Dynamic-Occlusion-Aware Risk Identification for Autonomous Vehicles Using Hypergames

by

Maximilian Kahn

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2021

Author's Declaration

This thesis consists of material all of which I authored or co-authored: see Statement of Contributions included in the thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Statement of Contributions

This research was performed as a collaboration between members of the Waterloo Intelligent Systems Engineering Lab. Maximilian Kahn wrote the programs for performing 2D raycasting, SOV injection, extracting speed distributions from naturalistic data, and instantiating the level-0 and level-1 hypergame simulations. He was also responsible for running experiments and analyzing the results. Atrisha Sarkar wrote the programs for extracting subject/relevant vehicles from naturalistic data, trajectory generation, solving the hierarchical game using Nash equilibrium, as well as developed the DOR measure. She was also the one to propose using hypergames as a way to incorporate occlusion in game-theoretic models. Professor Krzysztof Czarnecki developed the 4-leveled (S0-S3) severity model.

Abstract

A particular challenge for both autonomous vehicles (AV) and human drivers is dealing with risk associated with dynamic occlusion, i.e., occlusion caused by other vehicles in traffic. In order to overcome this challenge, we use the theory of hypergames to develop a novel dynamic-occlusion risk measure (DOR). We use DOR to evaluate the safety of strategic planners, a type of AV behaviour planner that reasons over the assumptions other road users have of each other. We also present a method for augmenting naturalistic driving data to artificially generate occlusion situations. Combining our risk identification and occlusion generation methods, we are able to discover occlusion-caused collisions (OCC), which rarely occur in naturalistic driving data. Using our method we are able to increase the number of dynamic-occlusion situations in naturalistic data by a factor of 70, which allows us to increase the number of OCCs we can discover in naturalistic data by a factor of 40. We show that the generated OCCs are realistic and cover a diverse range of configurations. We then characterize the nature of OCCs at intersections by presenting an OCC taxonomy, which categorizes OCCs based on if they are left-turning or right-turning situations, and if they are reveal or tagging-on situations. Finally, in order to analyze the impact of collisions, we perform a severity analysis, where we find that the majority of OCCs result in high-impact collisions, demonstrating the need to evaluate AVs under occlusion situations before they can be released for commercial use.

Acknowledgements

This work would not have been possible without the tremendous amount of support I received from Atrisha Sarkar. In fact, much of this work was built on top of Atrisha's research into hierarchical games. She has been a mentor and a friend, and her wealth of knowledge on game theory and trajectory generation (to name just a few areas) has been hugely helpful. My supervisor, Professor Krzysztof Czarnecki, has also been an invaluable source of thoughtful suggestions and academic knowledge, and our discussions together have greatly shaped this work.

While this thesis is the culmination of 11 months of work, it represents only half the time I have spent as a Master's student in the Waterloo Intelligent Systems Engineering (WISE) lab, and so I would like to extend my gratitude to all the WISE lab members whom I had the pleasure of working with. In particular, I would like to thank Dr. Michal Antkiewicz for all the help he has given me over the years, especially for showing me the ropes when I first joined the lab in the summer of 2019.

Dedication

This is dedicated to my family, who have put up with me living at home, mooching off their food for the past year, while I worked on my thesis. I would also like to dedicate this to my friends, Claire and Kristophe. May our ambitions of one day starting a commune come true.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Imagine you are in the left-turn lane in a busy intersection. There is a vehicle ahead of you that has started their left turn, and as a result is *occluding* your view of oncoming vehicles from the opposite side of the intersection. You have two options, you can either wait to see if there are any oncoming vehicles before making your turn, or, you can trust that since the vehicle in front of you is making a turn, the way should be clear for you too. Being a cautious driver you decide to wait and see if there are any oncoming vehicles. Just as the vehicle in front of you finishes its turn, an oncoming vehicle, which was previously occluded, drives past you at 50 km/h. Had you decided to follow the vehicle in front of you into the intersection you would have been involved in a front-to-front collision.

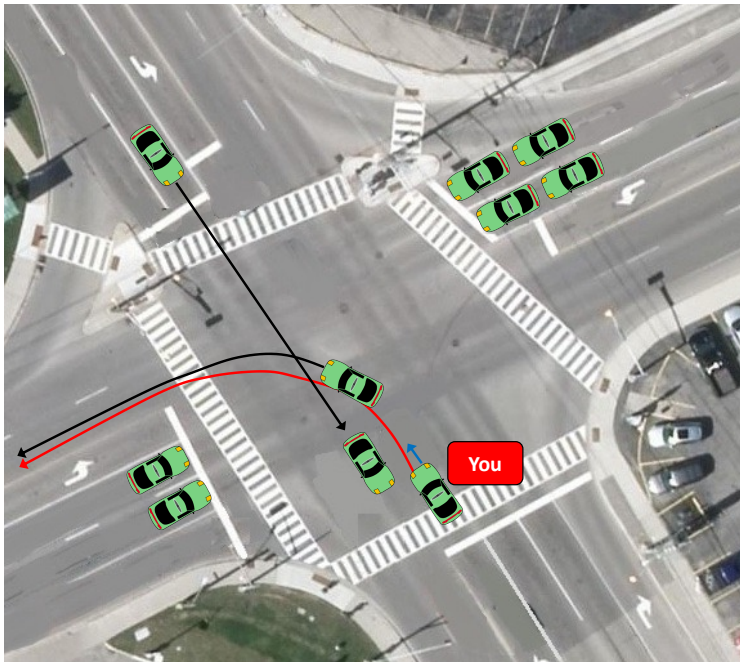


Figure 1.1: You are following a lead vehicle and cannot see the oncoming vehicle. Your blue trajectory represents the safe action, to wait and see if there are any oncoming vehicles before proceeding with your turn. Your red trajectory is the unsafe action, in this case resulting in a collision with the oncoming vehicle.

Dynamic occlusion, i.e., obstructed views caused by other vehicles in traffic, can occur in any traffic environment, and can result in both low- and high-severity collisions. With that being said, there are certain road geometries, such as four-way intersections, where dynamic occlusion occurs more frequently and is more likely to result in high-severity collisions. Dynamic occlusion at intersections is a particular problem given that there is ample opportunity for vehicles to block each other's view, and since vehicles are often moving along paths that intersect, there is also ample opportunity for head-on collisions. Occlusion is a critical factor in causing collisions at intersections with the National Motor Vehicle Crash Causation Survey reporting that 7.8% of all intersection-related collisions were caused by the driver's incorrect decision to turn with an obstructed view of traffic [26].

One of the primary goals of autonomous driving is to avoid making the same mistakes as human drivers. However, autonomous vehicle (AV) development has not gone without its own challenges. After an array of highly-publicized accidents involving AVs [27, 111, 23],

many people are justifiably wary about the safety capabilities of this new technology [88]. AVs have the potential to make the world a better place by optimizing traffic flow, giving those who are unable to drive the ability to commute on their own, and reducing the number of drunk-driving accidents, to name just a few examples. However, in order for AVs to become widely used and trusted by the public, we have to first demonstrate they are safe. To that end, we focus on the challenging subproblem of navigating intersections in the presence of dynamic occlusion.

We focus on *dynamic* occlusion because unlike static occlusion (i.e., obstructed views caused by buildings, trees, and parked cars), where information about road structures can be leveraged from high-definition (HD) maps to predict where occlusion may occur, dynamic occlusion is fundamentally transient in nature, making it more difficult to predict. The transient nature of dynamic occlusion also means that the scope of the problem is much larger. Dynamic occlusion occurs when two or more vehicles are visually obstructed from each other by at least one occluding vehicle, and as a result, there are an infinite number of ways in which dynamic occlusion can occur. As a result, AV safety evaluations that only account for traffic situations found in existing naturalistic driving data will not achieve complete coverage of all dynamic occlusion situations an AV will face in its operational lifetime.

To address this problem at scale, we present a situation-based accelerated evaluation method which is able to efficiently generate realistic and critical dynamic-occlusion situations that go beyond the set observed in existing naturalistic driving data. Using a novel dynamic-occlusion risk (DOR) measure, we are able to estimate to what extent occlusion contributes to driving risk. Along with the presentation of our methods, we formulate the following guiding research questions:

RQ1: Are there patterns to how dynamic occlusion emerges at an intersection?

RQ2: Are there certain locations in an intersection where dynamic occlusion tends to occur?

RQ3: How severe are collisions caused by dynamic occlusion?

RQ4: How long does dynamic occlusion have to last to result in a critical situation?

Therefore, we expand on previous occlusion-aware risk estimation work by incorporating game-theoretic models of the traffic environment into our vehicle simulation, which allows us to test multiple AVs at once, all of whom use a multi-agent view to plan manouevres. Each AV is controlled using a *strategic planner*, which reformulates driving as a game-theoretic problem where traffic participants are players in a game [71, 106, 107, 72, 42, 36].

Each agent has knowledge of the actions and preferences available to all other agents and therefore can reason over their available strategies in order to solve for an optimal strategy—optimality being defined by a game's solution concept, one such example being Nash equilibrium.

However, a problem arises when evaluating strategic planners using standard game-theoretic models, namely, game theory assumes there is *common knowledge* shared amongst the agents in the game, such as each agent knowing the rules of the game as well as knowing each other agent's preferences. Implicit in the common knowledge assumption is the idea that each agent is aware of every other agent in the game. However, this is not the case in traffic situations with occlusion. In these kinds of situations, drivers may all be included in the same traffic game, but due to occlusion, may not be aware of each other. Therefore, a novel contribution of this thesis is to use the theory of hypergames as an extension to standard game theory, which allows each agent to have their own subjective view of the game being played, and as a result allows for occlusion to be incorporated into the game-theoretic traffic models we use.

More broadly the research contributions of this thesis are:

- a method for efficiently generating dynamic occlusion situations from naturalistic traffic data,
- the novel use of hypergame theory as an extension to standard game theory allowing for the incorporation of occlusion in traffic games,
- a novel dynamic-occlusion risk (DOR) measure, which measures to what extent occlusion contributes to driving risk,
- a taxonomy describing the different dynamic-occlusion configurations that can lead to high-risk situations,
- empirical insights into the nature of dynamic-occlusion situations, including a severity analysis, how long occlusion tends to last in high-risk occlusion situations, and where occluding vehicles tend to be positioned.

This thesis is divided into seven chapters. In chapter 2, we cover relevant literature for this work in the areas of occlusion-aware risk estimation and black-box safety validation of AVs. Following this, in chapter 3 we describe in more detail both how we use game theory as a model for AV decision-making, as well as how we incorporate dynamic occlusion into our approach. In chapter 4 we outline the whole workflow, with a running example, for

how our situation-based accelerated testing is performed. Following that, we describe our results in chapter 5, comparing the variety and number of occlusion situations we can generate using our method to what can be extracted from existing data. In chapter 6, we outline threats to the validity of our findings. Finally, we conclude in chapter 7 by summarizing our contributions.

Chapter 2

Literature Review

Our proposed approach can be divided into two components. The first component is an accelerated situation-based testing method where we generate critical and realistic occlusion situations not found in existing data. The second component is identifying risky occlusion situations. Our accelerated situation-based testing method closely resembles *falsification*[1], a technique used in black-box testing, where the task is to identify stimuli—that could come from other vehicles, pedestrians, etc.—that induce failures in the AV. In this work, the stimuli are occlusion situations and the failures are collisions (caused by occlusion). Therefore, we first provide a brief overview of black-box testing, describing where it is situated in the broad area of AV safety validation.

We then review literature related to three common techniques used in black-box testing, falsification (used in this work), most-likely failure analysis, and failure probability estimation, comparing the benefits and drawbacks of each technique. The second component of our approach is performing risk estimation on the occlusion situations we generate. Therefore, we also cover related works in the area of occlusion-aware risk estimation.

2.1 Safety Validation and AV Black-box Testing

AV safety validation is a process that seeks to ensure the system-under-test (SUT) behaves according to a given safety specification in a driving environment. Safety specifications

[1] In the sense that we treat the strategic planners we test as black boxes. However, we note that strategic planners are white-box planners in actuality, we simply do not exploit this property in this work.

typically include conditions like avoiding collisions, but can also more broadly include definitions for unsafe situations such as a violation of minimum following distance or executing a turn while oncoming vehicles are occluded. Safety validation typically occurs in three different modes, real-world testing, simulation testing, and mixed-reality testing [8].

Solely relying on real-world testing is not a feasible way to approach safety validation as setting up physical test drives is costly and time consuming [8]. Crucially, an AV can drive for thousands of miles on public roads and not encounter any adverse event, since these events (e.g., collisions) are rare. In order to show that an AV is safe, it would need to drive for hundreds of millions, up to hundreds of billions of miles on public roads [53]. Even if these miles could be driven—perhaps over a large fleet of AVs—this validation would need to happen every time the AV receives a significant software update. A similar problem emerges for basic simulation testing. While one could operate a fleet of AVs in a simulated driving environment, the requirement of driving hundreds of millions of miles is computationally infeasible for many AV developers.

Fortunately, specialized approaches have been developed to lower the time and cost of safety validation. These approaches broadly fall into two categories, white-box and black-box testing. In white-box testing, knowledge of the internal workings of the SUT can be used to develop a formal proof of safety. For example, model checking [28, 55] and automated theorem proving [89, 37, 98] use mathematical models of the SUT to prove the existence or absence of safety specification violations. White-box testing has been used for adaptive cruise control (ACC) systems [75, 73], but in general these approaches, which must consider all execution possibilities, struggle to scale to larger problem spaces; furthermore, many AV planners employ the use of deep neural networks and are thus not easily interpretable.

In contrast, black-box testing does not assume the internals of the SUT are known. Like its name suggests, black-box testing assumes the SUT is a "black box" which accepts input, performs a hidden calculation on that input, and returns the output. This generality allows black-box approaches to be applied to a much larger class of systems. Unlike white-box testing where the goal is to prove that the SUT will not violate the safety specification, black-box testing tries to *falsify* the SUT by generating counterexamples showing where the SUT fails.

Broadly speaking, black-box testing is made up of the SUT, the environment in which the SUT operates, and an adversary that generates *disturbances* with the goal of causing the SUT to fail. Corso et al. separate black-box testing into three distinct tasks [33]. These are, falsification (i.e., finding situations where the SUT fails to meet the safety specification), most-likely failure analysis (i.e., finding the most likely cases where the SUT fails to meet

the safety specification), and failure probability estimation (i.e., estimating the probability that the SUT will fail to meet the safety specification).

2.1.1 Falsification

The goal of falsification is to generate examples where the SUT violates the safety specification. Abbas et al. [1] propose a falsification framework that tries to induce a failure in the SUT by both changing the position and velocity of the SUT and other road participants and by varying the number of vehicles in the scenario. Their SUT implementation also includes object detection, which allows them to induce perception errors by varying the weather and lighting conditions. The simulation environment is set in the video game, GTA V. The authors raise the apt question of how failures found in simulation translate into real world failures. This is a challenge for simulation-based testing as a whole and is something we, in this thesis' work, also contend with (we address how realistic our synthetic occlusion situations are in chapter 5). Abbas et al. focus their attention on showing that object detection in simulation translates well to the real world by demonstrating in their results that the object detector's performance in simulation was observed to be a lower bound on its performance, so it would perform at least as well in the real-world as in simulation. This allows them to translate their safety validation results from simulation to the real world.

Domain knowledge can be incorporated in order to increase the efficiency of finding safety specification violations. Koschi et al. [61] develop two falsification methods, forward and backward search, that use rapidly-exploring random trees to search for failure cases for an ACC system. They define an *unsafe state* to be a state where a collision will occur if the lead vehicle fully brakes and use these unsafe states to guide the SUT to failure cases. Indeed, we employ a similar tactic in this thesis—in order to generate occlusion-caused collisions (OCC), we inject synthetic vehicles that cause occlusion and use these artificial occlusion scenarios to lead us to occlusion-caused collisions.

Before a failure case can occur, the driving situation must first become safety compromised. Falsification approaches will often use this fact to aid in the search for failure cases. For example, Althoff and Lutz [5] take safe traffic situations, and by manipulating the positions of the traffic participants, create dangerous situations—i.e., situations where there is a small space of actions that can be used to avoid a collision. A challenge with approaches that manipulate the positions of the traffic participants is to make sure the transformed situations are still realistic. Since our own work uses synthetically injected vehicles to generate dangerous occlusion scenarios, this is a problem that we also had to overcome.

2.1.2 Most-Likely Failure Analysis

Unlike falsification, which only seeks to find failure cases without any other constraint, most-likely failure analysis goes one step further and seeks to find failure cases with the highest occurrence frequency. This provides developers with more information on how to most effectively implement changes to make AVs safer. Furthermore, all failure cases are at most as likely as the most-likely failure case, therefore this provides a bound on the likelihood of failures.

A prominent approach in this area is Adaptive Stress Testing (AST) [59, 31]. Unlike traditional falsification approaches where the goal is to find a set of cases where the SUT violates the safety specification, AST instead learns an *adversarial policy* that causes the SUT to fail. In other words, AST uses an adversary, typically a reinforcement learning agent, that learns what kinds of disturbances to both the environment and SUT cause the SUT to fail. Koren et al. [59] use AST in a situation where the SUT is approaching a crosswalk where two pedestrians are crossing. The adversary has six available actions to disturb the environment and cause a failure. It can control each pedestrian's longitudinal and lateral acceleration and the amount of noise in each pedestrian's position and acceleration measurement. They compare deep reinforcement learning (DRL) to a Monte Carlo tree (MCTS) search method and show that DRL is able to use information from simulation more efficiently and therefore can generate more failure cases than MCTS. However, both of these methods are not without their drawbacks. In particular, they found that a large portion of the failure cases were not caused by the SUT but by the pedestrians, who would run and jump into the SUT (which is not typical pedestrian behaviour!).

The problem is that they had setup the objective function such that it rewards the adversary for any kind of collision, regardless if the collision is caused by the SUT or not. In a follow-up paper, Corso et al. [31] handcrafted a new reward function that takes into account the notion of *blame*, so that the adversary is only rewarded when the SUT causes a collision. They use *Responsibility-Sensitive Safety* [100]—a mathematical formulation of blame, which encodes notions like minimum safe following distance and proper response to adverse driving situations—as a model for blame in their updated reward function. In order to generate varied failure cases they also include a dissimilarity metric in the reward function to encourage the adversary to learn to induce different collision configurations.

