Figure 5.3, which consists of a high-level speed planning module and a low-level speed tracking controller. The high-level planner computes an optimal speed profile to match the motion with the VUT by solving an OCP for a simplified vehicle longitudinal dynamic model. Then, the low-level controller applies the preview control algorithm for a more detailed dynamic model to track the high-level speed profile. This algorithm can be directly applied to many other test scenarios at the intersection,



(a) OCP + Preview

(b) OCP + Preview

(c) PSP + Preview

(d) The speed profiles

Figure 5.4: The simulation results of three motion synchronization methods for the ULT scenario. (a)-(c) show target-hitting performance comparison. (d) shows the speed curve comparison. The dashed lines represent the t_{LT} for each run respectively.

including roundabout entering, straight crossing path, etc.

5.3 Results in Simulation and Field Testing

5.3.1 Simulation Results for Motion Synchronization

We first present the simulation results of the proposed motion synchronization method for the ULT scenario. The VUT speed profile is recorded from an actual run of the experimental vehicle inside Mcity, using the planning and control stack introduced in [162], as shown in Figure 5.4d as the blue curve. The speed profile has several acceleration and deceleration phases because the route involves several turns at intersections. This varying speed profile presents challenges for speed synchronization on the POV. In simulations, the longitudinal dynamic model is the same as is discussed in 5.2.1.

Method	OCP+Preview	OCP+PID	PSP+Preview
Δx **mean error**	0.43 m	0.48 m	0.55 m
v_{POV} **mean error**	0.22 m/s	0.25 m/s	0.27 m/s

Table 5.2: Results on the accuracy of motion synchronization in simulation for the ULT scenario.

We compare the results of the two proposed methods (OCP+Preview and PSP+Preview) with another method, which is OCP for speed planning + PID control for speed tracking. We run simulations with the same initial condition for all test cases. The results are shown in Figure 5.4 (a)-(c) and in Table 5.2. It is shown that the OCP+Preview achieves the lowest error across multiple test cases for both target Δx^r and v^r_{POV}. For the other two methods, they both can achieve similar accuracy in some cases (e.g. when target $v^r_{POV} = 5.0$ m/s), but not in other cases, especially when the v^r_{POV} increases. In Figure 5.4d, we compare the resulting speed curve from the three methods in one case. Though the point-wise target-hitting performance is similar for these three runs, the proposed method generates the smoothest speed profile and tracks the target speed after t_{LT} more accurately.

5.3.2 Field Testing

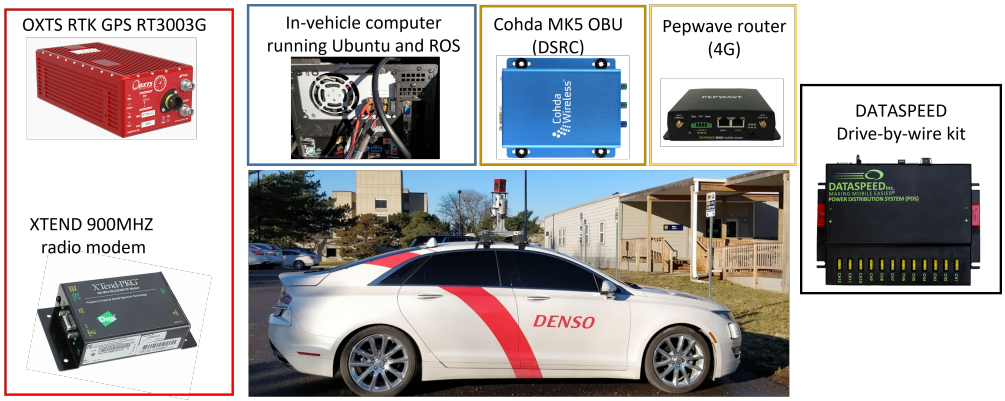


Figure 5.5: The hardware set-up of the experimental vehicle platform for POV.

We implement the test procedure for cut-in and ULT scenarios on our experimental vehicle platform which serves as the POV, and conduct field testing inside Mcity with a real VUT. Both vehicles are equipped with an RTK-GPS (Oxford RT3003) that can provide accurate position measurement ($\pm$ 2 cm), and a Pepwave cellular router that broadcasts Basic safety messages (BSMs) at around 5Hz (a DSRC onboard unit was used previously). The algorithms are implemented in C++ on a computer running Ubuntu OS and Robotic Operating System (ROS). The control frequency is 25 Hz. The hardware set-up of the POV is shown in Figure 5.5, and the VUT has a similar set-up.

Case Number	Cut-In		ULT	
	Δx (m)	Δv (m/s)	Δx (m)	v_{POV} (m/s)
1	12.0	1.0	40.0	5.0
2	10.0	1.0	36.0	5.0
3	8.0	1.0	32.0	5.0
4	12.0	1.5	40.0	6.0
5	10.0	1.5	36.0	6.0
6	8.0	1.5	32.0	6.0
7	12.0	2.0	40.0	7.0
8	10.0	2.0	36.0	7.0
9	8.0	2.0	32.0	7.0

Table 5.3: Test cases for real-world testing.

(a) (b)

Figure 5.6: Conducting real-world testing for (a) the cut-in scenario and (b) the unprotected left turn scenario. In each figure, the left vehicle is the POV while the right one is the VUT.

in The VUT uses planning and control algorithms from [162]. For the POV, we use the OCP as the speed planning algorithm and the above preview controller for speed-tracking. For lateral control, the POV tracks the given path with the preview controller from [165], which regulates the steering input. Figure 5.6 displays photos taken at the real-world testing for the cut-in and ULT scenarios. Nine predetermined test cases are conducted for each scenario to showcase the capability of test execution and scoring, whose initial conditions are listed in Table 5.3. All the cut-in cases have low risk level. For the ULT scenario, case No.4, No.7 are trivial, while all others have low risk.

For the cut-in scenario, the POV and the VUT travel on the highway section of Mcity, which is a 300-meter long two-lane straight road. The POV makes a lane change when the target Δx^r and Δv^r are met, as shown in Figure 5.7a. The motion synchronization results across nine cases are shown in Figure 5.7b. The proposed method can hit the target initial conditions for the cut-in

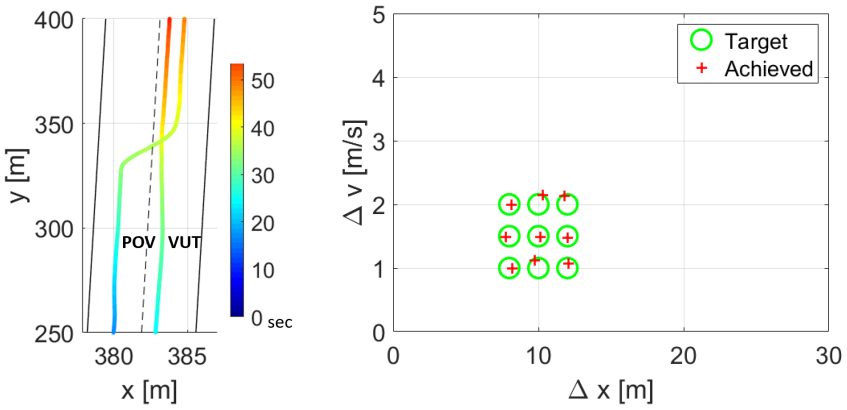


Figure 5.7: (a) The trajectories of POV and VUT in one cut-in case. Different colors represent waypoints at different timesteps (b) Motion synchronization results for multiple cut-in cases. The root-mean-square error (RMSE) on Δx is 0.19m, RMSE on Δv is 0.08 m/s.

accurately in all of the cases.

For the ULT scenario, we first present the testing results of one test case in Figure 5.8(a)-(c), with $v^r_{POV} = 6.0m/s$, $\Delta x^r = 40m$. The trajectory of the POV and VUT are demonstrated in Figure 5.8a. The POV hits the target initial condition accurately, creates a challenging scenario for the VUT and forces it to decelerate. To test the consistency of the motion synchronization method, we conducted 5 repetitive runs for this test case. The achieved result is $v_{POV} = 6.02 \pm 0.34m/s$, $\Delta x = 39.83 \pm 0.17m$, as shown in Figure 5.8c. The POV is able to consistently and accurately hit the target initial condition.

Then, we present the motion synchronization results at nine test cases in real-world tests in Figure 5.8d. All initial conditions can be accurately hit by the POV.

5.3.3 ABC Test Demo

Finally, we demonstrate our effort in extending this testing methodology to more scenarios. We conducted the Mcity ABC test demo in June 2021, where the VUT runs a route with seven challenging scenarios in autonomous mode. The choreography of all scenarios is shown in Figure 5.9a. Two POVs participated in this demo, which are both controlled by the proposed algorithms in this chapter. Here is a brief introduction to all scenarios that involve the POVs:

1. Ramp merge-in: The POV #1 participated in this scenario. POV #1 is controlled to arrive at the merge point with a predetermined distance margin and speed, and then coast at a constant speed. The VUT is anticipated to yield by slowing down. The motion synchronization

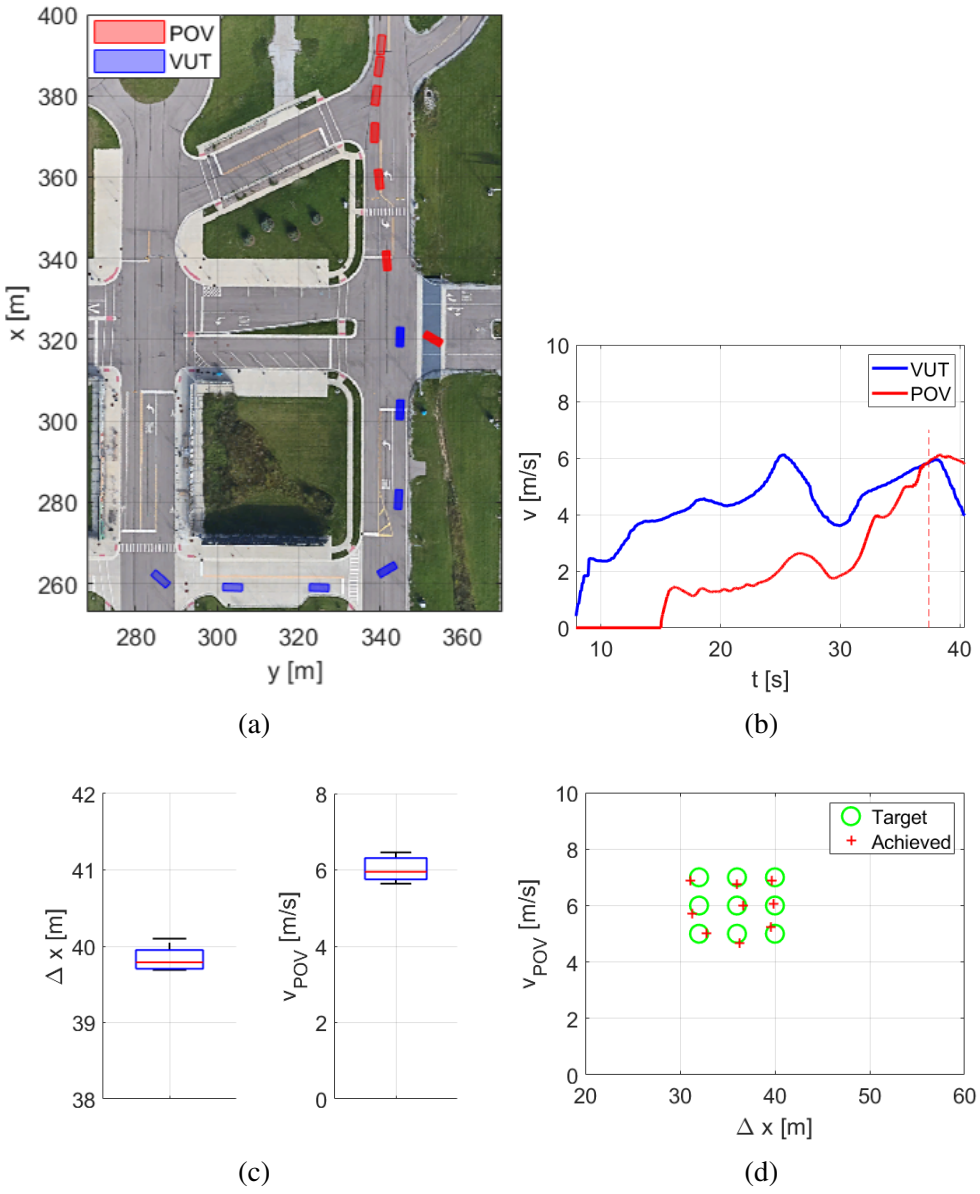
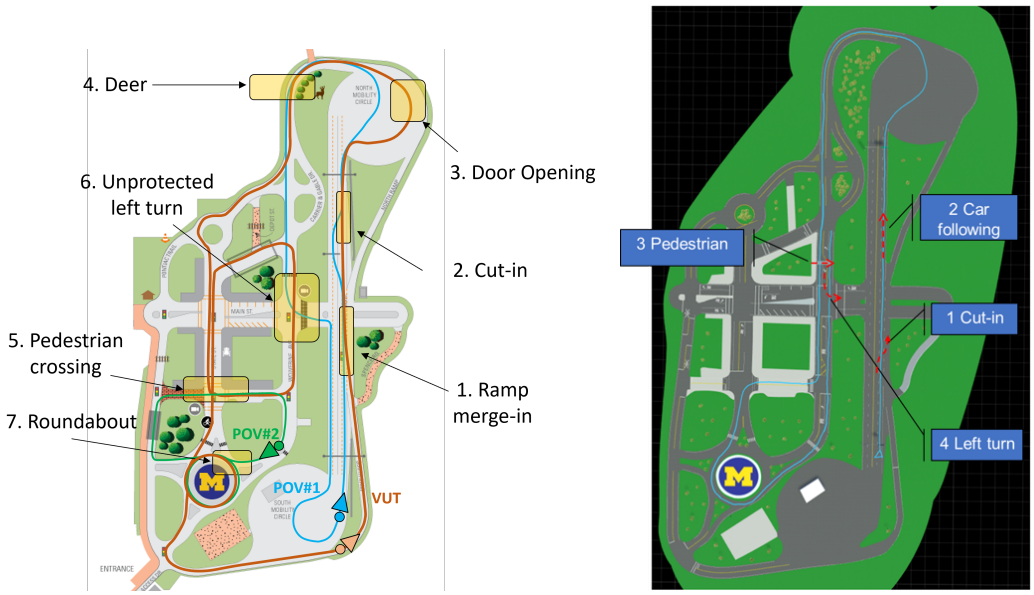


Figure 5.8: Motion synchronization results at the ULT scenario in field testing. (a) Trajectories of real POV and VUT in one test run. (b) The speed profiles; the dashed line represents t_{LT}. (c) Achieved results after 5 repeated runs of one test case. (d) Results of multiple ULT test cases. The root-mean-square error on v_{POV} and Δx are 0.20m/s, 0.56m respectively.



(a) (b)

Figure 5.9: The choreography of (a) the Mcity ABC test demo in the real world and (b) the Mcity CARLA challenge in simulation.

method is the same as introduced in 5.1.3.

2. Cut-in: the setting is the same as 5.3.2, where the POV #1 makes a lane change ahead of the VUT with a lower speed.

3. ULT: the setting is the same as 5.3.2, where the POV #1 turns left ahead of the VUT at a set speed with a set distance margin.

4. Roundabout: The POV #2 participated in this scenario. POV #2 is controlled to arrive at the entrance point of one branch of the roundabout ahead of the VUT with a certain time margin and speed. The VUT is anticipated to yield by slowing down. The motion synchronization method is the same as introduced in 5.1.3.

Note that the roundabout and ramp merge-in scenarios are viewed as reactive scenarios here for simplicity, different from the formulation in Chapter 3. The behavior of the POV is fixed after hitting the initial conditions. The link to the demo video is here.

5.4 Digital Twin of Mcity Test in CARLA Simulator

In addition, we have created the digital twin of the Mcity ABC test in the CARLA simulator [169]. This simulated version enables more agile development and more efficient testing.