Qin et al. [91] present an augmentation to the AST reward function. They use Signal Temporal Logic (STL) [76] to mathematically formalize the safety specification for the SUT. Instead of manually crafting a reward function, they reward the adversary based on whether it generated failure cases within the STL constraints or not. Like all simulation-based approaches (our method included), AST suffers from the "garbage in garbage out"

problem, i.e., AST requires accurate traffic participant behavior, environments, and driving scenarios in order to generate realistic failure cases.

2.1.3 Failure Probability Estimation

Unlike the previous two tasks which focus on finding particular failure cases, failure probability estimation instead focuses on estimating the probability that the AV will violate its safety specification. Works in this area use the notation $X \sim P_0$ to denote a realization of sampling from a base distribution that models standard traffic behaviour (e.g., X can be parameters such as the weather, lighting, time of day, behaviour of other traffic participants, etc.). Here, the safety specification $f : \mathcal{X} \to \mathbb{R}$, maps these parameters to a safety value such that low values of $f(x)$ correspond to a low safety score. The goal of failure probability estimation is then,

$$p_\gamma := \mathbb{P}_0(f(X) \leq \gamma). \tag{2.1}$$

In other words, to estimate p_γ, which is the probability that the AV will perform below some safety threshold, γ. As noted by Kalra and Paddock [53], evaluating this probability in the context of autonomous vehicle driving safety becomes a rare event analysis problem and so simply sampling from P_0 to compute p_γ requires a prohibitively large amount of time. One way to solve this problem is to use importance sampling techniques. The key idea being to modify the base distribution, P_0, in such a way that failures can be sampled more frequently. Then, the results are modified to reflect the changes made to P_0, in order to compute the unbiased probability of failure. The goal of importance sampling is to construct a biased distribution which minimizes the number of samples needed to compute an accurate estimate of p_γ.

O'Kelly et al. [85] use cross-entropy importance sampling, a parametric importance sampling method, in order to estimate p_γ. The idea being to iteratively get closer to the optimal importance sampling distribution by constructing intermediate distributions, P_{θ_k}. The parameters of the k^{th} distribution, θ_k, are obtained based on samples taken from the previous distribution, $P_{\theta_{k-1}}$. Over time, regions of $\mathbb{X}$ where $f(x)$ is low (i.e., unsafe) will be more heavily weighted resulting in a closer approximation of the optimal importance sampling distribution. Their SUT is a black-box AV with a deep-learning-based perception system. They test the SUT in a multi-lane highway environment modeled after the I-80 in the United States, where the goal is to safely traverse the 2 km stretch of highway modeled in the simulator. The other vehicles are trained using generative adversarial

imitation learning on the NGSim dataset [4]. The simulation parameters, X, are the initial positions, orientations, and velocities for each vehicle, as well as a set of 404 weights which represent the last layer of the neural network used to control the other vehicles. The safety specification, $f(x)$, is defined as the minimum time-to-collision (TTC), from the centre of the SUT, across the simulation rollout. They find that the cross-entropy method was able to sample rare failure events with a speedup of 18 times that of naive Monte Carlo, as well produced importance sampling estimators with a lower variance.

A drawback of the cross-entropy method is that it can be numerically unstable. The authors get around this problem by limiting the search space to regions that are tractable. They found that the importance sampler would induce violations of the minimum TTC by shifting the other vehicles such that the SUT was surrounded by other vehicles while driving, as well as increasing the initial velocity of trailing vehicles. While situations like this are dangerous, they are not necessarily a result of any failure on the AV's part. Future work could include incorporating blame into the safety specification to specifically search for situations where the SUT is at fault. They also found that, as a result of defining TTC from the centre of the SUT rather than from the closest point along the SUT's edge, the induced failure cases frequently were sideswipe collisions as opposed to front-to-front or front-to-back collisions. This being a good reminder to be careful when designing $f(x)$ as how it is designed can have a direct impact on the safety results.

Norden et al. [81] use adaptive multi-level splitting (AMS), a non-parametric importance sampling method, to compute p_γ. The key idea being to decompose the rare event probability, p_γ, into the product of non-rare probabilities that can be computed using Markov chain Monte Carlo (MCMC). Instead of immediately using the threshold level, γ, intermediate threshold levels are used, $\infty := L_0 < L_1 \cdots < L_k := \gamma$, eventually converging to γ. This is in contrast to parametric adaptive importance sampling methods (e.g., the cross-entropy method [85]), which can have difficulty with AV-related tasks since the problem space is often high dimensional, failure regions are discontinuous, and likelihood ratios are numerically unstable. AMS solves this by not needing to generate models for the failure regions(s) nor compute likelihood ratios. The drawback of AMS is that the types of failure cases it finds are limited by the number of samples drawn and the generative properties of the MCMC sampler used. The SUT in this work is a proprietary SAE-level 2 [29] driver assistance system, which uses camera images to reconstruct an image of the environment. The goal is to safely navigate 1 km of highway road, in the Carla simulator, in the presence of 5 other vehicles (which are trained using imitation learning). The authors use the same simulation parameters as [85] with the addition of four parameters controlling precipitation on the ground, the altitude angle of the sun, cloudiness, and precipitation in the air. They define $f(x)$ as the minimum TTC but with respect to each vehicle's bounding box rather

than the centre of each vehicle.

They found that when searching for dangerous rare events, i.e., $\gamma \leq 0.1$ seconds, AMS is able to sample three orders of magnitude more failure cases than naive Monte Carlo. Given $\gamma = 0.1$ seconds, they found that the failure rate was roughly 1 in 10^4 simulation rollouts. However, for non-rare events (i.e., $\gamma \geq 0.8$ seconds) naive Monte Carlo outperforms AMS. In fact, naive Monte Carlo also outperforms the cross-entropy method for non-rare events [85], suggesting that importance sampling methods should be used only for rare event sampling. Using PCA dimensionality reduction, the authors categorize failures into four modes, collisions involving direct sunlight, or heavy rain, and collisions where there is either a low relative velocity, between vehicles, or high relative velocity. They further note that collisions are much more likely in heavy rain than in direct sunlight, suggesting that the perception system does not perform well on scenes with overcast and rainy weather. However, while errors in the perception system is a contributing factor to failures in the SUT, it is not solely responsible. It is the interaction between the perception system and the motion planner that causes the SUT to fail. They give the example where there is direct sunlight and the perception system is uncertain of the exact position of the vehicle in front of the SUT. The motion planner knows that the perception system is uncertain, but instead of planning a conservative trajectory that accounts for the perception system's uncertainty in the lead vehicle's position, the motion planner plans a trajectory that is too aggressive resulting in the SUT colliding with the leading vehicle. This highlights that black-box testing should ideally include whole system testing. It is not enough to only test individual components of the SUT since errors often accumulate around the interactions between components.

Sinha et al. [102] build on the work in [81] by employing "neural bridging" to outperform AMS and naive Monte Carlo sampling. Their proposed technique is closely related to AMS with the major difference being how the intermediate distributions P_k are computed. AMS uses "hard barriers" through the use of indicator functions while bridge sampling uses "soft barriers" through the use of smoother exponential functions. Furthermore, where AMS iteratively steers the probability towards p_γ through the use of the intermediate levels, L_0 to L_k, neural bridging uses a gradient-based approach which avoids inefficient random-walk behaviour. The authors test their method on a variety of tasks including OpenAI's CarRacing environment [86], where they compare two neural-network-based approaches; specifically, they compare the probability that either one achieves a score lower than 150 (this is a rare event as both approaches, on average, achieve scores around 900). The parameter search space, P_0, in this case is characterized by 12 checkpoints that define the geometry of the racing track. The task then, is to compute the probability that randomly sampling a track will result in a score below 150. The authors find that naive Monte

Carlo is not able to distinguish the failure rates between the two SUTs because of its high uncertainty, whereas the authors' method clearly shows that one of the methods has a lower probability of failure than the other.

Sarkar and Czarnecki [94] develop a novel driving behaviour model based on the theory of bounded rationality to model traffic behaviour for evaluation of autonomous vehicles. One of the key ideas of their work is incorporating the fact that human drivers often behave sub-optimally. Their work, like ours, generate driving policies based on safety and progress-to-goal utilities. However, where we use a pure utility maximization principle (i.e., where agents choose the optimal strategy 100% of the time), the authors instead use a bounded rationality principle where each vehicle in their simulation may choose a sub-optimal strategy with some probability. They use their behaviour model to generate a range of driver behaviours and estimate the probability of a near-crash situation (i.e., a rare event) under these behaviours. They find that their rare event sampling approach outperforms both naive Monte Carlo and Cross Entropy sampling in terms of sampling rare events.

In general, failure probability estimation is a powerful tool for validating autonomous systems, providing information on the overall behaviour of the SUT. It is clear that such approaches will most likely be a necessary component in the safety certification pipeline for autonomous vehicles. However, an added difficulty of these approaches is the identification of P_0. In order to carry out failure probability estimation, the reviewed works assume that P_0 is known. However, this assumption must not be overlooked when performing safety validation as the effectiveness of these approaches is contingent on P_0 being an accurate representation of what goes on in the real world. If P_0 does not closely replicate what is observed in reality, the results from failure probability estimation could lead to overconfidence in the safety level of the SUT.

2.2 Occlusion-Aware Risk Estimation

In addition to rare-event generation, we also review related work in occlusion-aware risk estimation—with a specific focus on works that incorporate traffic intersections in their analysis. While some of the works we review here only propose a risk estimation method, many present an occlusion-aware planning method which incorporates risk from occlusion into the planning process. Occlusion-aware risk estimation fundamentally revolves around dealing with uncertainty, specifically the potential existence of occluded vehicles. As such, there are primarily two paradigms for navigating traffic situations with occlusion, these being *probabilistic* and *set-based* methods.

2.2.1 Probabilistic Approaches

What makes autonomous driving a particular challenge is the inherent uncertainty of other traffic participants' intentions and the uncertainty of how occluded areas of the driving environment will evolve over time. Probabilistic methods provide a framework to both reason about uncertainty while also performing approximately-optimal decision making. One such approach is the Partially Observable Markov Decision Process (POMDP) [19, 74, 20, 105, 97], which is a generalization of the Markov Decision Process (MDP), but where the agent cannot observe the true environment state (e.g., in the case of AVs, because of noisy sensor measurements or because of occlusions).

Bouton et al. [19] use a POMDP planner to navigate a blind crosswalk (i.e., a crosswalk with static occluding objects) and blind unsignalized T-intersection. A challenge with POMDPs is that they scale exponentially with the number of agents being modeled and quickly become computationally infeasible to solve. To get around this, the authors use utility fusion as a way to approximate the solution to multi-agent POMDPs. However, another drawback of POMDPs is that the state space must be discretized in order for POMDPs to be used. This results in the continuous driving environment being transformed into a relatively small subset of discrete states which often do not allow for optimal decision-making. To get around this, Brechtel et al. [20] use a continuous POMDP solver that learns a suitable state representation that depends on the driving situation. Lin et al. [74] propose a POMDP approach to handle turning at intersections under static and dynamic occlusion. Where prior works assumed that the number of vehicles on the road and their positions are known quantities (which is not the case in many real traffic situations), the authors here do not. To get around this, the authors make the worst case assumption that vehicles are always present in occluded regions. In other words, they inject virtual vehicles at the vision boundaries of the occluded areas which the AV's planner incorporates into its decision making. However, simply injecting these vehicles in these worst-case configurations results in a deadlock, i.e., the AV is unable to move as it perpetually anticipates a vehicle emerging in the next second. To get around this, the authors created a priority list for different actions, prioritizing acceleration over deceleration, which resulted in the AV slowly creeping forward even if it believes a vehicle could emerge from the occluded region in the next moment.

Narksri et al. [80] propose another solution to the deadlock problem. They use a particle-filter-based visibility-dependent behavior model, which allows the planner to reason about how the occluded vehicle will react to the AV once both vehicles are no longer occluded. Similar to [80], Hubmann et al. [50] note that handling the general possibility of an occluded vehicle appearing at any time analytically is infeasible, since the worst-case

position of such a vehicle depends on the AV's trajectory, which has yet to be found. Furthermore, simply limiting the longitudinal velocity of the AV such that it can perform an emergency stop if an occluded vehicle were to suddenly emerge often leads to the deadlock problem. Therefore, they solve this problem by simulating the future field-of-view (FOV) of the AV, and incorporating both static and dynamic occluding objects in the simulation. Potential occluded vehicles are then computed by mapping the future FOV onto the lanes of a topological map.

Occlusion-aware deep reinforcement learning (DRL) [51, 54] has also been used to safely navigate blind unsignalized intersections. While DRL can outperform simplistic rule-based approaches (such as a TTC rule-based approach [51]) it is difficult to evaluate how safe DRL would prove to be in the real world since DRL struggles to adapt to unseen scenarios.

Other probabilistic approaches incorporate a risk metric in order to safely navigate occluded spaces. McGill et al. [77], propose a real-time probabilistic risk assessment tool, which incorporates cross traffic, sensor errors and driver attentiveness in its risk calculation. Lee et al. [66] generate an occlusion-aware collision risk estimate by first predicting the point where an occluded vehicle could emerge and then predicting the future path of the vehicle, checking to see under what speed conditions the path would intersect with the AV's path. Instead of predicting potential future paths, Yu et al. [113] use a particle-based approach, relying on road geometry and a motion model to populate the intersection with particles representing the potential future positions of occluded vehicles. The more particles an area has the higher the occlusion risk.

A drawback of probabilistic methods as a whole is that they are fundamentally approximative, i.e., they do not provide safety guarantees. Using set-based methods on the other hand, allows one to generate collision-free trajectories, even in the presence of occlusion.

2.2.2 Set-based Approaches

The key idea behind set-based approaches is to over-approximate the occupancy predictions of other observable and unobservable traffic participants. Fail-safe trajectories can then be generated by ensuring they do not intersect with the over-approximated occupancies of other traffic participants. Set-based methods cannot guarantee that the AV will not be involved in an accident because that guarantee requires that every other traffic participant behaves safely as well. What they can do is guarantee that—under the assumption that AV localization and object detection is working properly—the AV cannot be *blamed* for causing the accident (the notion of blame in the driving context is formally described in [100]).

Orzechowski et al. [87] propose a reachable-set approach using a Kamm's circle to outline each vehicle's laterally reachable set and each vehicle's maximum velocity for its longitudinally reachable set. They test their approach on three urban scenarios where occlusion is caused by static and dynamic objects. While this approach can ensure fail-safe trajectories, a drawback of being over-approximative is that the planner can end up being too conservative, resulting in the AV being unable to progress. Another consideration of such an approach is the trade off between comfort and safety since some fail-safe trajectories require higher jerk values in order to ensure there are no intersections between the reachable sets of both the AV and other traffic participants.

Koschi and Althoff [60] extend the previous work by incorporating virtual pedestrians and cyclists along with virtual vehicles, which are injected at the occlusion boundary edge. They then over-approximate the reachable sets for these participants in order to create fail-safe trajectories. A challenge with this approach is to tune the parameters of the model—such as the maximum velocity and acceleration the AV assumes other traffic participants can perform—such that the parameters cover the range of behaviors other traffic participants may exhibit while not being too conservative so as to prevent the AV from making progress. Unlike previous work where evaluation is only done in a low-fidelity simulation environment, the authors tested their approach on public roads using a BMW series 7 test vehicle.

A distinction between our work and the occlusion-aware risk estimation works reviewed here is the number and variety of occlusion situations tested. These works mention testing their method on 1-3 different occlusion situations, in contrast we test our method on roughly 106,000 different occlusion situations. Another important distinction is the form and use of the risk measure in our work compared to the works we review. In the works we review the risk measure typically defines regions in the intersection where there is high occlusion risk, which can be used to inform the AV's trajectory planner how best to generate trajectories in the presence of occlusion. However, in our work we use DOR, which instead measures *to what extent occlusion contributes to the overall driving risk*. A high DOR score means that the traffic situation is risky primarily due to occlusion. A low DOR score does not mean that the situation is not risky but simply that whatever risk is present is not from occlusion. Therefore, unlike the other occlusion-aware risk measures reviewed here, DOR is a categorization tool, which AV developers can use to quickly identify risky occlusion situations.

Chapter 3

Background

This chapter is split into three sections. In the first section, we describe how we use strategic planners to control AV behaviour. Each strategic planner plays a *hierarchical* traffic game, which is a combination of a high-level manoeuvre game and a low-level trajectory game. In the second section, we cover the theory of hypergames, which we use to incorporate occlusion into our traffic game models. In the final section, we describe in greater detail how dynamic occlusion occurs, as well as provide a definition for DOR, which relies on concepts covered in the hypergame section.

3.1 Hierarchical Games

Hierarchical traffic games are largely inspired by research into modelling driving as a hierarchy of tasks in the field of traffic psychology [108, 68, 79]. A particularly relevant example comes from Michon [79] who proposes a 3-leveled hierarchy for driving tasks consisting of a strategical level, tactical level, and operational level. The strategical level operates on a long time horizon, and includes the determination of the goal location, mode of transportation, and the costs and risks associated with the trip. The tactical level operates on a much shorter time horizon of typically just a few seconds and includes the determination of which driving manoeuvres should be carried out to reach the goal location (e.g., overtaking, gap closing, lane changing, turning, etc.). Finally, the operational level operates on a millisecond time horizon and involves the determination of the actuation patterns that control the vehicle such as steering and acceleration.

AVs follow a similar decomposition of driving tasks, with a high-level route planner, an intermediate-level behaviour planner, and a low-level trajectory planner [9] (with more

granular levels abstracted away for the sake of simplicity). A benefit of such an approach is that the costs[1] associated with planning can be dealt with separately at each level. On the strategical level, a *route planner* evaluates the most effective way to reach the goal location while taking into account costs associated with driver safety and comfort, such as having to wait in traffic, as well as driving in regions with higher collision rates. At the tactical level, a *behaviour planner*, generates driving manoeuvres based on the traffic environment (e.g., traffic lights, signs, positions of other vehicles, etc.). Costs and risks at this level involve ensuring that manoeuvres are chosen that provide progress along the AV's global path while also ensuring only safe and preferably comfortable manoeuvres are chosen. These manoeuvres are based on trajectories generated by a low-level *trajectory planner*. Once a trajectory is ready to be executed, it is converted into actuation commands, which are then carried out by a vehicle controller. The controller must balance executing actuation commands that match the trajectory profile while also accounting for passenger comfort, by minimizing the jerk profile and staying within comfortable acceleration bounds.