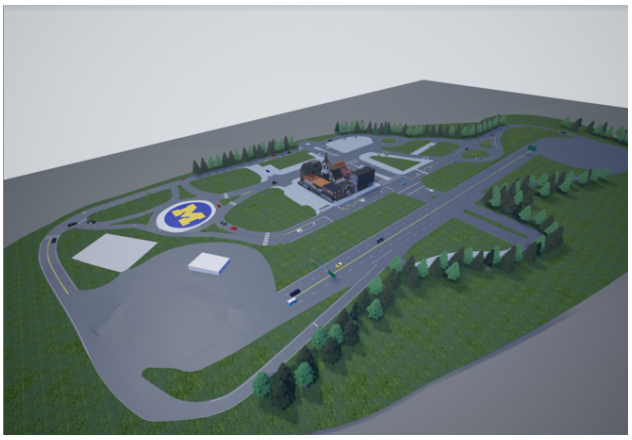


Figure 5.10: The Mcity in the CARLA simulator

CARLA has been a popular open-source simulator for autonomous driving research. Based on the Unreal game engine, it provided s state-of-the-art rendering quality, realistic physics, rich sensor models and well-developed APIs for interacting with the simulation environment [169]. Our group has developed a custom map of Mcity inside the CARLA simulator, which reconstructs the 2-D features of the real Mcity test track, as shown in Figure 5.10.

We implement a virtual version of the Mcity ABC test in the Mcity CARLA Map. Four scenarios are implemented: cut-in, car-following, ULT and pedestrian-crossing scenarios. We develop the API to control the generation and triggering of the POV or the pedestrian based on ScenarioRunner, the traffic scenario definition and execution engine for CARLA [170]. We integrate the tasks of test case sampling, simulation flow control, motion synchronization, test execution, and result visualization into a software with a graphical user interface (GUI), as shown in Figure 5.11.

The user can first select the target scenario, and then the number and difficulty levels of the test cases. Next, the user can connect the VUT of their own to the CARLA server, or specify a provided baseline VUT algorithm to be tested. The two baseline methods are: manual control using the keyboard, and the motion planning and control algorithm developed in [171]. By clicking ”Run”, the test cases for the target scenario will be generated and visualized in the bottom left corner, using methods introduced in 2.5.2. Then, the test cases will be executed one by one automatically. The POV or the pedestrian will adjust its speed and timing against the motion of the VUT using methods in 5.1.3 to hit the set initial condition of test cases. Screenshots for supported scenarios are presented in Figure 5.12. After the completion of each test case, the test results and details will be presented in the right lower panel of the GUI. More details could be found here.

In addition, we connected the targeted scenarios to create a route for competitive testing of self-driving algorithms, which become the ”Mcity CARLA challenge”. The choreography is similar to the real-world ABC testing, as demonstrated in 5.9b. The participated VUT is asked to follow a

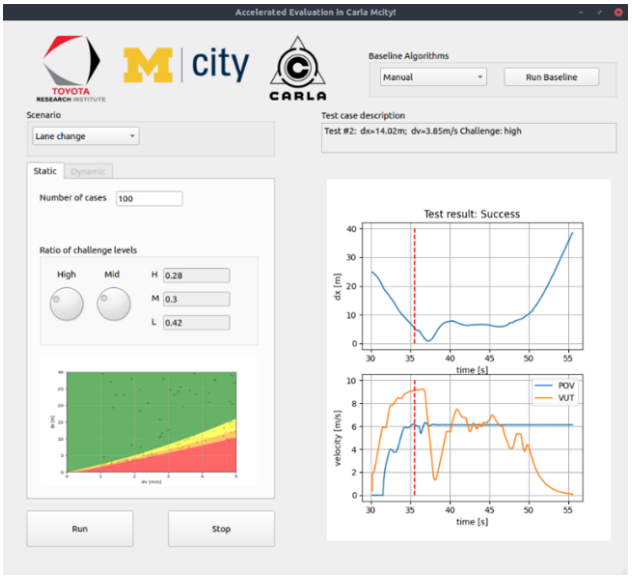


Figure 5.11: Software GUI for the ABC test in the CARLA Mcity.

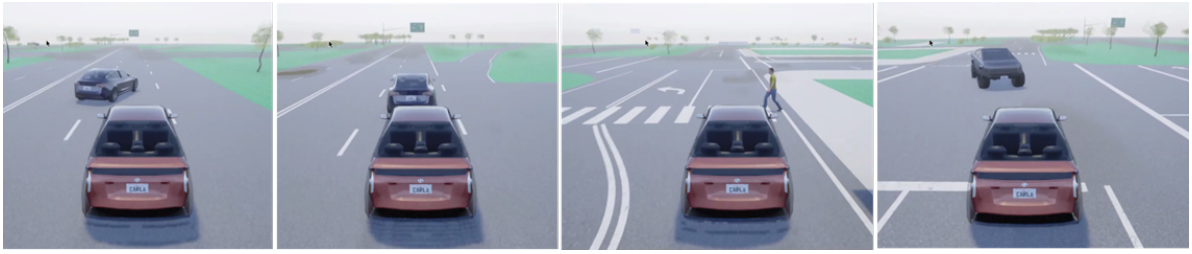


Figure 5.12: Screenshots of supported scenarios by the ABC test in the CARLA Mcity. From left to right: cut-in, car-following, pedestrian-crossing, ULT scenario.

nominal route, (the blue trace), on which the four scenarios are executed sequentially to challenge the VUT. The parameters of each scenario will be randomly sampled from its testing space according to the given risk level to ensure that the test cases cannot be anticipated in advance. In each run, only one participant will connect to the server remotely and control the VUT. The VUT will be graded based on safety, traveling time, path deviation, speed compliance and driving smoothness. More details can be found here.

5.5 Summary

This chapter presents the procedure and results of implementing behavior competence testing for HAVs in reactive scenarios. We first solve the motion synchronization problem for scenarios with both floating and fixed conflict points. Specifically, we propose two speed planning methods for scenarios with a fixed conflict point, one is based on a parameterized speed profile, and the other is

based on optimal control. Then, we present the longitudinal vehicle dynamics model of the experiment POV considering lag and delay. We then introduce the speed tracking controller based on the preview control algorithm, which analytically solves the control law considering the complex powertrain dynamics and future speed targets. Finally, we report simulation and real-world experiment results for both cut-in and ULT scenarios with the full speed planning and control algorithm. It is demonstrated that we can conduct behavior competence tests repeatably, accurately and robustly. In addition, we extend the testing framework into a digital twin of Mcity in the CARLA simulator, and create software tools for executing the tests in the simulation.

Finally, though we have focused on the implementation for reactive scenarios, the presented methods will also apply to interactive scenarios, since they follow the same pipeline in Figure 1.2. For the preparation phase, the same motion synchronization algorithm can be directly applied to interactive scenarios. For the challenge phase, the interactive POVs will no longer follow the predetermined maneuver. Instead, they may follow any of the closed-loop policies introduced in Chapter 3 or Chapter 4. The preview controller will be able to track the speed profile in real-time. The implementation of interactive scenarios will be a part of our future work.

CHAPTER 6

Conclusions and Future Work

6.1 Conclusions

In this dissertation, we present a framework and a suite of methods for conducting scenario-based safety evaluation for HAVs. The framework consists of three parts: scenario modeling and testing space formulation, test case generation, and the implementation of testing in simulated or real-world environments. We introduce the evaluation methodologies for both reactive and interactive scenarios, and for all three evaluation paradigms under the ABC test concept.

For reactive scenarios, we first create statistical models using naturalistic traffic data. We create the testing space with key variables that define the initial condition of a scenario. Then, we decompose the testing space into different risk level sets (RLSs) based on the required effort from the VUT to avoid collisions using backward reachability analysis. Subsequently, we generate test cases by conducting importance sampling on the RLSs. The feasibility and interpretability of test cases are guaranteed. Moreover, we are able to achieve unbiased crash rate estimation for the VUT in simulation tests with only a fraction of test cases compared with the baseline CMC method. The methodology is shown to be widely and directly applicable to multiple reactive scenarios.

For interactive scenarios, we enrich the testing space with an interaction-aware POV library, which is constructed based on the theory of level-k game and the concept of social value orientation. The POV library enables the representation of a wide variety of possible interactive behaviors of the POV. On the other hand, since the risk levels cannot be directly characterized for interactive scenarios, we proposed an adaptive test case generation method that discovers challenging yet diverse test samples from the testing space with both discrete and continuous attributes. This method is able to tailor test cases for each VUT to identify its failure modes more efficiently and comprehensively than other sampling methods.

In addition, we propose a systematic way of generating corner cases for interactive scenarios. Objectively adversarial PORU behaviors are created by injecting ambiguity into the nominal interaction-aware PORU agents, which are based on hierarchical decision-making and MPC. Based

on this method, the corner case testing process is designed and implemented for two typical interactive scenarios, highway merging and pedestrian crossing. The novel concept of ambiguity also serves a valid metric of the intrinsic difficulty of interactive driving scenarios.

Finally, we present the implementation and results of the behavior competence tests in both a real test track and the CARLA simulation software for multiple reactive scenarios. With the proposed planning and control algorithms, we can place the POV in the right place at the right timing with the right speed across different scenarios. Therefore, test cases can be executed in an accurate, robust, repeatable, and automated fashion.

In this dissertation, we demonstrate the procedure of scenario-based HAV safety evaluation in practice from start to finish, which may provide some instructions and guidelines for governments or 3rd-party organizations that want to conduct such a certification process for HAVs. With stochastically-generated test cases, VUTs cannot be "tuned" to pass tests like before, therefore the generalizability of test results is improved. With interpretable difficulty levels in both reactive scenarios (i.e. risk levels) and interactive scenarios (i.e. ambiguity levels), test cases can be generated in a principled and fair way for different VUTs. Specifically, our emphasis on modeling interactive scenarios could raise more attention on traffic interaction in the community of HAV evaluation. Most existing evaluation efforts do not handle interactive scenarios much differently from reactive scenarios, which could benefit from more explicit and realistic interactive POV models like ours. In addition, the implementation of behavior competence testing provides a promising prototype for a fully-automated testing procedure, which can be widely adopted for different scenarios by practitioners with little revision.

On the other hand, the proposed methods for scenario modeling and test case generation may provide tools and insights for self-driving research and development entities on the safety analysis and performance improvement of the HAV. The proposed adaptive test case generation methods can be applied to quickly identify the failure modes of the existing HAV stack, which could accelerate the iterative development of HAV software and hardware. The several proposed interactive driver models based on the theory of mind could enrich the agent simulation models in high-fidelity traffic simulators, which have been used extensively for both driving policy training and testing by many top HAV companies and research organizations. In addition, the proposed notion of "ambiguity" also provides a new viewpoint on the adversary for interaction-aware HAVs. Although it cannot explain all the interactive corner cases exhaustively, it may inspire the community to come up with more concepts to categorize the difficulty or complexity of interactive scenarios.

6.2 Future Directions

Reward learning for PORUs. In Chapter 3 and 4, the PORUs are modeled as utility-maximizing agents, where the utility functions are designed based on our domain knowledge. A natural extension would be to learn the utility functions of PORUs from real driving data. For this task, inverse reinforcement learning (IRL) techniques have been widely applied [159, 172, 173]. On the other hand, since PORUs have diverse individual attributes like personalities, SVO, etc., a key desideratum for such a method is to learn a universal underlying utility of PORUs while capturing this diversity in behaviors. Several recent studies have worked on this problem, assuming the diversity on discrete attributes [174–176]. In the future, we plan to develop an IRL method to learn the utility of PORUs that can consider both discrete (e.g. level-k) and continuous individual attributes (e.g. SVO) simultaneously.

Expanding to more complex scenarios. The studied scenarios in this dissertation mostly have one or two surrounding agents, all of which are PORUs. This has been a realistic assumption for acceptance testing with real vehicles. To expand the usefulness of the proposed scenario modeling techniques to development testing, we plan to work on modeling scenarios with more complex and realistic environments with more agents. There are two major potential challenges. The first one is to scale the current PORU modeling method to the simulation of large-scale traffic environments, while the second is to apply the test case generation method to high-dimensional testing spaces resulting from the more complex scenarios. To tackle these challenges, a possible direction is to combine the current PORU model with the state-of-the-art large-scale agent simulation framework reviewed in Section 1.2.2. The most relevant surrounding agents will be modeled by our PORU models, which create a local environment around the VUT with a controlled adversarial level. Other agents can be modeled by a naturalistic agent simulation model, as demonstrated in [81]. Moreover, this could reduce the dimensionality of the testing space by only varying the behaviors of the most relevant surrounding agents. Another stream of future work would be to investigate and improve the performance of the proposed adaptive sampling method in higher-dimensional testing space, ensuring good coverage of the failure modes for more complex scenarios.

Estimating the completeness of the failure mode identification process. The test case generation method proposed in Chapter 3 shows good performance in finding disjoint failure modes of the VUT. However, some natural follow-up questions are: How many failure modes could there be? How confident are we with that estimation? Characterizing the completeness of the failure modes identification process provides important insights for terminating the evaluation process, quantifying the HAV performance and enhancing public trust. Related work exists on probably approximately correct testing framework [67], and on uncertainty bound for probabilistic surrogate models like Gaussian Process Regression [177]. This topic will be explored more in the future.

Safeguard algorithm for real-world testing. For the implementation of behavior competence testing, another important consideration is the safety assurance for real-vehicle tests. Challenging test cases are informative, but a collision with a real VUT is hugely undesirable. In our current experiments, a safety driver always monitors the situation and takes avoidance maneuvers when needed. It can be dangerous for the driver, and the reaction of a human can be unreliable and inconsistent. In the next step, we plan to work on a safeguard algorithm for the POV, which can abort the test maneuver when the situation is deemed too dangerous, and then plan and execute a collision-avoidance maneuver for the POV at the last moment. The challenge lies in balancing the safety assurance and limiting the conservativeness of the safeguard, such that challenging tests can be executed safely, and will not be aborted prematurely.

Scoring criteria for acceptance testing. For the practitioners of acceptance testing, another crucial task to work on is the development of scoring and rating systems for HAVs. The VUT can be assessed with multiple objectives, including safety, smoothness, roadmanship (i.e. how human-like the VUT drives [178]), etc. The key is to integrate the multiple objectives into a final verdict for the VUT, which is not only a coherent measure of the VUT's performance, but also interpretable and perceivable by the public.

BIBLIOGRAPHY

[1] "J3016 - taxonomy and definitions for terms related to driving automation systems for on-road motor vehicles," 2021.

[2] D. J. Fagnant and K. Kockelman, "Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations," *Transportation Research Part A: Policy and Practice*, vol. 77, pp. 167–181, 2015.

[3] "Mercedes Drive Pilot Level 3 Autonomous System to Launch in Germany." [Online]. Available: https://www.caranddriver.com/news/a38475565/mercedes-drive-pilot-autonomous-germany/

[4] "TuSimple completes its first driverless autonomous truck run on public roads — TechCrunch." [Online]. Available: https://techcrunch.com/2021/12/29/tusimple-completes-its-first-driverless-autonomous-truck-run-on-public-roads/

[5] "Self-driving uber car that hit and killed woman did not recognize that pedestrians jaywalk." [Online]. Available: https://www.nbcnews.com/tech/tech-news/self-driving-uber-car-hit-killed-woman-did-not-recognize-n1079281

[6] "China's robotaxis charged ahead in 2021 — TechCrunch." [Online]. Available: https://techcrunch.com/2022/01/14/2021-robotaxi-china/

[7] "ISO - ISO 26262-1:2011 - Road vehicles — Functional safety." [Online]. Available: https://www.iso.org/standard/43464.html

[8] "ISO - ISO/PAS 21448:2019 - Road vehicles — Safety of the intended functionality." [Online]. Available: https://www.iso.org/standard/70939.html

[9] L. Fraade-Blanar, M. S. Blumenthal, J. M. Anderson, and N. Kalra, *Measuring automated vehicle safety: Forging a framework*, 2018.

[10] V. L. Neale, T. A. Dingus, S. G. Klauer, J. Sudweeks, and M. Goodman, "An overview of the 100-car naturalistic study and findings," *National Highway Traffic Safety Administration, Paper*, vol. 5, p. 0400, 2005.

[11] J. Sayer, D. LeBlanc, S. Bogard, D. Funkhouser, S. Bao, M. L. Buonarosa, A. Blankespoor *et al.*, "Integrated vehicle-based safety systems field operational test: Final program report," United States. Joint Program Office for Intelligent Transportation Systems, Tech. Rep., 2011.