Hierarchical games take the hierarchical nature of driving tasks and apply it in a multi-agent environment. Previous works by Fisac et al. [36], and Sarkar and Czarnecki [95] use hierarchical games to model challenging multi-agent driving scenarios such as overtaking and merging on a two-lane highway [36], and negotiating manoeuvres at a busy intersection [95]. However, there are differences between their implementations. Fisac et al. [36] model multi-agent driving as a hierarchical Stackelberg game, which frames traffic games as a leader-follower interaction. In a Stackelberg game, the leader makes the first move and then each of the followers sequentially make their move. Historically, Stackelberg games have been used to model competition between businesses. For example, consider a duopoly with a dominant leading firm and a follower firm. The leading firm sets the price of the good, and after doing so the follower firm matches the price and adjusts their production accordingly. Stackelberg games work well in this context because the leader-follower distinction is easily discernible. However, in the context of driving, where roles such as leader and follower rapidly change as vehicles merge into new lanes and overtake other vehicles, modeling driving as a Stackelberg game can be difficult.

In contrast, Sarkar and Czarnecki [95] model multi-agent driving as a hierarchical simultaneous move game, which does not require any distinction to be made between leader and follower. Figure 3.1 shows both game types modelling an interaction between two vehicles at an intersection. Agent 1 is driving straight through the intersection and has two available actions, the first being to drive straight through the intersection at the speed limit (T), or decelerating to a stop and yielding to the oncoming left-turning vehicle (D).

[1] Where the cost is the inverse of the utility value.

The left-turning vehicle, agent 2, has two available actions, the first being to wait for agent 1 to pass (W), or to proceed with its left turn (P).

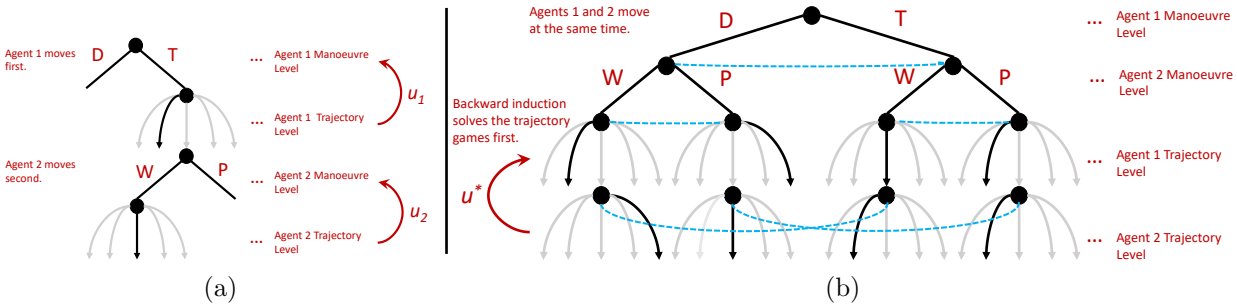


Figure 3.1: (a) A Stackelberg game between two vehicles. Agent 1 is the leader and therefore moves first. (b) A simultaneous-move game between two vehicles, the dotted blue lines indicate the information sets. Both game types are two-leveled hierarchical games, including both a trajectory level and a manoeuvre level. The red arrows indicate that utilities are determined in the trajectory-level game and then are used in the manoeuvre-level game. We show a single trajectory (highlighted in black) being selected from a set of five potential trajectories simply for the sake of simplicity. In our work, we generate 50 trajectories for each manoeuvre, where each trajectory has a time horizon of 6 s.

In the Stackelberg game shown in figure 3.1a, the leader, agent 1, chooses action T. Agent 1 then generates a trajectory to execute the manoeuvre. There are a broad spectrum of trajectories that can be used to execute manoeuvre T, and so agent 1 must decide on the optimal trajectory based on their predictions for what agent 2 will do. Then, with the knowledge of agent 1's action, agent 2 determines their best response. In this example, agent 2 chooses action W, and then generates a trajectory to execute the manoeuvre.

Stackelberg games are best used to model driving situations where one agent clearly has the ability to move first so that the distinction between leader and follower is unambiguous. This can be the case in merging and overtaking situations, for example, where one vehicle is takes the initiative to move first and other vehicles must react to the manoeuvre. However, the Stackelberg leader-follower assumption is also quite restrictive. There are a broad spectrum of situations that can emerge at intersections, where the distinction between leader and follower is transitory or in other cases entirely ambiguous. Therefore, we use simultaneous move games to model traffic dynamics, thereby avoiding the strong leader-follower assumptions used in Stackelberg games.

In the simultaneous move game agents reason about their optimal action by taking

into account the actions available to all other agents, without having knowledge of what action the other agents have committed to executing. Actions are divided into two levels, high-level manoeuvres and low-level trajectories. Agents first play the manoeuvre-level game, and then once they have selected a manoeuvre they play the trajectory-level game to determine which trajectory should be used to carry out the manoeuvre. Sarkar and Czarnecki [95] employ backward induction to solve the hierarchical simultaneous move game.

Figure 3.1b provides an overview of how backward induction can be used to solve the simultaneous move game between agents 1 and 2. Starting from the bottom of figure 3.1b, both agents first play the trajectory level game, generating a set of trajectories for each manoeuvre and then selecting the optimal trajectory for each manoeuvre based on the other agent's set of trajectories. Sarkar and Czarnecki use a weighted combination of progress and safety functions to measure the utility of each trajectory. The progress utility measures the percentage of the total trip distance the trajectory covers. The safety utility measures the minimum distance between the trajectory being evaluated and another vehicle's trajectory. The goal at this level is to select a trajectory for each manoeuvre that maximizes both the progress and safety utilities. Once a trajectory has been selected for each manoeuvre (meaning that each manoeuvre has an associated utility value), both agents play the manoeuvre level game, which reduces to a standard normal-form game. An example showcasing the entire process of instantiating and solving a simultaneous move game is shown in chapter 4.4.

We use the maxmin *solution concept* for solving trajectory-level games due to its ability to accurately model human driving behaviour [95] (we describe how maxmin is used as a solution concept in the following chapter). The solution concept for the trajectory-level game determines which trajectory becomes the representative trajectory for each manoeuvre. A solution concept must also be chosen for the manoeuvre-level game. Below we outline three different options.

- Nash equilibrium is a standard solution to non-cooperative games [95, 43, 90, 99], where no agent can increase their own utility by unilaterally changing their strategy.
- Stackelberg equilibrium [36] is another solution concept used in leader-follower games, where the leader computes the optimal response by the follower given each of the leader's actions. With this information, the leader can compute its optimal action given the optimal response from the follower [10, 7].
- Finally, Pareto equilibrium [104], is used in cooperative games, as opposed to both Nash and Stackelberg equilibrium, which are used in non-cooperative games—i.e.,

where agents are playing in competition against each other. Pareto equilibrium (also known as *Pareto optimality* and *Pareto efficiency*), is often used in a cooperative setting, and is defined as the state where no reallocation of resources is possible without at least one agent being worse off based on their preferences.

Since the primary contribution of the work is a safety validation methodology, rather than coming up with the most appropriate solution concept for planning, we choose Nash equilibrium as the solution concept for solving manoeuvre-level games mainly due to its ubiquity of use in motion planning literature [90, 41, 78]. Additionally, for cases where there are multiple Nash equilibria, we select the utilitarian social welfare maximizing Nash equilibrium, i.e., a Nash equilibrium that maximizes the sum of the utility of all agents. We calculate Nash equilibria in pure strategies. With that being said, the developed methodology, including occlusion situation generation and hypergame construction, are agnostic to the specific solution concept. Therefore, one can replace the Nash-equilibrium-based planner with a Stackelberg-equilibrium-based planner, for example, and the methods developed in the thesis will work seamlessly.

Nevertheless, in a study comparing how various *bounded-rationality* strategic planners perform, Nash equilibrium with quantal errors was shown to most closely model human driving behaviour [95]. Bounded rationality refers to the notion that humans have a bounded amount of computational resources to figure out what their best action is. Driving is an activity where bounded rationality plays an important role as humans must necessarily choose their best action with a limited computational budget (and thus may not always choose the optimal action). While we do not use Nash equilibrium with quantal errors in this work, we instead use Nash equilibrium as the solution concept which can be interpreted as an optimistic variant of Nash equilibrium with quantal errors. We leave the inclusion of bounded-rationality strategic planners for future work.

A hierarchical game is defined as the tuple, $G = (N, M, \mathcal{T}, U)$, where

- N is the number of agents in the game indexed by $i \in \{1, 2, 3, \ldots, N\}$ (we use *agent* and *vehicle* interchangeably),
- M_i is the set of manoeuvres available to agent i in the manoeuvre-level game (we use *action* and *manoeuvre* interchangeably).
- $\mathcal{T}_{m_i}$ is the trajectory representing manoeuvre, m_i, for agent i,
- U is the utility function $U \rightarrow \mathcal{T} : R^N$, which maps trajectories to the real-valued N-dimensional vector, R^N.

Each game is instantiated as a simultaneous-move game, where agent i's state at time t is defined as the 7-tuple $X_{i,t} = [x, y, v_x, v_y, \dot{v}_x, \dot{v}_y, \theta]$, where (x, y) represents the coordinates of the agent in R^2, (v_x, v_y) are the longitudinal and lateral velocities, $(\dot{v}_x, \dot{v}_y)$ are the longitudinal and lateral accelerations, and θ is the agent's yaw. The manoeuvres in the game are cubic spline trajectories [56, 3], and provide complete state information for each agent over a 6 s planning horizon. We denote a trajectory for agent i starting at time, t, as $\mathcal{T}_{m_i,t} = \{X_{i,t}, X_{i,t+0.1}, \ldots, X_{i,t+6.0}\}$. We generate trajectories with a time horizon of 6 s, where the timesteps are in increments of 0.1 s.

The utility of each trajectory is measured using a combination of progress and safety functions. For vehicle i with trajectory $\mathcal{T}_{m_i}$, the progress function maps the distance that $\mathcal{T}_{m_i}$ covers to the interval $[0, 1]$ (higher values indicating higher progress levels), while the safety function compares how close i gets to another vehicle j by computing the minimum distance between the trajectories $\mathcal{T}_{m_i}$ and $\mathcal{T}_{m_j}$, and mapping the minimum distance to a value in the range $[-1, 1]$ using a sigmoid function (where a higher value indicates a higher safety level). The total utility, U_i, is a function of both the progress and safety utilities, combined based on lexicographic thresholding [69],

$$U(\mathcal{T}_{m_i}) = \begin{cases} U_s(\mathcal{T}_{m_i}, \mathcal{T}_{m_j}) & \text{if } U_s(\mathcal{T}_{m_i}, \mathcal{T}_{m_j}) < \gamma, \\ U_p(\mathcal{T}_{m_i}) & \text{if } U_s(\mathcal{T}_{m_i}, \mathcal{T}_{m_j}) \geq \gamma. \end{cases} \tag{3.1}$$

Where U_p is the progress function, U_s is the safety utility, and γ is a threshold constant that determines if the safety or progress utility should be used to represent the total utility value of $\mathcal{T}_{m_i}$.

There may be some who are at this point confused about how we can use backward induction given that the simultaneous-move game does not seem to have any subgames. The key idea is that we can break the information sets in the trajectory game, given the assumption that a decelerate-to-stop (D) manoeuvre and a track-speed (T) trajectory combination, for example, cannot happen. The backward induction step is shown in figure 3.2, where the hierarchical game is divided into the trajectory-level games, shown as the four separate coloured boxes. We use the non-strategic maxmin solution concept for the trajectory-level games. Then, the solutions to the trajectory-level games become the utilities in the manoeuvre-level game, represented in normal form. The process shown in figure 3.2 is analogous to performing backward induction on the extensive form of the hierarchical game shown in figure 3.1b.

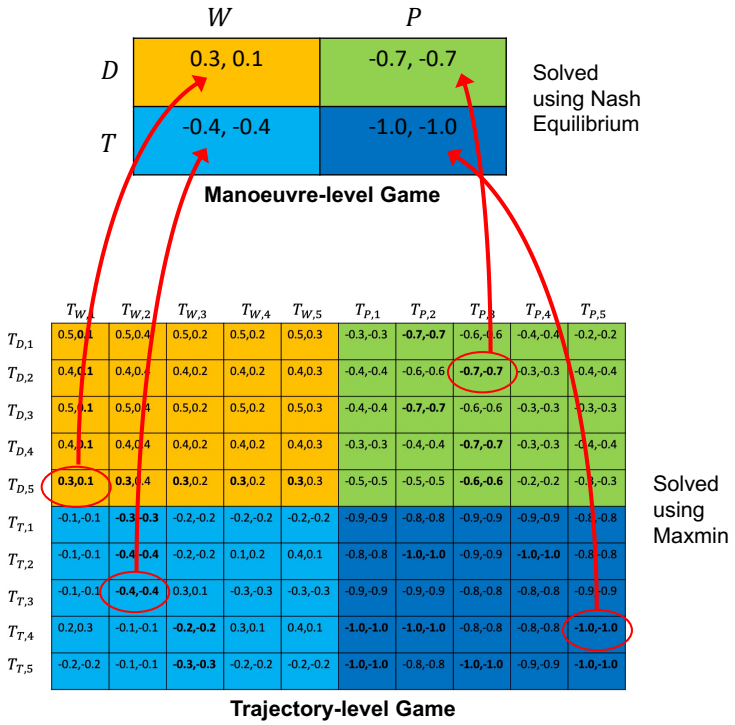


Figure 3.2: An example of how backward induction is performed on a hierarchical traffic game.

3.2 Hypergames

Standard game theory cannot model occlusion since games are instantiated with the implicit assumption that each agent is aware of every other agent participating in the game, which is not the case in occlusion situations where vehicles are hidden from sight. In order to incorporate occlusion in our work we propose the novel application of *hypergames* [11, 14, 63], to incorporate *misperception* into game-theoretic traffic modelling. The key idea being that agents do not always have a mutually-consistent understanding of the game they are playing. For example, consider two countries, A and B, who are preparing for war against each other. However, one of the informants in A mistakenly misinforms country A that country B possesses an army of 3000 soldiers when in reality they possess 5000 soldiers. A historian with complete knowledge of the army sizes of both A and B might be baffled by the decisions made by A if the historian uses standard game-theoretic models. This is because they are assuming both countries knew the size of the other's army. Using

a hypergame instead to model this conflict would allow the historian to reason over each country's individual views of the game. In the hypergame, two games are instantiated, one game from A's perspective (where they believe B has an army of 3000), and another game from country B's perspective. Reasoning over the hypergame would allow the historian to accurately model the individual beliefs that each agent possesses.

Hypergames have a rich history of being used to model situations spanning military conflicts [15, 13, 101, 112, 93, 109, 38], sports [16], resource allocation [84, 48], business [17, 44, 39, 12, 45], and cybersecurity [62, 57, 49]. In general, hypergames are typically used to model adversarial situations where agents do not have perfect information of each other, and as a result, misperceptions are a frequent occurrence. Specifically, a misperception can be [63, 39]:

(I) having an incorrect understanding of the preferences of other agents,

(II) having an incorrect understanding of the actions available to other agents,

(III) not having awareness of all other agents playing the game,

(IV) any combination of the above.

In the context of our work, we focus on (III) since this kind of misperception can be caused by occlusion.

Additionally, we use the extension to hypergame theory proposed by Wang et al. [110], which splits hypergames into a hierarchy of levels based on the perceptions of the agents playing the game. In a *level-0 hypergame* the hypergame reduces to a standard game, $H^0 = G$, because all agents have perfect information about all other players and therefore no misperceptions exist. In a *level-1 hypergame*, $H^1 = \{G_1, G_2, \ldots, G_N\}$, agents have their own view of the game and are not aware that other agents may have differing views of the game. Here, $G_i = \{N_i, K_i, A_i, U_i\}$ denotes agent i's view of the game. A *level-2 hypergame*, $H^2 = \{H_1^1, H_2^1, \ldots, H_N^1\}$ occurs when at least one agent is aware that a hypergame is being played. Here, $H_i^1 = \{G_{i,1}, G_{i,2}, \ldots, G_{i,N}\}$ represents the hypergame as perceived by agent i and $G_{i,j}$ is the jth agent's game as perceived by agent i. While hypergame levels can in theory be extended up to a finite level L, we only use level-2 and lower hypergames in order to model dynamic occlusion.

1. Level-0 (Occlusion-resolved perspective): In the level-0 hypergame, all agents possess an omnispective view of the game and so each agent is aware of all other agents

playing the game. We use the occlusion-resolved perspective to verify that collisions occurring in the level-1 (occlusion-naive) hypergame are a result of occlusion. If a collision occurs in the level-0 hypergame in addition to the level-1 hypergame then occlusion was not the cause of the accident since we eliminate occlusion as a variable in the level-0 hypergame.

2. Level-1 (Occlusion-naive perspective): Dynamic occlusion is incorporated at this level. This means each vehicle constructs a game from their own perspective, and each game only includes vehicles that are not occluded from that vehicle's perspective.

3. Level-2 (Occlusion-aware perspective): In a level-2 hypergame the vehicles know they are playing a hypergame and therefore understand that dynamic occlusion is present in the situation. We detail how a vehicle using their occlusion-aware perspective, can generate a *dynamic occlusion risk* (DOR) measure for the situation in section 3.3.

In order to better illustrate how we use hypergames to model dynamic occlusion, we provide the following example. Figure 3.3 shows an occlusion situation generated using our method. The vehicles circled in red are taken from naturalistic data, while the *synthetic occluding vehicle* (SOV) circled in blue is generated using our method. Both the SOV and vehicle 1 are in the left-turn lane. Vehicle 2 is driving straight through the intersection from the other direction and following behind vehicle 3. This is an *occlusion situation* because neither vehicle 1 nor 2 can see each other due to the SOV.