[12] K. Gay and V. Kniss, "Safety Pilot Model Deployment Lessons Learned and Recommendations for Future Connected Vehicle Activities," U.S. Department of Transportation, Tech. Rep., 2015.

[13] "Autonomous Vehicle Testing Permit Holders - California DMV." [Online]. Available: https://www.dmv.ca.gov/portal/vehicle-industry-services/autonomous-vehicles/autonomous-vehicle-testing-permit-holders/

[14] "Waymo's autonomous vehicles have clocked 20 million miles on public roads — Engadget." [Online]. Available: https://www.engadget.com/waymo-autonomous-vehicles-update-san-francisco-193934150.html

[15] H. Caesar, V. Bankiti, A. H. Lang, S. Vora, V. E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, and O. Beijbom, "nuscenes: A multimodal dataset for autonomous driving," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 11 621–11 631.

[16] M.-F. Chang, J. Lambert, P. Sangkloy, J. Singh, S. Bak, A. Hartnett, D. Wang, P. Carr, S. Lucey, D. Ramanan *et al.*, "Argoverse: 3d tracking and forecasting with rich maps," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 8748–8757.

[17] "Traffic Safety Facts 2019: A Compilation of Motor Vehicle Crash Data," NHTSA, Tech. Rep. [Online]. Available: https://crashstats.nhtsa.dot.gov/.

[18] N. Kalra and S. M. Paddock, "How many miles of driving would it take to demonstrate autonomous vehicle reliability," *RAND Corporation, Santa Monica, CA, Tech. Rep*, pp. 1129–1134, 2016.

[19] S. Riedmaier, T. Ponn, D. Ludwig, B. Schick, and F. Diermeyer, "Survey on scenario-based safety assessment of automated vehicles," *IEEE Access*, vol. 8, pp. 87 456–87 477, 2020.

[20] "Pegasus method - pegasus-en." [Online]. Available: https://www.pegasusprojekt.de/en/pegasus-method

[21] "SAKURA Project." [Online]. Available: https://www.sakura-prj.go.jp/

[22] S. Ulbrich, T. Menzel, A. Reschka, F. Schuldt, and M. Maurer, "Defining and substantiating the terms scene, situation, and scenario for automated driving," in *2015 IEEE 18th international conference on intelligent transportation systems*. IEEE, 2015, pp. 982–988.

[23] T. Menzel, G. Bagschik, and M. Maurer, "Scenarios for development, test and validation of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1821–1827.

[24] A. Svensson, *A method for analysing the traffic process in a safety perspective*. Lund Institute of Technology Sweden, 1998.

[25] E. D. Swanson, F. Foderaro, M. Yanagisawa, W. G. Najm, P. Azeredo *et al.*, "Statistics of light-vehicle pre-crash scenarios based on 2011–2015 national crash data," United States. Department of Transportation. National Highway Traffic Safety ..., Tech. Rep., 2019.

[26] E. de Gelder, O. O. den Camp, and N. de Boer, "Scenario categories for the assessment of automated vehicles," *CETRAN, Singapore, Version*, vol. 1, 2020.

[27] E. Thorn, S. C. Kimmel, M. Chaka, B. A. Hamilton *et al.*, "A framework for automated driving system testable cases and scenarios," United States. Department of Transportation. National Highway Traffic Safety Administration, Tech. Rep., 2018.

[28] "Waymo Safety Report," Tech. Rep., 2021.

[29] H. Peng and R. McCarthy, "Mcity abc test," 2019.

[30] G. Bagschik, T. Menzel, and M. Maurer, "Ontology based scene creation for the development of automated vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2018, pp. 1813–1820.

[31] J. Sun, H. Zhou, H. Zhang, Y. Tian, and Q. Ji, "Adaptive design of experiments for accelerated safety evaluation of automated vehicles," in *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2020, pp. 1–7.

[32] D. Zhao, H. Lam, H. Peng, S. Bao, D. J. LeBlanc, K. Nobukawa, and C. S. Pan, "Accelerated Evaluation of Automated Vehicles Safety in Lane-Change Scenarios Based on Importance Sampling Techniques," *IEEE Transactions on Intelligent Transportation Systems*, vol. 18, no. 3, pp. 595–607, 3 2017.

[33] S. Zhang, H. Peng, D. Zhao, and H. E. Tseng, "Accelerated Evaluation of Autonomous Vehicles in the Lane Change Scenario Based on Subset Simulation Technique," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 11 2018, pp. 3935–3940.

[34] S. Feng, Y. Feng, C. Yu, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part i: Methodology," *IEEE Transactions on Intelligent Transportation Systems*, 2020.

[35] C. E. Tuncali, T. P. Pavlic, and G. Fainekos, "Utilizing S-TaLiRo as an automatic test generation framework for autonomous vehicles," *IEEE Conference on Intelligent Transportation Systems, Proceedings, ITSC*, no. ii, pp. 1470–1475, 2016.

[36] D. Zhao, X. Huang, H. Peng, H. Lam, and D. J. LeBlanc, "Accelerated Evaluation of Automated Vehicles in Car-Following Maneuvers," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 3, pp. 733–744, 3 2018.

[37] S. Feng, Y. Feng, H. Sun, S. Bao, Y. Zhang, and H. X. Liu, "Testing scenario library generation for connected and automated vehicles, part ii: Case studies," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 9, pp. 5635–5647, 2020.

[38] A. Corso, P. Du, K. Driggs-Campbell, and M. J. Kochenderfer, "Adaptive stress testing with reward augmentation for autonomous vehicle validation," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 163–168.

[39] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on Control Systems Technology*, vol. 26, no. 5, pp. 1782–1797, 2018.

[40] L. Bergamini, Y. Ye, O. Scheel, L. Chen, C. Hu, L. Del Pero, B. Osiński, H. Grimmett, and P. Ondruska, "Simnet: Learning reactive self-driving simulations from real-world observations," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 5119–5125.

[41] S. Suo, S. Regalado, S. Casas, and R. Urtasun, "Trafficsim: Learning to simulate realistic multi-agent behaviors," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 10 400–10 409.

[42] D. Xu, Y. Chen, B. Ivanovic, and M. Pavone, "Bits: Bi-level imitation for traffic simulation," *arXiv preprint arXiv:2208.12403*, 2022.

[43] M. Igl, D. Kim, A. Kuefler, P. Mougin, P. Shah, K. Shiarlis, D. Anguelov, M. Palatucci, B. White, and S. Whiteson, "Symphony: Learning realistic and diverse agents for autonomous driving simulation," *arXiv preprint arXiv:2205.03195*, 2022.

[44] Q. Zhang, Y. Gao, Y. Zhang, Y. Guo, D. Ding, Y. Wang, P. Sun, and D. Zhao, "Trajgen: Generating realistic and diverse trajectories with reactive and feasible agent behaviors for autonomous driving," *arXiv preprint arXiv:2203.16792*, 2022.

[45] M. Zhou, J. Luo, J. Villella, Y. Yang, D. Rusu, J. Miao, W. Zhang, M. Alban, I. Fadakar, Z. Chen *et al.*, "Smarts: Scalable multi-agent reinforcement learning training school for autonomous driving," *arXiv preprint arXiv:2010.09776*, 2020.

[46] NCAP, "European new car assessment programme - test protocol – aeb systems," Tech. Rep., 2015.

[47] ——, "European new car assessment programme - test protocol – lane support systems," Tech. Rep., 2015.

[48] IIHS, "Autonomous emergency braking test protocol (version i)," 2013. [Online]. Available: http://www.iihs.org/iihs/ratings/technical-information/technical-protocols

[49] W.-L. Loh, "On latin hypercube sampling," *The annals of statistics*, vol. 24, no. 5, pp. 2058–2080, 1996.

[50] C. E. Tuncali, G. Fainekos, D. Prokhorov, H. Ito, and J. Kapinski, "Requirements-driven test generation for autonomous vehicles with machine learning components," *IEEE Transactions on Intelligent Vehicles*, vol. 5, no. 2, pp. 265–280, 2019.

[51] H.-H. Yang and H. Peng, "Development and evaluation of collision warning/collision avoidance algorithms using an errable driver model," *Vehicle System Dynamics*, vol. 48, no. sup1, pp. 525–535, 12 2010.

[52] T. A. Wheeler and M. J. Kochenderfer, "Factor graph scene distributions for automotive safety analysis," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2016, pp. 1035–1040.

[53] M. O'Kelly, A. Sinha, H. Namkoong, R. Tedrake, and J. C. Duchi, "Scalable end-to-end autonomous vehicle testing via rare-event simulation," *Advances in neural information processing systems*, vol. 31, 2018.

[54] S. Feng, Y. Feng, H. Sun, Y. Zhang, and H. X. Liu, "Testing Scenario Library Generation for Connected and Automated Vehicles: An Adaptive Framework," *IEEE Transactions on Intelligent Transportation Systems*, pp. 1–10, 3 2020.

[55] J. Norden, M. O'Kelly, and A. Sinha, "Efficient black-box assessment of autonomous vehicle safety," *arXiv preprint arXiv:1912.03618*, 2019.

[56] S.-K. Au and J. Beck, "Important sampling in high dimensions," *Structural safety*, vol. 25, no. 2, pp. 139–163, 2003.

[57] M. Althoff and S. Lutz, "Automatic Generation of Safety-Critical Test Scenarios for Collision Avoidance of Road Vehicles," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2018-June. IEEE, 6 2018, pp. 1326–1333.

[58] M. Klischat and M. Althoff, "Generating critical test scenarios for automated vehicles with evolutionary algorithms," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 2352–2358.

[59] G. Chou, Y. E. Sahin, L. Yang, K. J. Rutledge, P. Nilsson, and N. Ozay, "Using control synthesis to generate corner cases: A case study on autonomous driving," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 11, pp. 2906–2917, 2018.

[60] P. Akella, M. Ahmadi, R. M. Murray, and A. D. Ames, "Formal test synthesis for safety-critical autonomous systems based on control barrier functions," in *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 2020, pp. 790–795.

[61] C. E. Tuncali and G. Fainekos, "Rapidly-exploring random trees-based test generation for autonomous vehicles," *arXiv preprint arXiv:1903.10629*, 2019.

[62] T. Dreossi, T. Dang, A. Donzé, J. Kapinski, X. Jin, and J. V. Deshmukh, "Efficient guiding strategies for testing of temporal properties of hybrid systems," in *NASA Formal Methods Symposium*. Springer, 2015, pp. 127–142.

[63] A. Wachi, "Failure-scenario maker for rule-based agent using multi-agent adversarial reinforcement learning and its application to autonomous driving," *arXiv preprint arXiv:1903.10654*, 2019.

[64] S. Zhang, H. Peng, S. Nageshrao, and H. E. Tseng, "Generating socially acceptable perturbations for efficient evaluation of autonomous vehicles," *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 2020-June, pp. 1341–1347, 2020.

[65] B. Chen, X. Chen, Q. Wu, and L. Li, "Adversarial evaluation of autonomous vehicles in lane-change scenarios," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[66] B. Gangopadhyay, S. Khastgir, S. Dey, P. Dasgupta, G. Montana, and P. Jennings, "Identification of test cases for automated driving systems using bayesian optimization," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2019, pp. 1961–1967.

[67] L. Li, N. Zheng, and F.-Y. Wang, "A theoretical foundation of intelligence testing and its application for intelligent vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. 22, no. 10, pp. 6297–6306, 2020.

[68] G. E. Mullins, P. G. Stankiewicz, and S. K. Gupta, "Automated generation of diverse and challenging scenarios for test and evaluation of autonomous vehicles," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 1443–1450, 2017.

[69] Y. Abeysirigoonawardena, F. Shkurti, and G. Dudek, "Generating adversarial driving scenarios in high-fidelity simulators," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8271–8277.

[70] J. Wang, A. Pun, J. Tu, S. Manivasagam, A. Sadat, S. Casas, M. Ren, and R. Urtasun, "Advsim: Generating safety-critical scenarios for self-driving vehicles," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 9909–9918.

[71] "How agent simulation enables the collective safety assessment of AV in simulation at Cruise — LinkedIn." [Online]. Available: https://www.linkedin.com/pulse/how-agent-simulation-enables-collective-safety-assessment-jing-lu/?trackingId=5pC9%2B41URJWSQCfIDBLBkw%3D%3D

[72] W. Ding, M. Xu, and D. Zhao, "Cmts: A conditional multiple trajectory synthesizer for generating safety-critical driving scenarios," in *2020 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2020, pp. 4314–4321.

[73] D. Rempe, J. Philion, L. J. Guibas, S. Fidler, and O. Litany, "Generating useful accident-prone driving scenarios via a learned traffic prior," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 17 305–17 315.

[74] J. Kapinski, J. V. Deshmukh, X. Jin, H. Ito, and K. Butts, "Simulation-based approaches for verification of embedded control systems: An overview of traditional and advanced modeling, testing, and verification techniques," *IEEE Control Systems Magazine*, vol. 36, no. 6, pp. 45–64, 2016.

[75] S. A. Seshia, D. Sadigh, and S. S. Sastry, "Formal methods for semi-autonomous driving," in *2015 52nd ACM/EDAC/IEEE Design Automation Conference (DAC)*. IEEE, 2015, pp. 1–5.

[76] M. E. O'Kelly, H. Abbas, S. Gao, S. Kato, S. Shiraishi, and R. Mangharam, "Apex: Autonomous vehicle plan verification and execution," SAE Technical Paper, Tech. Rep., 2016.

[77] T. Kaga, M. Adachi, I. Hosotani, and M. Konishi, "Validation of control software specification using design interests extraction and model checking," SAE Technical Paper, Tech. Rep., 2012.

[78] A. Abhishek, H. Sood, and J.-B. Jeannin, "Formal verification of braking while swerving in automobiles," in *Proceedings of the 23rd International Conference on Hybrid Systems: Computation and Control*, 2020, pp. 1–11.

[79] S. Shalev-Shwartz, S. Shammah, and A. Shashua, "On a formal model of safe and scalable self-driving cars," *arXiv preprint arXiv:1708.06374*, 2017.

[80] S. Kitajima, K. Shimono, J. Tajima, J. Antona-Makoshi, and N. Uchida, "Multi-agent traffic simulations to estimate the impact of automated technologies on safety," *Traffic injury prevention*, vol. 20, no. sup1, pp. S58–S64, 2019.

[81] S. Feng, X. Yan, H. Sun, Y. Feng, and H. X. Liu, "Intelligent driving intelligence test for autonomous vehicles with naturalistic and adversarial environment," *Nature communications*, vol. 12, no. 1, pp. 1–14, 2021.

[82] "Future certification of automated/autonomous driving systems," 2019.

[83] J. Sun, H. Zhang, H. Zhou, R. Yu, and Y. Tian, "Scenario-based test automation for highly automated vehicles: A review and paving the way for systematic safety assurance," *IEEE Transactions on Intelligent Transportation Systems*, 2021.

[84] Y. Ma, C. Sun, J. Chen, D. Cao, and L. Xiong, "Verification and validation methods for decision-making and planning of automated vehicles: A review," *IEEE Transactions on Intelligent Vehicles*, 2022.

[85] National Highway Traffic Safety Administration, "Traffic Safety Facts: 2017 data: pedestrians," NHTSA, Tech. Rep. March, 2019.

[86] M. Enzweiler and D. Gavrila, "Monocular Pedestrian Detection: Survey and Experiments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 12, pp. 2179–2195, 12 2009.

[87] P. Dollár, C. Wojek, B. Schiele, and P. Perona, "Pedestrian Detection: A Benchmark," in *CVPR*, 2009.

[88] B. Chen, D. Zhao, and H. Peng, "Evaluation of automated vehicles encountering pedestrians at unsignalized crossings," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, no. February. IEEE, 6 2017, pp. 1679–1685.

[89] B. Schroeder, N. Rouphail, K. Salamati, E. Hunter, B. Phillips, L. Elefteriadou, T. Chase, and Y. Zheng, "Empirically-Based Performance Assessment and Simulation of Pedestrian Behavior at Unsignalized Crossings," Tech. Rep., 2014.

[90] R. Knoblauch, M. Pietrucha, and M. Nitzburg, "Field Studies of Pedestrian Walking Speed and Start-Up Time," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 1538, pp. 27–38, 1996.