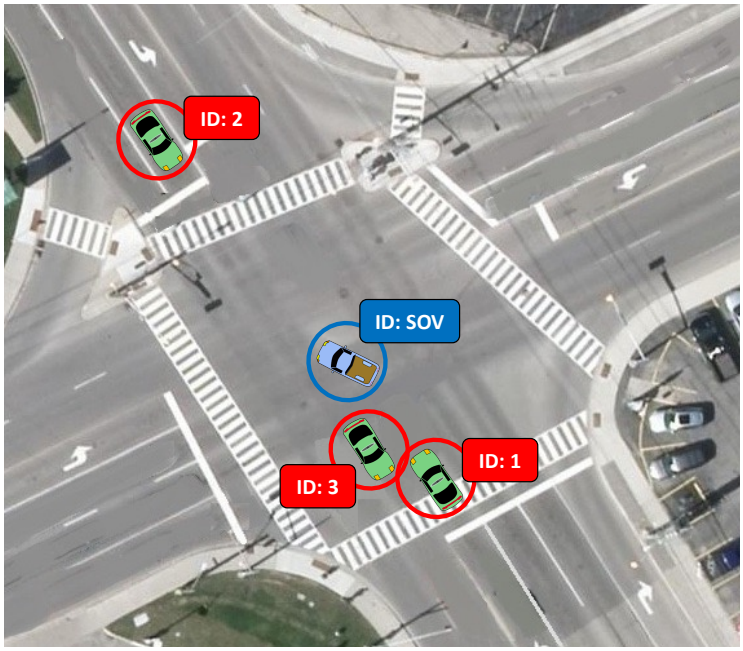


Figure 3.3: An example of an occlusion situation generated using our method. We use this occlusion situation as a running example throughout this thesis.

We model this situation as the level-1 hypergame shown in figure 3.4, in which we show the perspectives of the game from vehicle 1's perspective 3.4a, and the game from the SOV and vehicle 3's perspectives 3.4b. Vehicle 2 is not included in vehicle 1's game because vehicle 2 is occluded from vehicle 1. In contrast, all vehicles are included in figure 3.4b because from the SOV and vehicle 3's view no vehicle is occluded. We did not include the game tree for vehicle 2 but it would look similar to figure 3.4a except vehicle 2 would be included and vehicle 1 would not.

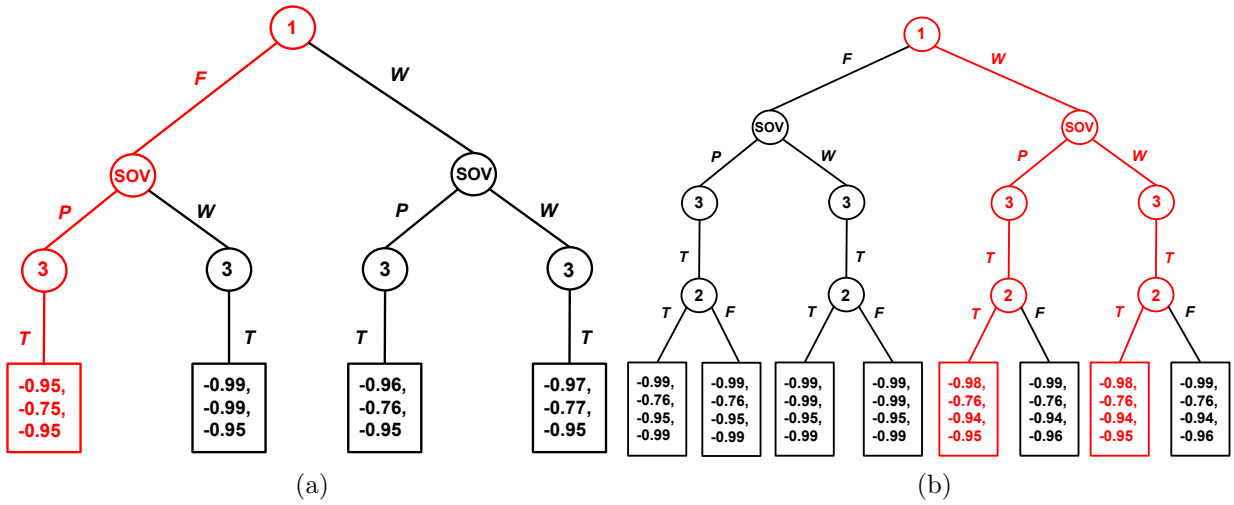


Figure 3.4: (a) Vehicle 1's view of the traffic game. (b) Both the SOV and vehicle 3's view of the traffic game. The red paths and utilities represent the Nash equilibria actions and utility values.

In figure 3.4b vehicle 1 has two available manoeuvres, following the lead vehicle into the intersection (F) or waiting for vehicle 2 to pass (W) (a full list of the available manoeuvres and their descriptions we use in this work are provided in table A.1). The SOV can either execute its left turn (P) or wait for vehicle 2 to pass (W). Vehicle 2 can either execute a follow manoeuvre whereby it matches its speed to vehicle 3 as it drives through the intersection (F) or it can instead drive at the speed limit through the intersection (T).

The paths highlighted in red indicate a Nash equilibrium. For example, in figure 3.4b (W, P, T, T) is a Nash equilibrium, which represents the maneuvers for vehicle 1, the SOV, and vehicles 2 and 3 respectively. From vehicle 1's perspective (in figure 3.4a), the only Nash equilibrium is (F, P, T), which has vehicle 1 following the SOV into the intersection. Thus, vehicle 1's misperception of the full traffic game being played can lead to the dangerous situation where it executes a left turn while vehicle 2, unaware of vehicle 1, drives straight through the intersection. Indeed, simulating this situation we find that vehicles 1 and 2 end up colliding due to the occlusion caused by the SOV.

At this point, a question that may arise to some readers is why we opt to use hypergames as a framework to model occlusion as opposed to Bayesian games, which allow for the possibility that players in the game may have incomplete information about other players. For example, in a Bayesian game a player may not know the exact payoff function that other players use and so instead model a probability distribution representing their beliefs over

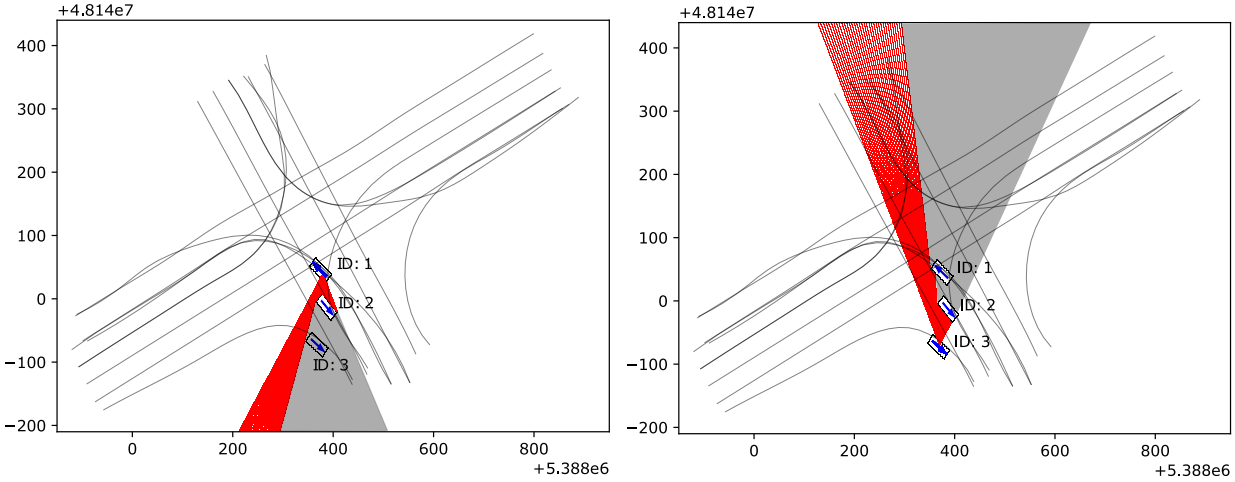
the possible payoff functions that other players may be using. The first reason why we opt to use hypergames over Bayesian games is because of the simplicity of hypergames. Bayesian games, while a powerful modelling tool, require that an accurate model of the probability distribution over each player's beliefs can be computed. Computing these distributions is a non-trivial task. In contrast, hypergame theory provides a straightforward way to incorporate occlusion into traffic games by creating individual games from each player's perspective, which is a significantly simpler task than instantiating and solving Bayesian games. For our work then, hypergames suffice. Secondly, while Bayesian games allow for the possibility that players may have incomplete knowledge of other players, it is not clear how Bayesian games handle the case where players are not even aware that other players (i.e., occluded vehicles) are involved in the game. Hypergames on the other hand provide a simple and effective framework that incorporates this case with little computational overhead, making it again the clear choice for this work.

3.3 Dynamic Occlusion and DOR

Dynamic occlusion occurs from specific spatial configurations involving three or more vehicles. We use the occlusion-indicator function $O(i, j, k) \in \{0, 1\}$, to check for occlusions, where $O(i, j, k) = 1$ indicates that vehicle k is occluded from vehicle i by vehicle j, and $O(i, j, k) = 0$ indicates that vehicle k is visible to vehicle i. Note that occlusion is *not* necessarily symmetric, meaning that $O(i, j, k) = 1$ does not imply that $O(k, j, i) = 1$. We illustrate this with the examples shown in figures 3.5a and 3.5b.

In figure 3.5a frames the situation from vehicle 1's perspective. In this case, $O(1, 2, 3) = 1$ because vehicle 2 is occluding vehicle 3 from vehicle 1. However, in figure 3.5b from vehicle 3's perspective, $O(3, 2, 1) = 0$, because while vehicle 2 is partially occluding vehicle 1, vehicle 3 still has partial vision—5 rays still manage to collide with vehicle 1—and therefore still has vision of vehicle 1. In general, the positioning and orientation of the human driver—or the suite of sensors in the case of an AV—determine if occlusion is symmetric or not.

If for all vehicles in the traffic situation $O(i, j, k) = 0$ then the highest level hypergame that can be played is 0 since each vehicle is aware of all other vehicles in the situation and they all play the common game $H^0 = G$. However, if there is at least one case where $O(i, j, k) = 1$ for any combination of vehicles i, j, and k then the level of the hypergame must be at least 1 since there will be at least one vehicle that will have a different view of the traffic game due to occlusion.



(a) (b)

Figure 3.5: An example of asymmetric occlusion between two vehicles. (a) Vehicle 3 is occluded from vehicle 1, since no rays from vehicle 1 collide with vehicle 3. (b) Vehicle 1 is still visible to vehicle 3 since 5 raycasts collide with vehicle 1.

One goal of this work is to be able to evaluate how much dynamic occlusion contributes to driving risk. In order to do this, we define *dynamic occlusion risk* (DOR). However, since we assume all the vehicles in the traffic game are occlusion-naive it follows that none of the vehicles can compute the DOR measure (which requires an occlusion-aware perspective). To get around this we introduce the notion of a third party observer, such as an AV, that is able to observe all vehicles in the traffic situation (e.g., an AV could use V2X communication infrastructure to gain information on the positions of all vehicles in the situation). This third party would then compute the solutions to H_0 and H_1, which would result in two sets of trajectories, $\mathcal{T}_{H_0}$, the set of trajectories used by vehicles in the occlusion-resolved view (H_0) and, $\mathcal{T}_{H_1}$, the set of trajectories used in the occlusion-naive view (H_1). Using a surrogate safety metric, $\mathcal{S}$ (we use minimum distance gap), we now define DOR as follows:

$$DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) = \mathcal{S}(\mathcal{T}_{H_0}) - \mathcal{S}(\mathcal{T}_{H_1}). \tag{3.2}$$

$\mathcal{S}(\mathcal{T}_{H_0})$ is the minimum distance gap between all trajectories executed in H_0 and $\mathcal{S}(\mathcal{T}_{H_1})$ is the minimum distance gap between all trajectories executed in H_1. DOR measures the difference in outcomes when all vehicles have an omnispective view of each other

(H_0) and when dynamic occlusion obstructs vehicles from seeing each other (H_1), thus $DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) > 0$ implies that dynamic occlusion contributes to whatever driving risk is present in the traffic situation. We define an *occlusion-caused collision* (OCC) when $DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) > 0$ and $\mathcal{S}(\mathcal{T}_{H_1}) = 0$ (since this implies there was a collision in H_1 but not in H_0). More succinctly, OCC $:= DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) = \mathcal{S}(\mathcal{T}_{H_0}) > 0$.

Chapter 4

Method

In this chapter we describe how we generate occlusion situations, as well as how we evaluate strategic planners on those occlusion situations. Figure 4.1 shows an overview of our workflow. Since our occlusion situation generation method requires augmenting traffic data, the first step is to select a naturalistic traffic dataset. Once selected, step 2 decomposes full traffic scenes from the dataset into *partial scenes*, which represent subsets of vehicles grouped by how relevant they are to each other's behaviour (e.g., a lead vehicle would be grouped with the follower vehicle since the lead vehicle can affect the follower's behaviour). In step 3, we augment each partial scene by spawning vehicles in a variety of positions in order to cause occlusion. With the occlusion situations generated in step 3, we play the level-0 and level-1 hypergames in steps 4 and 5 respectively. With the results from steps 4 and 5, we compute the DOR measure for each occlusion situation in step 6, and then identify and record OCCs in step 7. We classify our proposed method as a situation-based accelerated evaluation method for AVs [115, 92], which includes

1. extracting situations from naturalistic driving data (steps 1 and 2),
2. guided generation of novel test situations based on a target condition (step 3),
3. identification of risk based on a quantifiable measure and assessment of the system under test using the selected risk measure (steps 4-7).

We cover each of these steps in greater detail in the following sections.

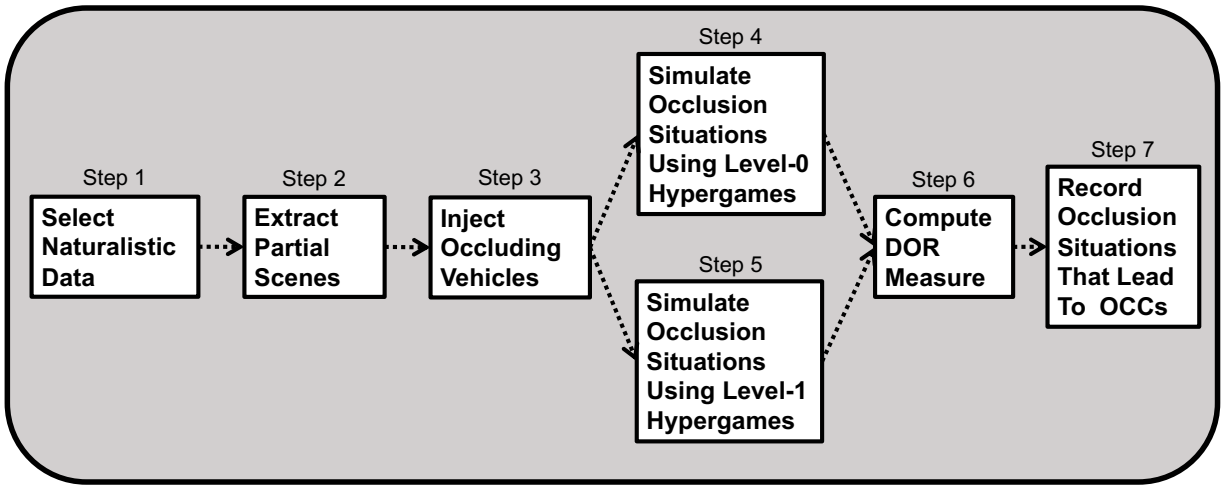


Figure 4.1: An overview of our situation-based accelerated evaluation method.

4.1 Step 1: Selecting a Naturalistic Traffic Dataset

Our method augments naturalistic driving scenes with SOVs to create high-risk occlusion situations. In order to realistically simulate the traffic situations we generate, we require the ground-truth positions, orientations, and velocities of the naturalistic vehicles. We use the Intersection dataset from the Waterloo Multi-agent database[1] which contains recorded traffic data from the University-Weber intersection in Waterloo, Canada. The traffic data was first recorded by drone and the subsequent video recordings were sent to a third party for data processing. The dataset contains over 3.5k unique vehicles, and over 1.9 million data points representing each vehicle's ground-truth trajectory information (i.e., position, orientation, velocity, acceleration, etc.) as they cross the intersection.

4.2 Step 2: Partial Scene Extraction

Full traffic scenes taken directly from the intersection dataset can involve a large number of vehicles, not all of whom are relevant to each other. It can be computationally expensive and inefficient to solve the hierarchical game for all vehicles present in a naturalistic driving scene, which can include twenty or more vehicles. We want each occlusion situation we generate to only include, in addition to the synthetic vehicle we inject, vehicles that are

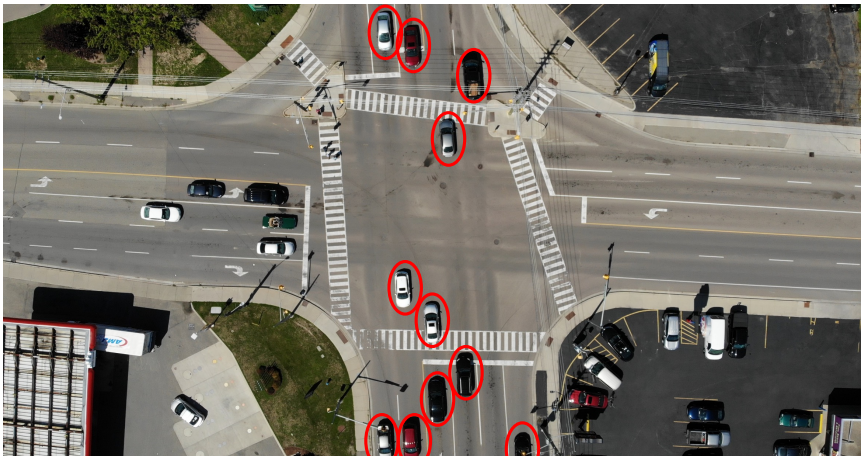
[1] http://wiselab.uwaterloo.ca/waterloo-multi-agent-traffic-dataset/

relevant to the situation. Figure 4.2 shows the process we take to decompose a full traffic scene from the dataset into smaller subsets of partial scenes, and then finally into occlusion situations. In order to decompose the full scene into partial scenes, we use a subject-vehicle/relevant-vehicle relationship to discern which vehicles should be grouped together. The idea being that each vehicle in the full scene will have their own partial scene where they are the subject vehicle. We then determine the set of relevant vehicles for that subject vehicle. Our goal, restated in the subject/relevant vehicle framework, is then to inject SOVs into the partial scenes we extract in order to cause high-risk occlusion between the subject vehicle and *one or more of the relevant vehicles*. By designating each vehicle in the full scene as the subject vehicle in their own partial scene we make sure to cover high-risk occlusion situations arising from every vehicle in the full scene.