[91] K. Ismail, T. Sayed, and N. Saunier, "Automated Analysis of Pedestrian-Vehicle," *Transportation Research Record: Journal of the Transportation Research Board*, vol. 2198, no. 2198, pp. 52–64, 2010.

[92] "View Mobotix camera in Canada. URL: https://www.insecam.org/en/view/516710/."

[93] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," 4 2018.

[94] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision, Second Edition*, 2000.

[95] J. Munkres, "Algorithms for the assignment and transportation problems," *Journal of the society for industrial and applied mathematics*, vol. 5, no. 1, pp. 32–38, 1957.

[96] S. Bennett, A. Felton, and R. Akçelik, "Pedestrian movement characteristics at signalised intersections," in *23rd Conference of Australian Institutes of Transport Research*, no. December, 2001, pp. 10–12.

[97] G. Lee and C. Scott, "EM algorithms for multivariate Gaussian mixture models with truncated and censored data," *Computational Statistics and Data Analysis*, vol. 56, no. 9, pp. 2816–2829, 2012.

[98] A. B. Owen, *Monte Carlo theory, methods and examples*, 2013.

[99] J. Blanchet and H. Lam, "State-dependent importance sampling for rare-event simulation: An overview and recent advances," *Surveys in Operations Research and Management Science*, vol. 17, no. 1, pp. 38–59, 1 2012.

[100] M. Althoff, "Reachability Analysis and its Application to the Safety Assessment of Autonomous Cars," *Fakultät für Elektrotechnik und Informationstechnik*, p. 221, 2010.

[101] J. Nilsson, J. Fredriksson, and A. C. Ödblom, "Verification of Collision Avoidance Systems using Reachability Analysis," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 10 676–10 681, 2014.

[102] F. Borrelli, A. Bemporad, and M. Morari, *Predictive Control For Linear and Hybrid Systems*. Cambridge University Press, 2017.

[103] N. Li, K. Han, A. Girard, H. E. Tseng, D. Filev, and I. Kolmanovsky, "Action governor for discrete-time linear systems with non-convex constraints," *IEEE Control Systems Letters*, vol. 5, no. 1, pp. 121–126, 2020.

[104] K. Vogel, "A comparison of headway and time to collision as safety indicators," *Accident Analysis and Prevention*, vol. 35, no. 3, pp. 427–433, 2003.

[105] M. Herceg, M. Kvasnica, C. N. Jones, and M. Morari, "Multi-Parametric Toolbox 3.0," in *2013 European Control Conference (ECC)*, Zürich, Switzerland, 7 2013, pp. 502–510.

[106] K. Lee and H. Peng, "Evaluation of automotive forward collision warning and collision avoidance algorithms," *Vehicle System Dynamics*, vol. 43, no. 10, pp. 735–751, 10 2005.

[107] W. C. Horrace, "Some results on the multivariate truncated normal distribution," *Journal of Multivariate Analysis*, vol. 94, no. 1, pp. 209–221, 5 2005.

[108] T. I. Gorman, "Prospects for the collision-free car: The effectiveness of five competing forward collision avoidance systems," Ph.D. dissertation, Virginia Tech, 2013.

[109] X. Wang, D. Zhao, H. Peng, and D. J. LeBlanc, "Analysis of unprotected intersection left-Turn conflicts based on naturalistic driving data," in *IEEE Intelligent Vehicles Symposium*, 2017, pp. 218–223.

[110] V. A. Butakov and P. Ioannou, "Personalized driver/vehicle lane change models for adas," *IEEE Transactions on Vehicular Technology*, vol. 64, no. 10, pp. 4422–4431, 2014.

[111] L. Sun, W. Zhan, Y. Hu, and M. Tomizuka, "Interpretable Modelling of Driving Behaviors in Interactive Driving Scenarios based on Cumulative Prospect Theory," in *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*. IEEE, 10 2019, pp. 4329–4335.

[112] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[113] A. Kesting, M. Treiber, and D. Helbing, "General lane-changing model mobil for car-following models," *Transportation Research Record*, vol. 1999, no. 1, pp. 86–94, 2007.

[114] Y. Hu, W. Zhan, L. Sun, and M. Tomizuka, "Multi-modal probabilistic prediction of interactive behavior via an interpretable model," in *2019 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2019, pp. 557–563.

[115] J. Li, H. Ma, and M. Tomizuka, "Interaction-aware multi-agent tracking and probabilistic behavior prediction via adversarial learning," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 6658–6664.

[116] W. Schwarting, A. Pierson, J. Alonso-Mora, S. Karaman, and D. Rus, "Social behavior for autonomous vehicles," *Proceedings of the National Academy of Sciences of the United States of America*, vol. 116, no. 50, pp. 2492–24 978, 2019.

[117] J. F. Fisac, E. Bronstein, E. Stefansson, D. Sadigh, S. S. Sastry, and A. D. Dragan, "Hierarchical game-theoretic planning for autonomous vehicles," in *Proceedings - IEEE International Conference on Robotics and Automation*, vol. 2019-May, 2019, pp. 9590–9596.

[118] J. H. Yoo and R. Langari, "A stackelberg game theoretic driver model for merging," *ASME 2013 Dynamic Systems and Control Conference, DSCC 2013*, vol. 2, pp. 1–8, 2013.

[119] J. Geary, H. Gouk, and S. Ramamoorthy, "Active altruism learning and information sufficiency for autonomous driving," *arXiv preprint arXiv:2110.04580*, 2021.

[120] B. M. Albaba and Y. Yildiz, "Driver modeling through deep reinforcement learning and behavioral game theory," *IEEE Transactions on Control Systems Technology*, 2021.

[121] A. Sarkar and K. Czamecki, "A behavior driven approach for sampling rare event situations for autonomous vehicles," in *2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 11 2019, pp. 6407–6414.

[122] L. Sun, W. Zhan, M. Tomizuka, and A. D. Dragan, "Courteous Autonomous Cars," *IEEE International Conference on Intelligent Robots and Systems*, pp. 663–670, 2018.

[123] B. Toghi, R. Valiente, D. Sadigh, R. Pedarsani, and Y. P. Fallah, "Social coordination and altruism in autonomous driving," *IEEE Transactions on Intelligent Transportation Systems*, vol. 23, no. 12, pp. 24 791–24 804, 2022.

[124] T. H. Ho and X. Su, "A dynamic level-K model in sequential games," *Management Science*, vol. 59, no. 2, pp. 452–469, 2 2013.

[125] K. Zhang, Z. Yang, and T. Başar, "Multi-agent reinforcement learning: A selective overview of theories and algorithms," *arXiv preprint arXiv:1911.10635*, 2019.

[126] R. Nagel, "Unraveling in guessing games: An experimental study," *The American Economic Review*, vol. 85, no. 5, pp. 1313–1326, 1995.

[127] B. M. Albaba and Y. Yildiz, "Modeling cyber-physical human systems via an interplay between reinforcement learning and game theory," *Annual Reviews in Control*, vol. 48, pp. 1–21, 2019.

[128] M. A. Costa-Gomes, V. P. Crawford, and N. Iriberri, "Comparing Models of Strategic Thinking in Van Huyck, Battalio, and Beil's Coordination Games," *Journal of the European Economic Association*, vol. 7, no. 2-3, pp. 365–376, 4 2009.

[129] C. G. McClintock and S. T. Allison, "Social Value Orientation and Helping Behavior," *Journal of Applied Social Psychology*, vol. 19, no. 4, pp. 353–362, 3 1989.

[130] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[131] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double Q-Learning," in *30th AAAI Conference on Artificial Intelligence, AAAI 2016*, 2016, pp. 2094–2100.

[132] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[133] "Roundabouts." [Online]. Available: https://www.iihs.org/topics/roundabouts

[134] R. Tian, S. Li, N. Li, I. Kolmanovsky, A. Girard, and Y. Yildiz, "Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts," in *2018 IEEE Conference on Decision and Control (CDC)*. IEEE, 2018, pp. 321–326.

[135] S. Masi, P. Xu, and P. Bonnifait, "A curvilinear decision method for two-lane roundabout crossing and its validation under realistic traffic flow," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1290–1296.

[136] "SOS - What Every Driver Must Know."