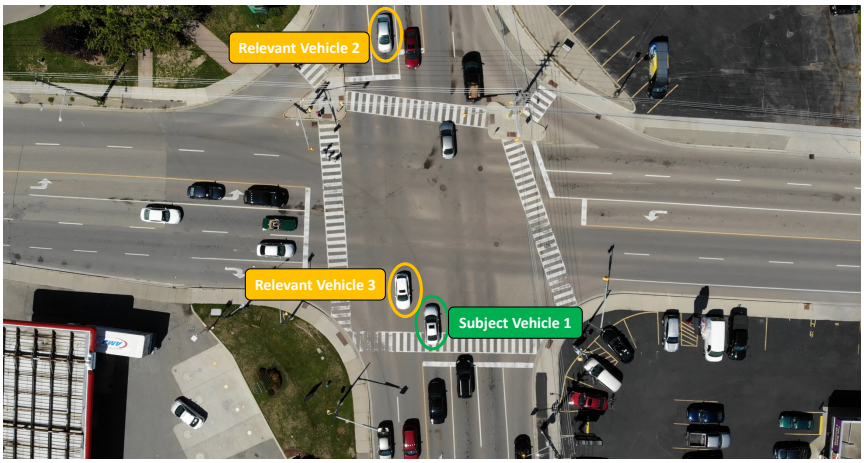
However, since our goal is to generate high-risk occlusion situations involving the subject vehicle, we want to ensure subject vehicles are allowed to execute manoeuvres in the intersection. This is not possible if they are stopped at a red light. Therefore, subject vehicles must be either (a) straight-through vehicles with a green or yellow light, (b) left-turn vehicles with a green or yellow light, or (c) right-turn vehicles with a green, yellow, or red light. The vehicles that are eligible to be subject vehicles in figure 4.2a are circled in red. The vehicles not circled in red are not eligible to be subject vehicles. We ignore them in our analysis because even if they were to be occluded from other vehicles, the occlusion would not result in a high-risk of a collision, since these vehicle are not allowed to proceed through the intersection.

Next, we compute the set of relevant vehicles for each subject vehicle. A relevant vehicle is either

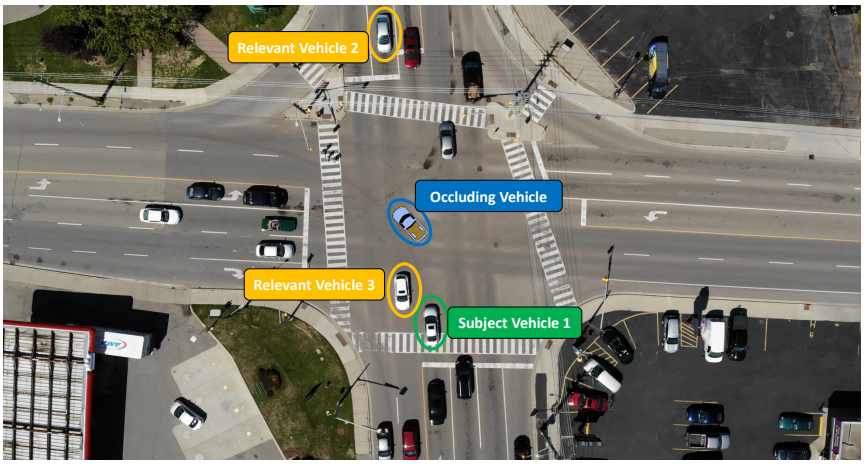
1. a leader to the subject vehicle,
2. in a conflicting lane to the subject vehicle,
3. a leader to a vehicle in a conflicting lane to the subject vehicle.



(a)



(b)



(c)

Figure 4.2: (a) The full traffic scene. Circled vehicles have a green light. (b) An example of a partial scene. (c) The same vehicles from the partial scene in (b) but an SOV has been injected, causing occlusion between vehicles 1 and 2.

In figure 4.2b we show an example of one partial scene extracted from the full scene to illustrate how the process is performed. The subject vehicle is in the left-turn lane and has moved into the intersection to make its turn. Relevant vehicle 2, at the top of figure 4.2b, is driving straight-through the intersection and is therefore in a lane that conflicts with the subject vehicle's path. Relevant vehicle 3 at the bottom of the figure is leading relevant vehicle 2.. The reason we include leading vehicles is because they can change the behaviour of follower vehicles, since following vehicles must modulate their speed to match the leader's as well as maintain a distance gap.

In order to efficiently use the dataset, we extract full scenes from the dataset—which are then decomposed into partial scenes—1.0 s apart to generate occlusion scenarios. The WMA intersection dataset is roughly 3300 s long, meaning we end up extracting roughly 3300 full scenes. Figure 4.3 shows our workflow for generating occlusion situations from a full scene taken at time t.

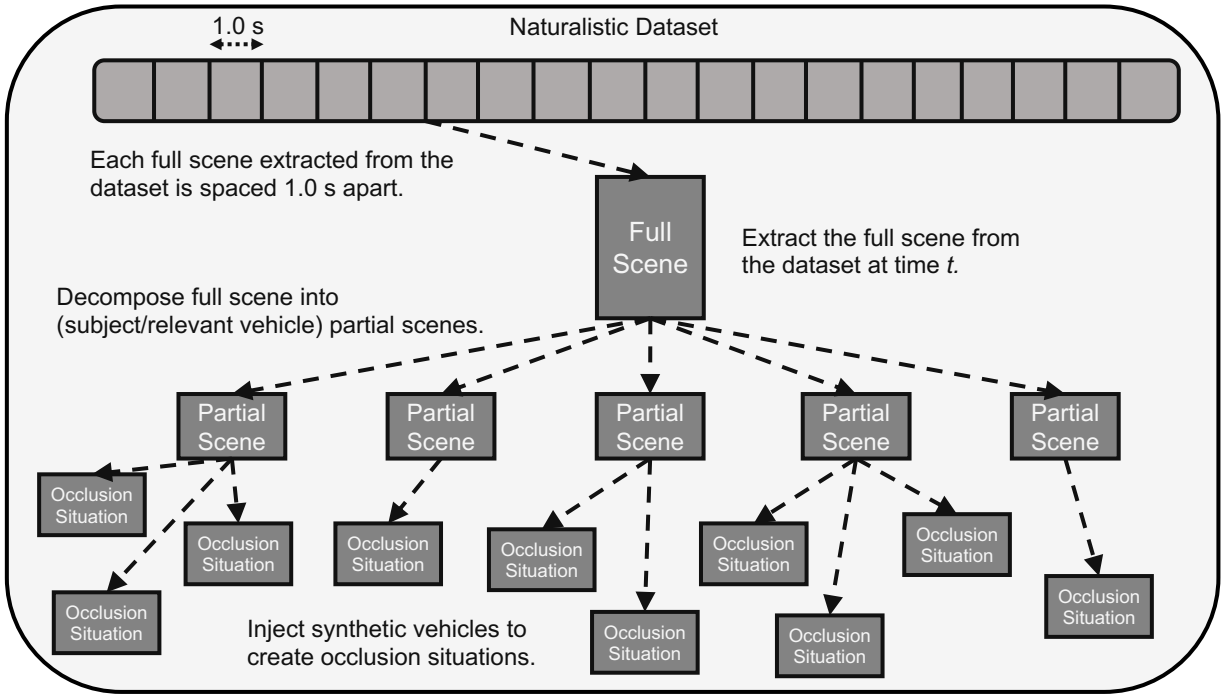


Figure 4.3: Our workflow for generating occlusion situations from a full scene taken at time t from the dataset.

First, all eligible subject vehicles are selected, and then partial scenes are created for each subject vehicle. A range of occlusion situations are then created from each partial

scene by injecting SOVs (each occlusion situation only containing one SOV). Note that some partial scenes result in more occlusion situations while others result in less. This is due to how the vehicles are positioned in the partial scene, thus some partial scenes allow for many occlusion situations while others do not. The process of creating occlusion situations from partial scenes is explained next.

4.3 Step 3: Injecting SOVs

There are many approaches one could take to injecting SOVs into partial scenes. One approach is to randomly sample positions from the intersection map, spawning SOVs at those sampled locations, and checking to see if they cause occlusion. However, this approach is not ideal as it can result in unrealistic SOV positions, such as spawning SOVs between lanes or on the sidewalk. Instead, we use domain knowledge to construct valid spawn locations. The set of all SOV positions we sample from is shown in figure 4.4, where the lines represent the centrelines for each lane in the intersection and the red dots represent the positions of each SOV spawn location.

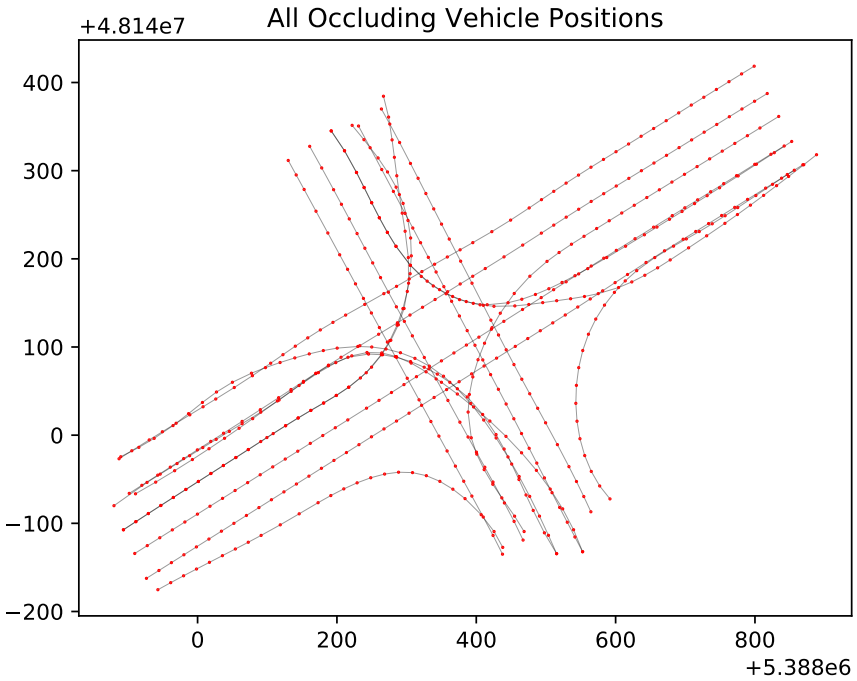
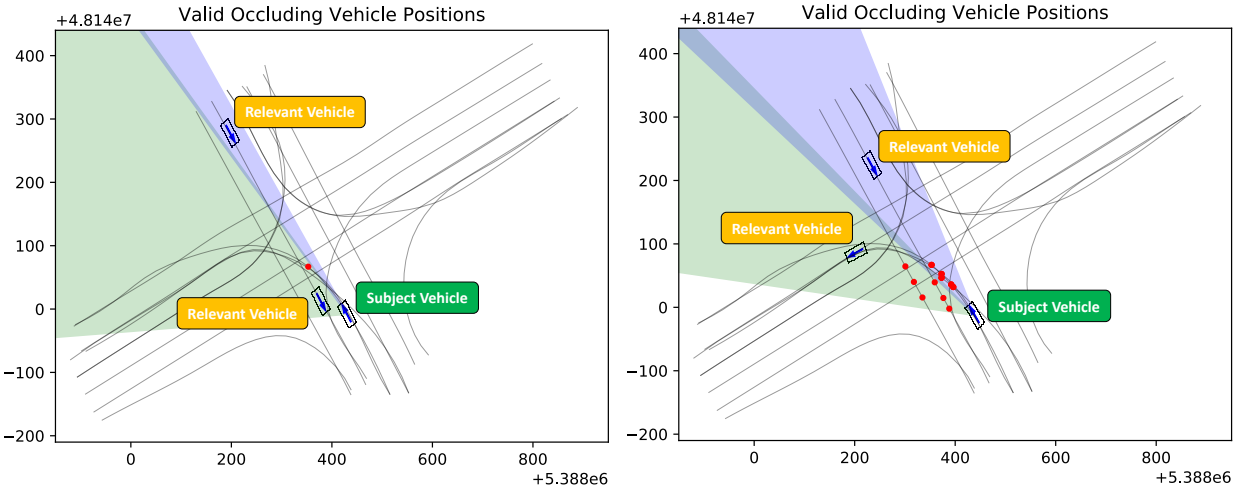


Figure 4.4: All positions in the intersection map that can be used to spawn SOVs.

Spawning SOVs on the centrelines helps to ensure that the resulting occlusion situations are realistic since vehicles typically drive on or near the centreline. Each spawn location is approximately 1 m apart which provides a good balance between covering the entire intersection map without having a large computational overhead of having to handle too many spawn locations.

Given a partial scene, our task is to select the subset of spawn locations that are valid for the subject vehicle in the partial scene. Figure 4.5 shows two partial scenes (a) and (b) where the red dots represent the valid SOV spawn locations for each scene. A valid SOV spawn location must (i) be within the FOV region of the subject vehicle, (ii) be a minimum of 1 m away from the subject and relevant vehicles, and (iii) must be in a lane with a green or yellow light. This last point is to prevent SOVs from being spawned in the middle of the intersection when they should not be there, however, it does also mean that we do not generate right-turning SOVs that have a red light. Figure 4.5a shows that there is only one valid SOV spawn location in this partial scene. In contrast, in figure 4.5b there are multiple valid SOV spawn locations.



(a) (b)

Figure 4.5: Determining valid SOV spawn locations across two different partial scenes. Note that (a) is the same partial scene as the one shown in figure 4.2b.

In both examples, the subject vehicle's FOV is divided into two regions, the green region directed at one relevant vehicle, and the blue region directed at the other relevant vehicle. The size of these regions is determined by how much *attention* the subject vehicle

pays to each vehicle. Attention is calculated based on how close the vehicle is to the subject vehicle. The motivation behind this is that vehicles closer to the subject vehicle are more relevant to the subject vehicle's behaviour and should therefore receive more attention. For example, in figure 4.5a the closer relevant vehicle receives the majority of the subject vehicle's attention, due to its proximity, and therefore the FOV region is much larger than the other vehicle's.

We calculate how much attention, A_i, the subject vehicle pays to vehicle i using

$$A_i = \frac{\sum_j^{\mathcal{N}} d_j - d_i}{\sum_j^{\mathcal{N}} d_j}. \tag{4.1}$$

Where d_i is the distance between the subject vehicle and vehicle i, and $\mathcal{N}$ is the number of vehicles visible to the subject vehicle. Once attention scores are computed for each vehicle, we normalize all the attention scores such that the sum equals 1.0. Therefore, $A_i = 0.4$ is analogous to saying the subject vehicle pays 40% attention to vehicle i. Of course, the attention scores are just half the story. In order for the attention scores to be meaningful they must be converted into FOV ranges. The subject vehicle has a total FOV range of η degrees. The attention scores determine how much of the total FOV range is allocated to each vehicle. Thus, if $A_i = 0.4$ then 40% of the total FOV budget goes to vehicle i. The FOV region for each vehicle is always centered on the vehicle so in this case the FOV region would extend $0.2 * \eta$ degrees to the right and $0.2 * \eta$ degrees to the left of vehicle i. This means there can be overlapping FOV regions for different vehicles, which can be seen in figures 4.5a and 4.5b.

Once the set of valid SOV spawn locations has been selected, the next step is to determine how much occlusion each SOV causes. We use a grid-based raycasting approach based on the work by Amanitides and Woo [6], as a way to simulate each vehicle's perception system. Each vehicle's bounding box is approximated as a set of squares on the grid, where each grid square is 10 cm $\times$ 10 cm. In order to determine which vehicles are visible to the subject vehicle, the subject vehicle performs a raycast check for each vehicle. We use the same attention-based FOV regions from the previous step to determine the direction and range of the rays. If a ray hits another vehicle's bounding box, it will end there and not continue. If ϵ or fewer rays from the subject vehicle hit vehicle i, then vehicle i is considered occluded from the subject vehicle. Figure 4.6a shows the same partial scene as figure 4.5a but with the visible regions to the subject vehicle. Figure 4.6b shows the occlusion situation—which is the same partial scene with the addition of the SOV. In the occlusion situation, relevant vehicle 2 is occluded by the SOV.

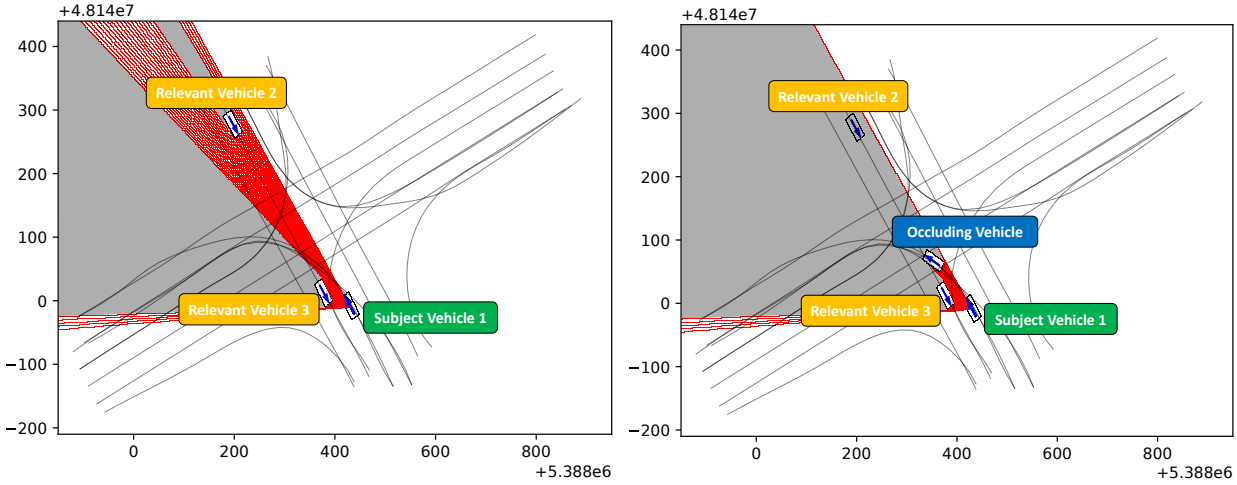


Figure 4.6: Example of a traffic situation from naturalistic data with (b) and without (a) a synthetic occluding vehicle. The grey region indicates that anything in that area is occluded from the subject vehicle (a) The subject vehicle can see both relevant vehicles. (b) The subject vehicle can see the closer relevant vehicle but not the other relevant vehicle due to the occluding vehicle.

We summarize the process of generating occlusion situations from naturalistic data in algorithm 1. The output of algorithm 1 is the set of occlusion situations, $D^{\mathcal{O}}$, created from augmenting partial scenes from the dataset. D is the set of all partial scenes from the dataset. The function `occluding_vehicle_sampling(S)`, takes as input a partial scene, S, and outputs V, the set of state vectors, X_o, for SOVs that have a valid spawn location for S. The state vector, along with position information, also includes orientation and initial speed information. Likewise, each vehicle, v, in the partial scene, S, is a state vector containing position, orientation, and speed information.

For each occlusion situation we simulate the level-0 and level-1 hypergames and compute the DOR measure. However, in order to simulate the occluding vehicle's behaviour we need to initialize the SOV with realistic values. The SOV's orientation is set to the centreline's tangent vector at the SOV's spawn location. However, the SOV's initial speed is more of a challenge. In order to set a realistic initial speed for each SOV we first analyze the speed distributions in each of the intersection's *lanelets*. A lanelet is a component of the lane, such that a set of connected lanelets forms a lane. For example, the straight-through lane covering the path from South to North includes the entrance lanelet, `ln_s_2`, the lanelet covering

Algorithm 1 Occlusion Situation Generation Algorithm

$D^{\mathcal{O}} \leftarrow \{\emptyset\}$ ▷ Initialize set of occlusion situations
for $S \in D$ **do**: ▷ Iterate through partial scenes
  $V \leftarrow$ occluding_vehicle_sampling(S)
  for $v_o \in V$ **do**:
    for $v \in S$ **do**
      if $\exists x \in S \setminus v; O(v, v_o, x) = 1$ **then** ▷ Check if v_o causes occlusion
        $S \leftarrow \{v_o\}$ ▷ Add SOV to partial scene
        $D^{\mathcal{O}} \leftarrow D^{\mathcal{O}} + \{S\}$ ▷ Add occlusion situation to $D^{\mathcal{O}}$
      end if
    end for
  end for
end for
return $D^{\mathcal{O}}$

the middle of the intersection, `l_s_n_l`, and the exit lanelet, `ln_n_-1`. The South-to-West left-turn lane includes the entrance lanelet, `ln_s_1`, the preparatory left-turn lanelet, `prep-turn_s`, the lanelet in which the vehicle executes their left-turn, `exec-turn_s`, and the exit lanelet, `ln_w_-1`. The intersection map with the lanelets overlaid can be found at the url, https://git.uwaterloo.ca/a9sarkar/traffic_behavior_modeling.

We create the speed distributions for each lanelet by extracting vehicle speeds from the dataset. However, simply extracting the speeds in this way ignores that vehicle speed is conditioned on factors such as traffic light state. Therefore, we split the speed distribution for each lanelet into conditional speed distributions based on

- the traffic light state,
- if the light just turned green,
- if the light is a dedicated green,
- if there is a vehicle in a conflicting lane.

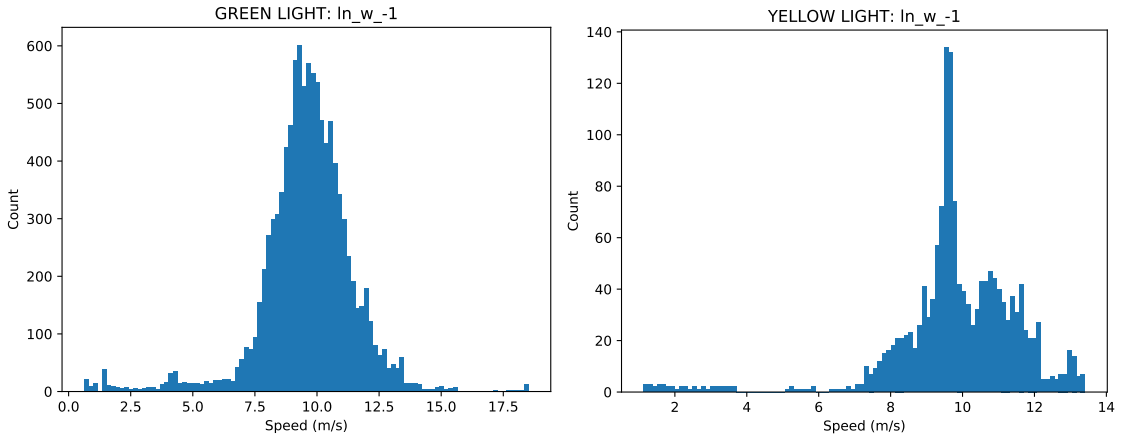
The traffic light state can affect the urgency with which vehicles travel and therefore can impact the vehicle's speed. Additionally, we found that recording vehicle speeds immediately after the traffic light changed to green resulted in different speeds as opposed to waiting for 5 s for vehicles to begin moving. Therefore, we divide some speed distributions based on if the speed was recorded within 5 s of the light changing to green (which we

denote as *without delay*), or recorded after 5 s of the light changing to green (which we denote as *with delay*). A dedicated green light can also significantly affect left-turn speeds since turning vehicles do not have to worry about conflicting vehicles, so we include this condition in our analysis. Finally, we divide left- and right-turn lanelet speed distributions based on if there is a conflicting vehicle present, which can affect the speed of turning vehicles.

We observed that the speed distributions broadly fell into either normal distributions or exponential distributions. Lanelets where there is never an opportunity for vehicles to stop (such as exit lanelets) typically follow a normal distribution, whereas lanelets where it is common for vehicles to yield to oncoming traffic typically follow an exponential distribution, with a peak at 0 m/s and a sharp slope downwards for higher speeds. For example, figures 4.7a and 4.7b show the green and yellow light speed distributions for the lanelet, `ln_w_-1`, which vehicles use to exit the intersection. The two distributions are roughly normal distributions. Figures 4.7c and 4.7d show the speed distributions for the left-turn lanelet `prep-turn_s` on a green light with (4.7c) and without (4.7d) a conflicting vehicle present. These two distributions are roughly exponential distributions. Unsurprising, there are, on average, lower recorded speeds when there is a conflicting vehicle present since the left turning vehicle is more likely to drive slowly in order to wait for the conflicting vehicle to pass. The highest speed recorded with a conflict measures just roughly 4 m/s while the highest speed with no conflict measures above 10 m/s.

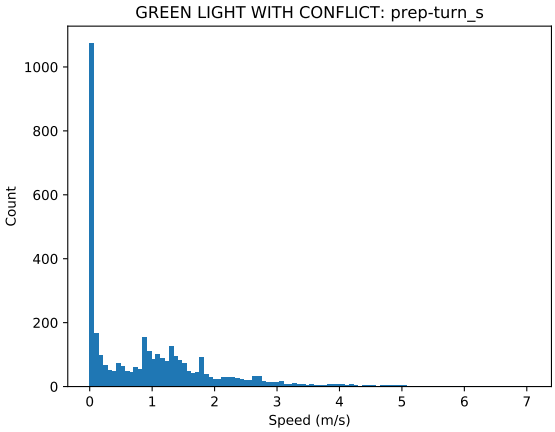
We observed that the speed distribution for lanelets that are the entrances to the straight-through lanes roughly follow an exponential distribution for the first few seconds after the light turns green and eventually become more normally distributed as time passes. The reason for this is that vehicles are initially stopped and therefore it takes time for them to begin moving. Furthermore, vehicles in these lanelets must wait for their leading vehicle to begin moving before they can move. Therefore, it takes around 5 s (which we empirically observed from the dataset) from the time the light turns green for most vehicles in the entrance lanelets to begin moving. Figures 4.7e and 4.7f are an example of this phenomenon. We see that the speeds for the first 5 s (i.e., *without delay*) in the entrance lanelet, `ln_s_2` roughly follow an exponential distribution with a sharp peak at 0 m/s and a dramatic tapering off for higher speeds. However, after a delay of 5 s, the second figure, 4.7e, shows that while there are still vehicles that are not moving, most are now moving and the speeds roughly follow a normal distribution.

One challenge we encountered while extracting these distributions was that there was not always enough data to extrapolate a speed distribution. This was the case with most of the yellow light speed distributions. In order to ensure reasonable initial speeds were chosen for the SOVs we opted to use default speeds when this was the case.

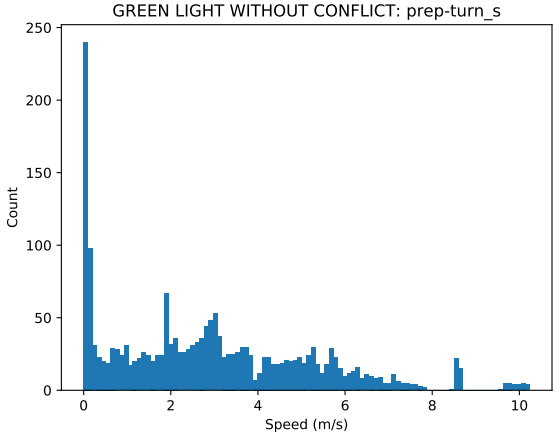


(a)

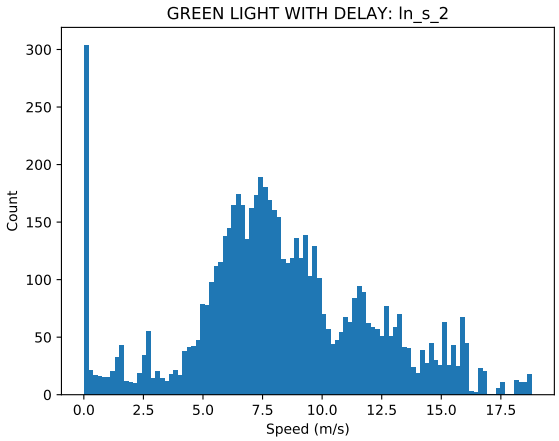
(b)



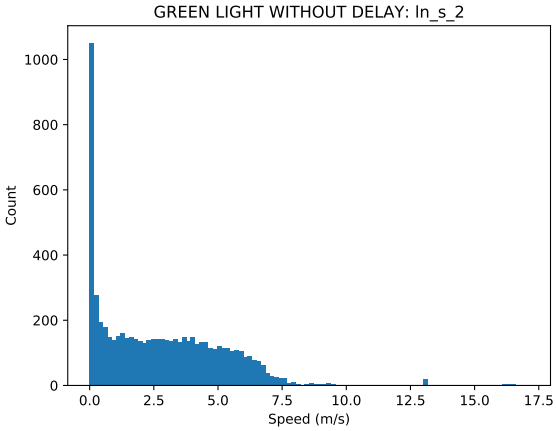
(c)



(d)



(e)



(f)

Figure 4.7: Examples of conditional speed distributions extracted from lanelets in the intersection.

There are primarily two types of manoeuvres that vehicles can perform in the traffic game, these are "go" manoeuvres and "stop" manoeuvres. Go manoeuvres include *track-speed*, *proceed-turn*, *follow-lead*, etc., while stop manoeuvres include *decelerate-to-stop*, *wait-for-oncoming*, and *wait-for-lead-to-cross*. Therefore, we use different initial speeds based on what manoeuvre the SOV intends to carry out. If the SOV is in a straight-through lane, we use initial speeds of 13 m/s and 2 m/s for go and stop manoeuvres respectively. For left-turn lanes we use initial speeds of 5 m/s and 1 m/s for go and stop manoeuvres respectively. For right-turn lanes we use initial speeds of 8 m/s and 2 m/s for go and stop manoeuvres respectively. We provide a full description of the conditional speed distributions used for each lanelet in appendix B.

4.4 Step 4: Hypergame Construction and DOR Estimation

In this section, we explain how we simulate the occlusion situations we generated in the previous step and how we estimate the DOR measure. We first illustrate the process using the occlusion situation generated in the previous section (see figures 4.2c and 4.6b) and then summarize the steps to replicate our approach in algorithm 2.

In order to compute the DOR measure for the occlusion situation shown in figure 4.2c, we need to check the outcomes of the level-0 and level-1 hypergames instantiated from the occlusion situation. We first consider the level-0 hypergame, where all vehicles by definition have perfect vision of each other. The available actions for each vehicle are determined based on the criteria outlined in table A.1. Broadly speaking, turning vehicles have the option to proceed with their turn, or wait for oncoming vehicles to pass, while straight-through vehicles can drive through the intersection or wait for turning vehicles to make their turn. If a vehicles has a lead vehicle then they can also choose to follow the lead vehicle with a follow manoeuvre.

We use backwards induction to solve the hierarchical game. This means that all vehicles first play the trajectory-level game to determine which trajectories to use for their manoeuvres. Selected trajectories from the trajectory level are then used to represent manoeuvres at the manoeuvre level. The solution to the manoeuvre-level game determines which manoeuvre each vehicle uses when simulating the occlusion situation.

For each manoeuvre available to each vehicle we generate 50 cubic spline trajectories. The path of the trajectory is fit to the centreline of the lane in which the vehicle is traveling. To represent a wide range of driving behaviours we sample the middle and end speeds for

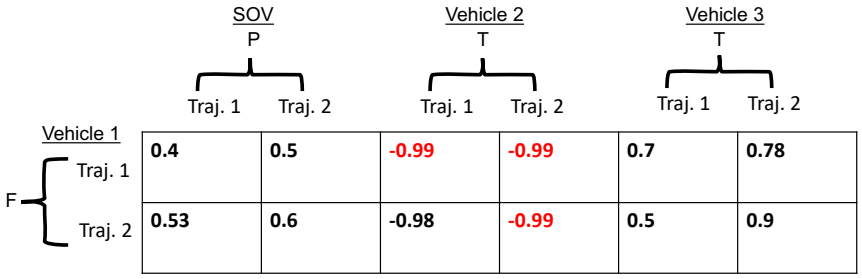
each trajectory from a range of speed values. For example, for vehicle 3's T manoeuvre, we generate 50 trajectories. Vehicle 3's initial speed is 43 km/h while the speed limit is 50 km/h. Therefore the objective for this manoeuvre is to have vehicle 3's terminal speed be closer to 50 km/h. To that end, we sample middle trajectory speeds between 45-48 km/h and terminal velocities around 49 and 52 km/h. Trajectories are then generated that try to hit these speed targets while maintaining a comfortable level of acceleration and jerk.

To further account for differences in driving behaviour we allow for lateral variation in the trajectories meaning that vehicles do not need to stick directly to their lane's centreline path. However, having 50 potential trajectories for each manoeuvre would make playing the trajectory-level game computationally expensive and so we filter the set of 50 trajectories down to 3 trajectories based on a set of criteria depending on if the trajectory is for a go manoeuvre or stop manoeuvre. We want the 3 trajectories we select to be an expressive set over the possible behaviours the driver might take. Therefore for go-manoeuvre trajectories we order the trajectories based on their terminal speed and select the trajectories with the minimum, middle, and maximum terminal speeds. For stop-manoeuvre trajectories, we instead order the trajectories based on how many seconds it takes for the vehicle to reach a terminal velocity of 0 m/s. We then select the stop trajectories that represent the minimum, middle, and maximum durations to reach a terminal speed of 0 m/s.

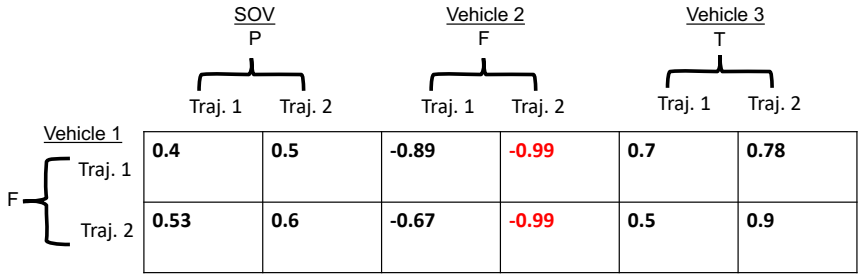
Now each vehicle has a set of actions and a set of 3 trajectories that can represent each action. To determine which trajectory to represent each action, we play the trajectory-level game. In our work we use the non-strategic maxmin decision rule to determine which trajectory to select. The maxmin decision rule makes worst-case assumptions about the utility of each trajectory and selects the maximum utility trajectory under these assumptions. Equation 4.2 shows how the trajectory, $\mathcal{T}^*_{m_i}$, which represents vehicle i's manoeuvre, m_i, is selected. The notation $-i$ refers to all other agents besides i.

$$\mathcal{T}^*_{m_i} = \underset{\forall \mathcal{T}_{m_i}}{\operatorname{argmax}} \Big(\underset{\forall \mathcal{T}_{m_{-i}}}{\operatorname{argmin}} \Big(U(\mathcal{T}_{m_i}, \mathcal{T}_{m_{-i}}) \Big) \Big) \tag{4.2}$$

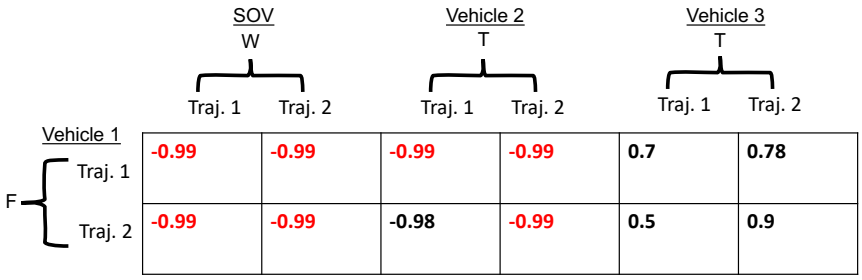
We demonstrate how the trajectory-level game is played by showing an example where vehicle 1 (i.e., the subject vehicle) uses the maxmin rule to select the trajectory for its F manoeuvre in the level-0 hypergame (figures 4.8a-4.8d). Importantly, which trajectory vehicle 1 selects for F depends on the maneouvres that the SOV, and vehicles 2 and 3 execute. Therefore, there are four cases to consider, which are covered in the four figures. In figure 4.8a, the task is to select vehicle 1's trajectory given the SOV, vehicle 2, and vehicle 3 perform manoeuvres P, T, and T respectively. Each box shows the combined safety and progress utility for one of vehicle 1's trajectories.