[137] M. Werling, J. Ziegler, S. Kammel, and S. Thrun, "Optimal trajectory generation for dynamic street scenarios in a frenet frame," in *2010 IEEE International Conference on Robotics and Automation*. IEEE, 2010, pp. 987–993.

[138] L. Wang, L. Sun, M. Tomizuka, and W. Zhan, "Socially-compatible behavior design of autonomous vehicles with verification on real human data," *IEEE Robotics and Automation Letters*, vol. 6, no. 2, pp. 3421–3428, 2021.

[139] G. E. Mullins, P. G. Stankiewicz, R. C. Hawthorne, and S. K. Gupta, "Adaptive generation of challenging scenarios for testing and evaluation of autonomous vehicles," *Journal of Systems and Software*, vol. 137, pp. 197–215, 2018.

[140] C. E. Rasmussen and C. K. I. Williams, *Gaussian Processes for Machine Learning*. Cambridge, Massachusetts: MIT Press, 2006.

[141] M. J. Kochenderfer and T. A. Wheeler, *Algorithms for optimization*. Mit Press, 2019.

[142] P. I. Frazier, "A tutorial on bayesian optimization," *arXiv preprint arXiv:1807.02811*, 2018.

[143] C. M. Bishop, *Pattern recognition and machine learning*. springer, 2006.

[144] T. Lattimore and C. Szepesvári, *Bandit algorithms*. Cambridge University Press, 2020.

[145] R. Agrawal, "Sample mean based index policies by o (log n) regret for the multi-armed bandit problem," *Advances in Applied Probability*, vol. 27, no. 4, pp. 1054–1078, 1995.

[146] O. Chapelle and L. Li, "An empirical evaluation of thompson sampling," *Advances in neural information processing systems*, vol. 24, 2011.

[147] X. Wang, H. Peng, S. Zhang, and K.-H. Lee, "An interaction-aware evaluation method for highly automated vehicles," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 394–401.

[148] W. Zhan, L. Sun, D. Wang, H. Shi, A. Clausse, M. Naumann, J. Kummerle, H. Konigshof, C. Stiller, A. de La Fortelle *et al.*, "Interaction dataset: An international, adversarial and cooperative motion dataset in interactive driving scenarios with semantic maps," *arXiv preprint arXiv:1910.03088*, 2019.

[149] X. Zhang, J. Tao, K. Tan, M. Torngren, J. M. G. Sanchez, M. R. Ramli, X. Tao, M. Gyllenhammar, F. Wotawa, N. Mohan *et al.*, "Finding critical scenarios for automated driving systems: A systematic mapping study," *IEEE Transactions on Software Engineering*, 2022.

[150] J.-A. Bolte, A. Bar, D. Lipinski, and T. Fingscheidt, "Towards corner case detection for autonomous driving," in *2019 IEEE Intelligent vehicles symposium (IV)*. IEEE, 2019, pp. 438–445.

[151] Z. Huang, Y.-J. Mun, X. Li, Y. Xie, N. Zhong, W. Liang, J. Geng, T. Chen, and K. Driggs-Campbell, "Hierarchical intention tracking for robust human-robot collaboration in industrial assembly tasks," *arXiv preprint arXiv:2203.09063*, 2022.

[152] L. Sun, W. Zhan, and M. Tomizuka, "Probabilistic prediction of interactive driving behavior via hierarchical inverse reinforcement learning," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2018, pp. 2111–2117.

[153] H. Zhao, J. Gao, T. Lan, C. Sun, B. Sapp, B. Varadarajan, Y. Shen, Y. Shen, Y. Chai, C. Schmid *et al.*, "Tnt: Target-driven trajectory prediction," in *Conference on Robot Learning*. PMLR, 2021, pp. 895–904.

[154] D. Sadigh, N. Landolfi, S. S. Sastry, S. A. Seshia, and A. D. Dragan, "Planning for cars that coordinate with people: leveraging effects on human actions for planning and active information gathering over human internal state," *Autonomous Robots*, vol. 42, no. 7, pp. 1405–1426, 2018.

[155] Y. Chai, B. Sapp, M. Bansal, and D. Anguelov, "Multipath: Multiple probabilistic anchor trajectory hypotheses for behavior prediction," *arXiv preprint arXiv:1910.05449*, 2019.

[156] R. Tian, L. Sun, M. Tomizuka, and D. Isele, "Anytime game-theoretic planning with active reasoning about humans' latent states for human-centered robots," in *2021 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2021, pp. 4509–4515.

[157] Y. Che, A. M. Okamura, and D. Sadigh, "Efficient and trustworthy social navigation via explicit and implicit robot–human communication," *IEEE Transactions on Robotics*, vol. 36, no. 3, pp. 692–707, 2020.

[158] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE mobile computing and communications review*, vol. 5, no. 1, pp. 3–55, 2001.

[159] B. D. Ziebart, A. L. Maas, J. A. Bagnell, A. K. Dey *et al.*, "Maximum entropy inverse reinforcement learning." in *Aaai*, vol. 8. Chicago, IL, USA, 2008, pp. 1433–1438.

[160] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, no. 3, pp. 279–292, 1992.

[161] "Model predictive control toolbox." [Online]. Available: https://www.mathworks.com/products/model-predictive-control.html

[162] S. Xu, R. Zidek, Z. Cao, P. Lu, X. Wang, B. Li, and H. Peng, "System and experiments of model-driven motion planning and control for autonomous vehicles," *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 2021.

[163] K. Kreutz and J. Eggert, "Analysis of the generalized intelligent driver model (gidm) for merging situations," in *2021 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2021, pp. 34–41.

[164] L. Kocis and W. J. Whiten, "Computational investigations of low-discrepancy sequences," *ACM Transactions on Mathematical Software (TOMS)*, vol. 23, no. 2, pp. 266–294, 1997.

[165] S. Xu and H. Peng, "Design, Analysis, and Experiments of Preview Path Tracking Control for Autonomous Vehicles," *IEEE Transactions on Intelligent Transportation Systems*, vol. PP, pp. 1–11, 2019.

[166] S. Xu, H. Peng, Z. Song, K. Chen, and Y. Tang, "Accurate and Smooth Speed Control for an Autonomous Vehicle," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, vol. 2018-June. IEEE, 6 2018, pp. 1976–1982.

[167] A. Vahidi and A. Sciarretta, "Energy saving potentials of connected and automated vehicles," *Transportation Research Part C: Emerging Technologies*, vol. 95, pp. 822–843, 2018.

[168] R. Rajamani, *Vehicle dynamics and control*. Springer Science & Business Media, 2011.

[169] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*. PMLR, 2017, pp. 1–16.

[170] "Scenariorunner for carla: Traffic scenario definition and execution engine." [Online]. Available: https://github.com/carla-simulator/scenario_runner

[171] Y. Zhong, Z. Cao, M. Zhu, X. Wang, D. Yang, and H. Peng, "Clap: Cloud-and-learning-compatible autonomous driving platform," in *2020 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, 2020, pp. 1450–1456.

[172] S. Levine and V. Koltun, "Continuous inverse optimal control with locally optimal examples," *arXiv preprint arXiv:1206.4617*, 2012.

[173] Z. Yang, R. Zhang, and H. X. Liu, "A hierarchical behavior prediction framework at signalized intersections," in *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*. IEEE, 2021, pp. 515–521.

[174] M. Kuderer, S. Gulati, and W. Burgard, "Learning driving styles for autonomous vehicles from demonstration," in *2015 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2015, pp. 2641–2646.

[175] L. Sun, Z. Wu, H. Ma, and M. Tomizuka, "Expressing diverse human driving behavior with probabilistic rewards and online inference," in *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2020, pp. 2020–2026.

[176] R. Tian, M. Tomizuka, and L. Sun, "Learning human rewards by inferring their latent intelligence levels in multi-agent games: A theory-of-mind approach with application to driving data," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2021, pp. 4560–4567.

[177] C. Fiedler, C. W. Scherer, and S. Trimpe, "Practical and rigorous uncertainty bounds for gaussian process regression," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 35, no. 8, 2021, pp. 7439–7447.

[178] H. Peng, S. Zhang, J. K. Lenneman, and E. Pulver, "Roadmanship systems and methods," Sep. 29 2022, uS Patent App. 17/210,038.