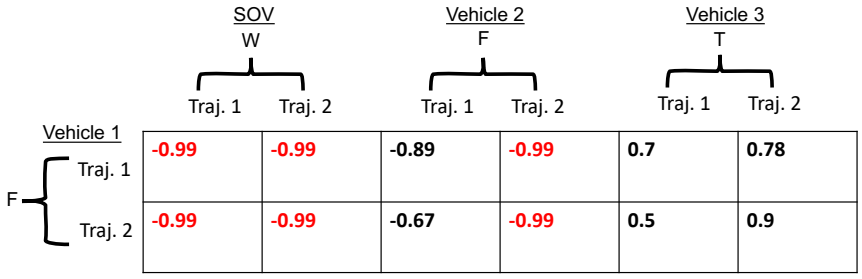
(a)



(b)



(c)



(d)

Figure 4.8: An example of trajectory selection for the subject vehicle's follow-lead (F) manoeuvre. Red utilities represent the minimum utility values for that row.

Each utility value is a function of another vehicle's trajectory because the safety utility

takes the minimum distance between both trajectories. If vehicle 1 uses trajectory 1, the minimum utility is -0.99 indicating a collision. This is because both vehicle 2's trajectories intersect with vehicle 1's trajectory resulting in the minimum safety value. Therefore, the effective utility for vehicle 1's trajectory 1 is -0.99. Taking the minimum utility from the second row we get -0.99 as well. In this case both trajectories have a minimum utility value of -0.99 and so the maxmin utility value between these trajectories is -0.99, and either trajectory can be selected to be the representative trajectory (we randomly select one of the two).

Using the maxmin decision rule allows us to model the worst-case outcomes that can result from choosing either of vehicle 1's trajectories. This same process is repeated for figures 4.8b-4.8d to arrive at the trajectory vehicle 1 should select for its F manoeuvre given the SOV, vehicle 2, and vehicle 3 perform manoeuvres P, F, and T (figure 4.8b), W, T, and T (figure 4.8c), and W, F, and T (figure 4.8d). Coincidentally, the minimum utility values for vehicle 1's F manoeuvre is consistently -0.99. This is because if the SOV (vehicle 1's leader) decides to perform W, vehicle 1 would end up rear-ending the SOV. If the SOV decides to execute P, vehicle 1 could begin its turn, following the SOV, but would end up colliding with vehicle 2, no matter if vehicle 2 chooses to perform F or T. Unsurprisingly then, F is an extremely dangerous action for vehicle 1 to take. Figure 4.9 reflects that F is an unsafe manoeuvre for vehicle 1 as all the utilities consistently result in -0.99 no matter what the other vehicles decide to do. We have shown the process of computing the utilities for vehicle 1's F manoeuvre, this process is then repeated for vehicle 1's W manoeuvre and then for the SOV's, and vehicles 2's and 3's available manoeuvres. Once all utilities have been computed, we will have constructed the manoeuvre-level game tree shown in figure 4.9a.

We solve the manoeuvre-level game by computing the set of Nash equilibria. In this case there are two Nash equilibria, (W, W, P, T), and (W, W, T, T), for vehicle 1, the SOV, and vehicles 2 and 3, respectively. In order to simulate the level-0 hypergame, we select the Nash equilibrium with the highest summed utility (calculated by summing the utility values for each vehicle's manoeuvre together). Here, both equilibria have the same summed utility so we randomly select one of the two to simulate. In this case the former equilibrium was chosen, meaning that vehicle 1 stops for vehicle 2 while the SOV, already in the middle of its turn at the start of the situation, finishes its turn before there is any danger of a collision with vehicle 2. Vehicle 3 exits the intersection, in front of vehicle 2.

Simulating the level-1 hypergame is a similar process to simulating the level-0 hypergame with the only difference being that both the trajectory-level and manoeuvre-level games for vehicle i only include the vehicles visible to vehicle i.

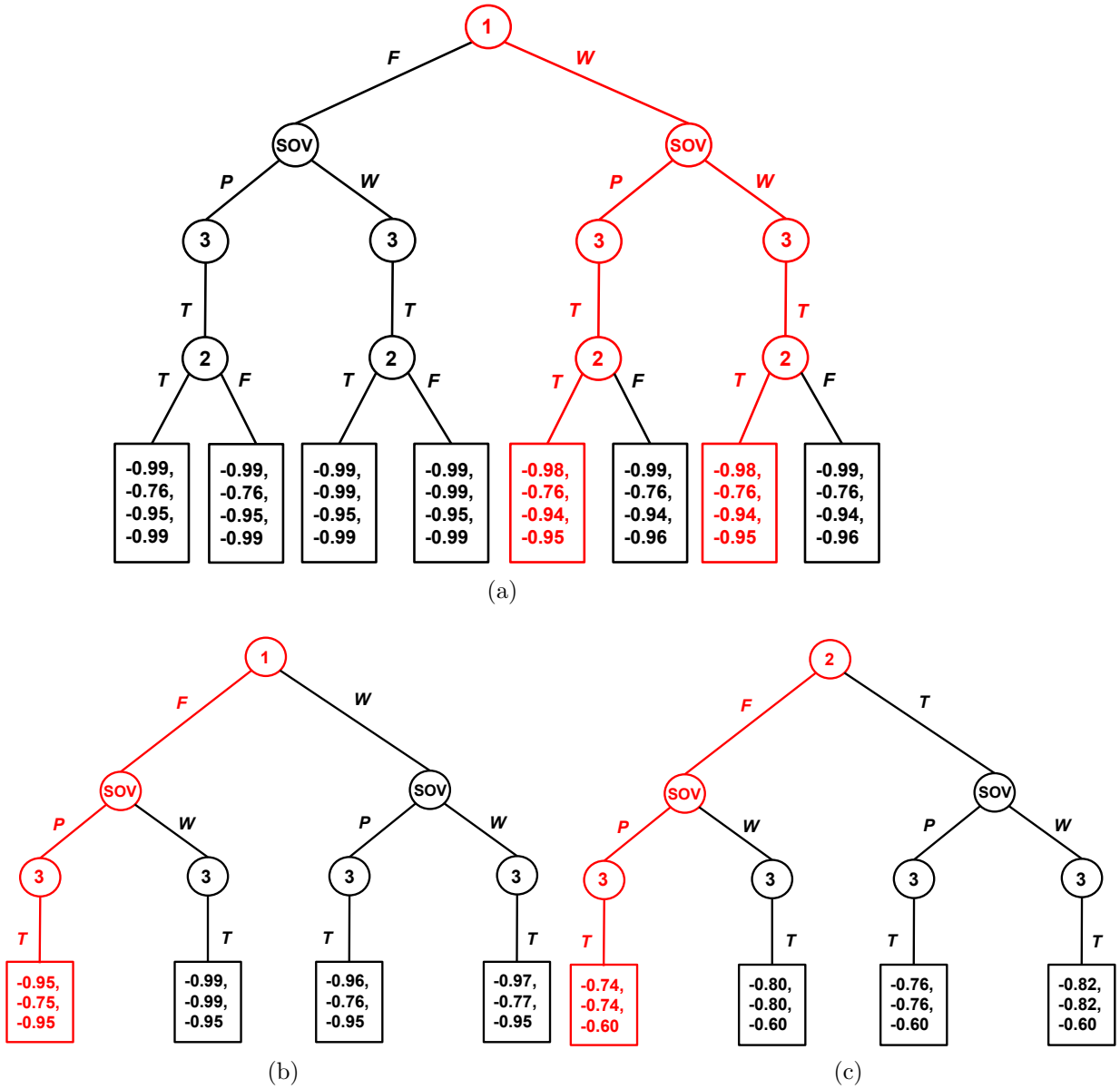


Figure 4.9: (a) The level-0 hypergame for the occlusion situation shown in figure 4.2c. (b) Vehicle 1's view in the level-1 hypergame. (c) Vehicle 2's view in the level-1 hypergame. The utilities are shown in the same order as the vehicles are listed in the game tree.

Since vehicle 1 is occluded from vehicle 2 and vice versa, we instantiate two additional game trees representing their views of the game (figures 4.9b and 4.9c). In vehicle 1's manoeuvre-level game, there is one Nash equilibrium, where vehicle 1 executes F, the SOV executes P, and vehicle 3 executes T. In vehicle 2's game, there is also one Nash equilibrium which is coincidentally the same set of actions as vehicle 1's game. Therefore, in vehicle

1's view, it's optimal action is to follow the SOV into the intersection and begin executing its turn, while in vehicle 2's game, its optimal action is to follow vehicle 3 out of the intersection. Since both the SOV and vehicle 3 have vision of all vehicles in the occlusion situation they play the game shown in figure 4.9a. The SOV chooses P, while vehicle 3 chooses T. Simulating these actions, there is a collision between vehicles 1 and 2.

In order to determine if the collision is avoidable, we replay the level-1 hypergame and have vehicles 1 and 2 execute an emergency braking manoeuvre the moment the other vehicle is no longer occluded. In this case, the emergency manoeuvres are not enough to prevent a collision. The SOV manages to finish its left-turn ahead of vehicle 2, but vehicle 1 is not so fortunate, colliding in a front-to-front collision with vehicle 2 as the former tries to complete its left turn while the latter tries to drive straight through the intersection[2]. In this occlusion situation $DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) = \mathcal{S}(\mathcal{T}_{H_0}) > 0$ since there is no collision in H_0 but there is a collision in H_1. Thus, this is an example of an OCC.

Our workflow to simulate the level-0 and level-1 hypergames for each occlusion situation generated in the previous steps and to identify OCCs using DOR is summarized in algorithm 2. The first thing done is to simulate H_0. We use the function, manoeuvres(S_o), to calculate all manoeuvres available to each vehicle for the occlusion situation, S_o, based on the criteria in appendix A. The function, trajectories(S_o, M), returns 3 trajectories for each manoeuvre in M. We then play the trajectory-level game in trajectory-game($S_o, M, \mathcal{T}$), which returns the selected trajectories $\mathcal{T}^*$ and the corresponding utilities U^*. The manoeuvre-level game is then solved in the function manoeuvre-game($S_o, M, \mathcal{T}^*, U^*$), which returns the trajectories, $\mathcal{T}_{H_0}$, which represent each vehicle's simulated manoeuvre in H_0. In order to simulate H_1, we split S_o into separate sets of vehicles, S_{v_1}, which only contain vehicles that are visible to v_1. Then, both the trajectory-level and manoeuvre games are instantiated and solved for the vehicles in S_{v_1}.

We then compute the DOR measure and check if there was an OCC. If there was an OCC we check to see if the collision can be avoided by having both colliding vehicles perform an emergency braking manoeuvre. To that end, the function emergency-braking($\mathcal{T}_{H_1,b}$) has each colliding vehicle execute an emergency braking manoeuvre the moment the other colliding vehicle becomes unoccluded. Note that we factor in a response time of 1.5 s [46] representing the time it takes from the moment occlusion ends to the moment the driver begins emergency braking. The trajectories, $\mathcal{T}_{H_1,b}$, represent the trajectories of both colliding vehicles, augmented to include emergency braking. If the collision still occurs then the occlusion situation, S_o, is added to the set of OCCs, $D^{\mathcal{C}}$. Finally, the algorithm returns $D^{\mathcal{C}}$.

[2]The simulation recording of this occlusion situation can be found at https://bit.ly/3F9fZ6Y.

Algorithm 2 OCC Identification Algorithm.

$D^{\mathcal{C}} \leftarrow \{\emptyset\}$ ▷ Initialize set of OCCs
for $S_o \in D^{\mathcal{O}}$ **do**
  $M \leftarrow \text{manoeuvres}(S_o)$
  $\mathcal{T} \leftarrow \text{trajectories}(S_o, M)$
  $\mathcal{T}^*, U^* \leftarrow \text{trajectory-game}(S_o, M, \mathcal{T})$
  $\mathcal{T}_{H_0} \leftarrow \text{manoeuvre-game}(S_o, M, \mathcal{T}^*, U^*)$ ▷ Compute manoeuvres used in H_0
  $T_{H_1} \leftarrow \{\emptyset\}$ ▷ Initialize set of H_1 trajectories
  for $v_1 \in S_o$ **do** ▷ Iterate through vehicles in occlusion situation
    $S_{v_1} \leftarrow S_o$
    for $v_2 \in S_o \setminus v_1$ **do**
      if $\exists x : O(v_1, x, v_2) = 1$ **then**
        $S_{v_1} \leftarrow S_{v_1} - \{v_2\}$ ▷ Remove occluded vehicle
      end if
    end for
    $M \leftarrow \text{manoeuvres}(S_{v_1})$
    $\mathcal{T} \leftarrow \text{trajectories}(S_{v_1}, M)$
    $\mathcal{T}^*, U^* \leftarrow \text{trajectory-game}(S_{v_1}, M, \mathcal{T})$
    $T_{v_1} \leftarrow \text{manoeuvre-game}(S_{v_1}, M, \mathcal{T}^*, U^*)$ ▷ Select v_1's manoeuvre for H_1
    $\mathcal{T}_{H_1} \leftarrow \mathcal{T}_{H_1} + \{\mathcal{T}_{v_1}\}$
  end for
  if $DOR(\mathcal{T}_{H_0}, \mathcal{T}_{H_1}, \mathcal{S}) = \mathcal{S}(\mathcal{T}_{H_0}) > 0$ **then** ▷ Compute DOR
    $\mathcal{T}_{H_1,b} = \text{emergency-braking}(\mathcal{T}_{H_1}, S_o)$ ▷ Check if collision is avoidable
    if $\mathcal{S}(\mathcal{T}_{H_1,b}) = 0$ **then**
      $D^{\mathcal{C}} \leftarrow D^{\mathcal{C}} + \{S_o\}$ ▷ Add to set of OCCs
    end if
  end if
end for
return $D^{\mathcal{C}}$

Chapter 5

Experiments and Evaluation

In order to evaluate and demonstrate the efficacy of our situation-based accelerated evaluation method we performed a variety of qualitative and quantitative analyses on our results. We first provide an overview of our experimental design, outlining the hyperparameter values used in the simulation. We then describe our experimental results. First, we compare the results from our synthetic occlusion situation generation method with naturalistic data. Next, we provide an occlusion situation taxonomy based on our results. We then demonstrate that the occlusion situations and OCCs we generate are realistic by comparing them to recorded occlusion situations from the real world. This is followed by an OCC severity analysis, where we show that OCCs tend to result in high-impact collisions with little time to react. Finally, we present use cases for our risk identification and occlusion situation generation methods and how they could fit into a larger AV safety validation pipeline.

5.1 Experiment Design

Each vehicle was designed to have the same bounding box size, with dimensions 4.1 m $\times$ 1.8 m, for length and width respectively, which is roughly the size of a small to medium-sized car in North America. We set $\epsilon = 3$, meaning that if 3 or fewer rays from vehicle i hit vehicle j, then j is considered occluded from vehicle i in our simulation. We set $\eta = 60$ degrees, which is the subject vehicle's total FOV budget.

5.2 Comparison with Naturalistic Data

As a baseline comparison to demonstrate the efficacy of our method, we compare the number of dynamic-occlusion situations generated using our method compared to the number of dynamic-occlusion situations that naturally occur in the WMA dataset (table 5.1). We check for occlusion in both cases using our 2D raycasting method, so as to make the results comparable. We find that using our method we are able to expand the number of occlusion situations in the dataset by roughly a factor of 70, as well as the number of OCCs in the dataset by a factor of 40. The OCCs that we generate can be viewed at https://bit.ly/3wHa5H1.

	Naturalistic data	Our approach
No. of dynamic-occlusion situations	1534	105,914
No. of OCCs	2	79

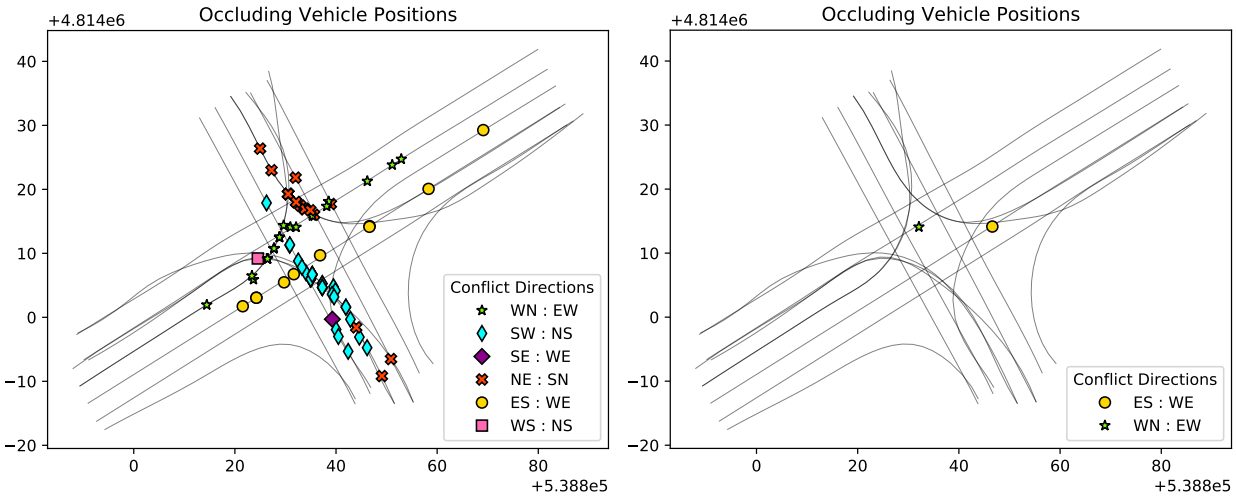
Table 5.1: Comparing the number of dynamic-occlusion situations and OCCs we generated to the number that occur in naturalistic data.

Importantly, we note that there is no recorded footage of a collision in the WMA dataset. The two OCCs that we found in the WMA dataset both occurred within our game-theoretic traffic simulation environment, which simplifies many of the environment details and thus can lead to discrepancies between what actually occurred and what the simulation predicts should occur. Even though there can be a discrepancy between simulated and recorded traffic behaviours, we show in section 5.4 that the dynamic-occlusion situations and OCCs we generate are realistic by comparing them with real-world scenarios that closely match our results.

Occlusion situations at the highest level fall into two categories, left-turn-across-path (LTAP) situations and right-turn situations (RT). Like their names suggest, LTAP situations involve vehicles making left turns in the presence of oncoming vehicles, while RT situations involve vehicles making right turns in the presence of oncoming vehicles. Table 5.1 shows that we are able to generate 79 unique[1] OCCs, which is a significant increase over the 2 that naturally occur in the WMA dataset. The generated OCCs also cover a

[1] *Unique* meaning that each occlusion situation involves a distinct configuration of vehicles. Given how we take an exhaustive approach to injecting SOVs we end up with many occlusion situations which are essentially the same, the only difference being that the SOV's position has been slightly shifted.

much more diverse range of configurations and positions as shown in figures 5.1 and 5.2. Analyzing the locations of the occlusions, Figure 5.1a shows that occluding vehicles tend to be positioned close to the center of the intersection (and along their respective straight-through path for LTAP scenarios). However, for RT scenarios, our results suggest that the occluding vehicle must be positioned at a particular location—at the end of the South-to-West (SW) left-turn lane for OCCs with conflict directions, (WS,NS), or at the end of the East-to-South (ES) left-turn lane for OCCs with conflict directions, (SE,WE). The large clusters in Figure 5.2a show that OCCs tend to occur as one of the colliding vehicles is in the process of completing its left-turn and the other colliding vehicle is crossing straight through the intersection.

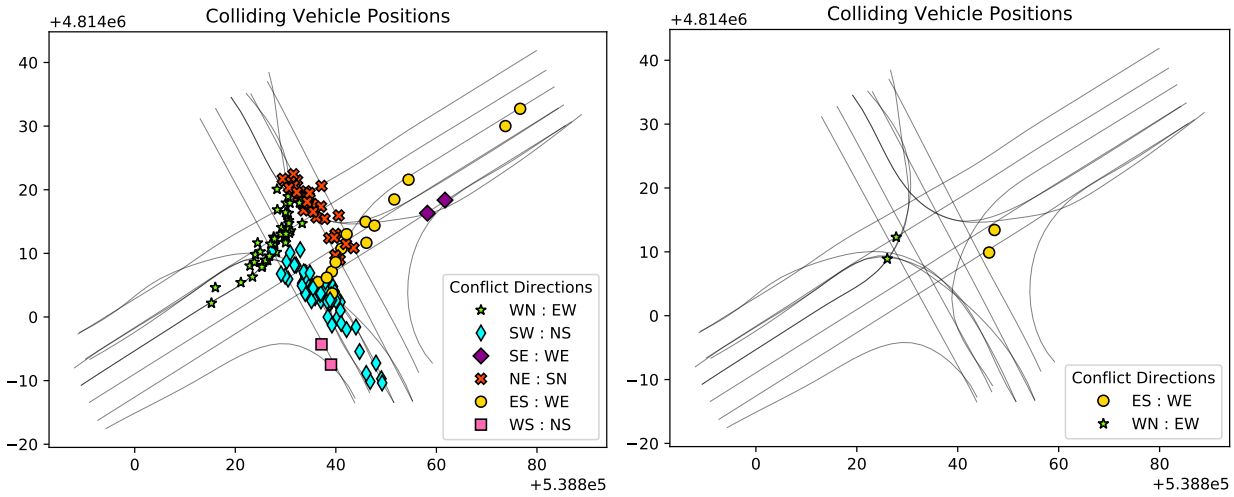


(a) (b)

Figure 5.1: (a) and (b) show the occluding vehicle positions across the 79 generated OCCs and 2 naturally occurring OCCs respectively. The conflict directions indicate the direction both occluded (colliding) vehicles were traveling.

Our results allow us to develop a sense for where occluding vehicles tend to be initially located during an OCC, as well as where OCCs tend to occur. This analysis would not be possible just using naturalistic data, since it requires a large number of diverse OCC examples.

An interesting phenomenon was that in 17 out of the 79 OCC cases, the vehicle occluding the colliding vehicles was not the SOV but rather a vehicle from the naturalistic data. The SOV did also cause occlusion in these situations (since that is the criteria we



(a) (b)

Figure 5.2: (a) and (b) show the colliding vehicle positions across the 79 generated OCCs and 2 naturally occurring OCCs respectively. The conflict directions indicate the direction both occluded (colliding) vehicles were traveling.

use to spawn SOVs), however, the occlusion the SOV causes does not lead to an OCC. What typically happens in these 17 OCCs is that the SOV itself is occluded from oncoming vehicles by a naturalistic vehicle and this results in a collision.

5.3 Categorizing Occlusion Situations

In order to better understand the nature of occlusion situations at intersections, we propose a taxonomy that decomposes occlusion situations into four levels (figure 5.3). Level 1 encompasses all occlusion situations at intersections, level 2 divides situations into RT or LTAP situations. In level 3, RT and LTAP situations are decomposed into *tag-on* and *reveal* situations. Tag-on situations refer to when a vehicle follows behind a lead vehicle. This can refer to a tailgating situation where the follower proceeds closely behind the lead vehicle, but more broadly refers to any situation where a vehicle is following another vehicle. LTAP tag-on situations can be further subdivided into LTAP *left-turn* tag-on (where only the left-turning vehicle tags on behind a lead vehicle), LTAP *straight* tag-on (where only the vehicle crossing straight through the intersection tags on behind a lead vehicle), and LTAP *both* left-turn and straight situations (where both left-turning and

straight-through vehicles tag on behind lead vehicles). We first explain the decomposition of LTAP situations and then do the same for RT situations.

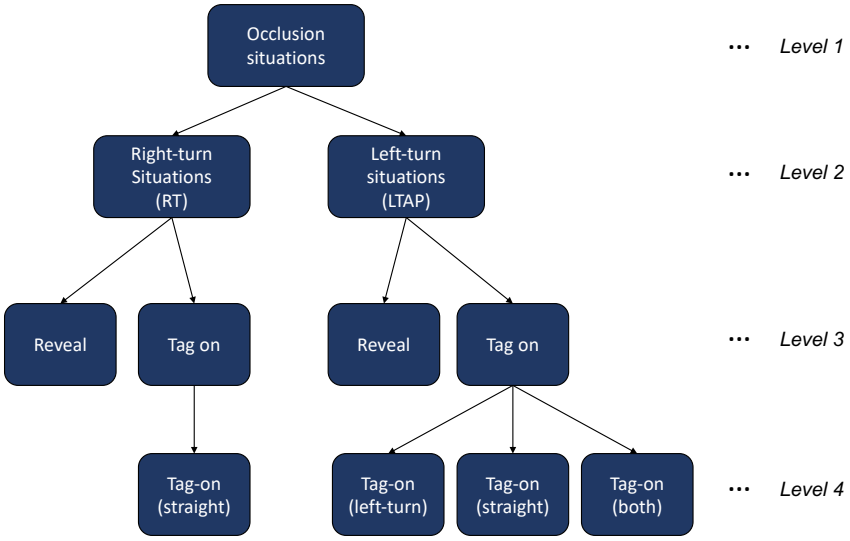
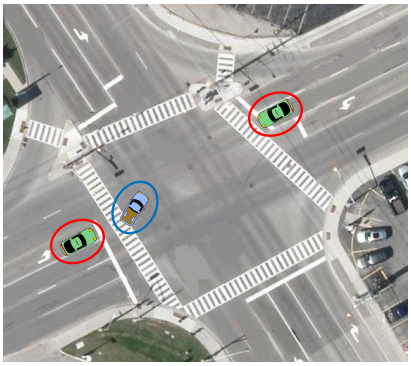
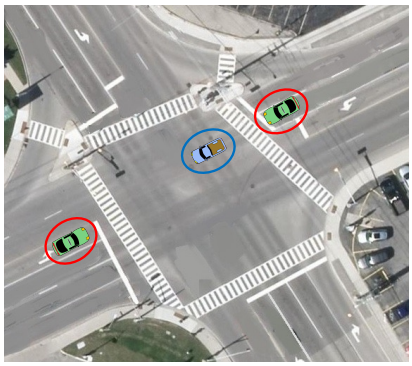


Figure 5.3: A taxonomy for occlusion situations.

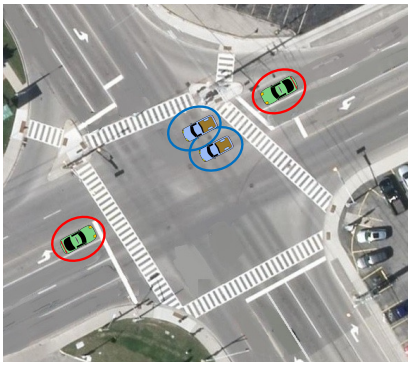
Figure 5.4a is an example of a LTAP left-turn-tag-on situation. The follower vehicle is making a left turn behind the lead vehicle, which happens to be occluding an oncoming vehicle in the top right corner of the map. Figures 5.4b and 5.4c show different configurations for a straight-tag-on situation. Here, the follower vehicle is driving straight through the intersection while following behind a lead vehicle that is occluding a vehicle making a left turn in the bottom left corner of the map. While there are different variations of this situation, the underlying idea remains the same—the lagging vehicle is following a lead vehicle straight through the intersection while being occluded (for a portion of the drive) from an oncoming left-turning vehicle. Combining the previous two ideas, we arrive at LTAP both-tag-on situations, where both left-turning and straight-through vehicles are following lead vehicles that are causing occlusion. Figure 5.4d is an example of this situation. (Note, Figure 5.4 does not cover all the variations of each occlusion situation since there are numerous ways to realize these situation categories—this is just a set of common examples.) With the exception of RT straight-tag-on situations (figure 5.4h), our method was able to discover occlusion situations that fit into each of the different categories laid out by the occlusion situation taxonomy (figure 5.5). This demonstrates that our method can produce a diverse range of occlusion situations, covering different traffic configurations.



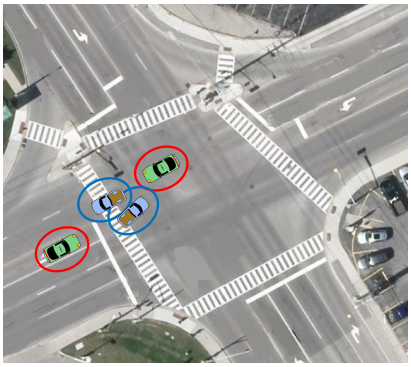
(a) LTAP: Left-turn tag on



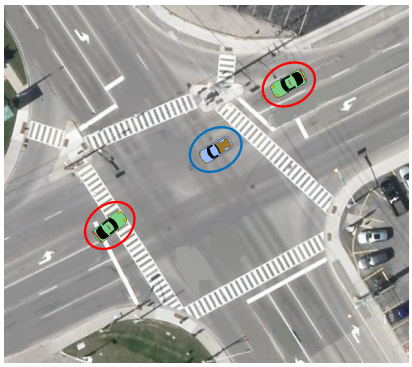
(b) LTAP: Straight tag on



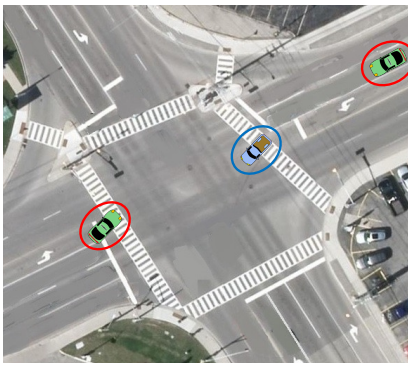
(c) LTAP: Straight tag on



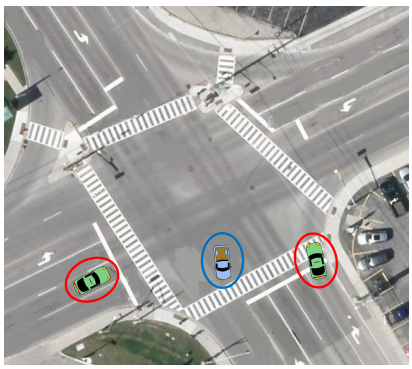
(d) LTAP: Both tag on



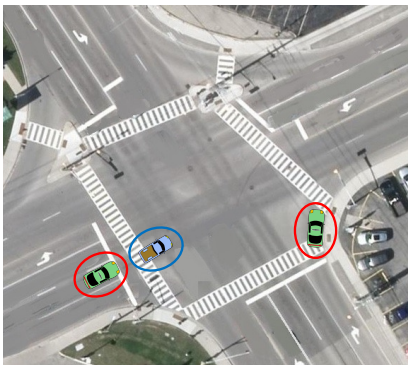
(e) LTAP: Reveal



(f) LTAP: Reveal

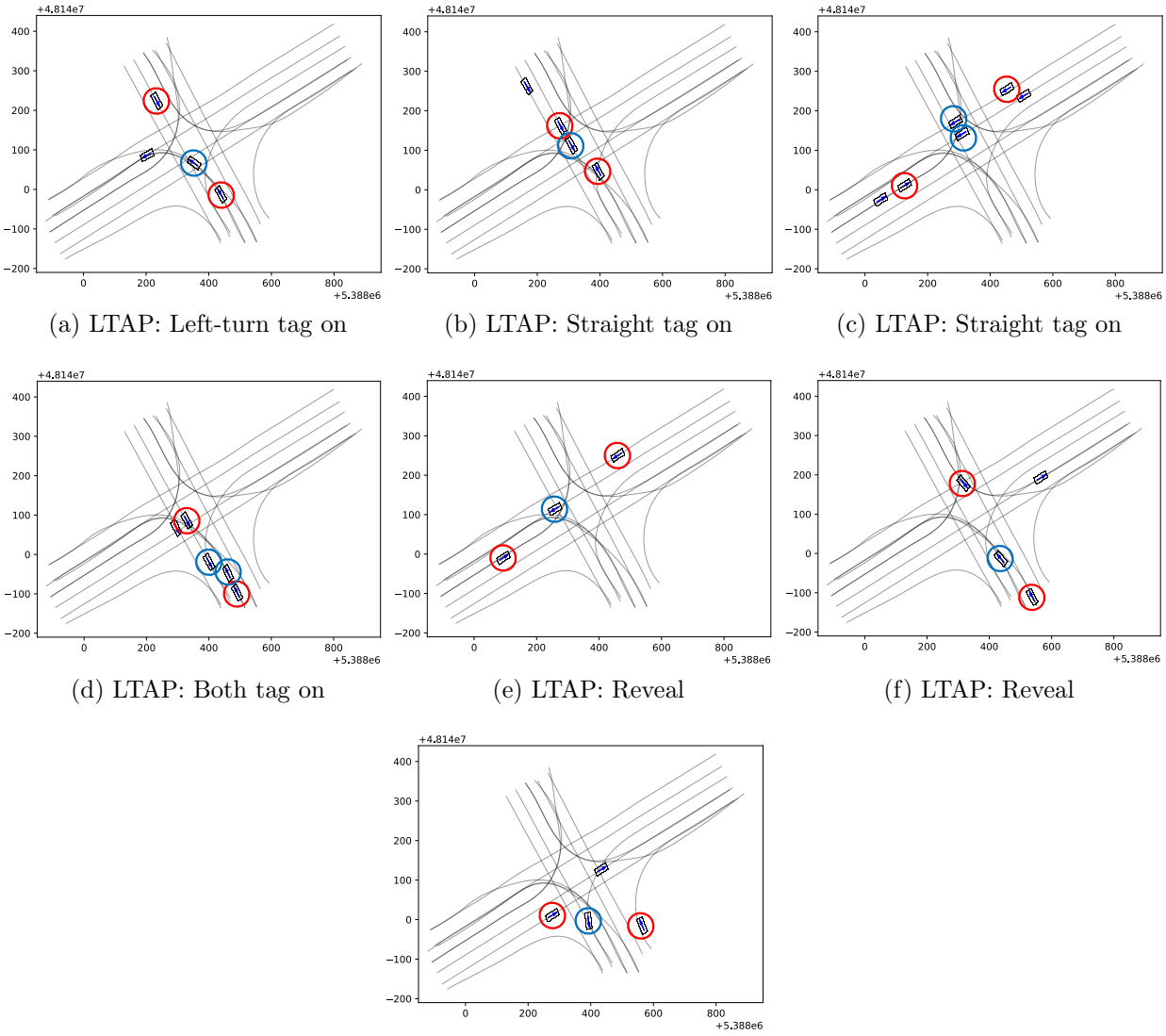


(g) RT: Reveal



(h) RT: Straight tag on

Figure 5.4: Example configurations for each of the occlusion situation categories laid out in the taxonomy in figure 5.3. The blue circles represent the SOVs, while the red circles represent the vehicles occluded from each other by the SOV.



(a) LTAP: Left-turn tag on

(b) LTAP: Straight tag on

(c) LTAP: Straight tag on

(d) LTAP: Both tag on

(e) LTAP: Reveal

(f) LTAP: Reveal

(g) RT: Reveal

Figure 5.5: Occlusion situation examples generated using our method in the same configurations as those shown in figure 5.4. The blue circles represent the SOVs, while the red circles represent the vehicles occluded from each other by the SOV.

Distinguishing occlusion situations based on whether they are a tag-on or reveal situation is useful because these two categories represent two primary driving behaviours. For tag-on situations, the follower matches their speed to the leader and tries to maintain a

minimum distance gap, whereas in reveal situations there is no leader and so the vehicle can choose to drive at a higher speed. Furthermore, in tag-on situations, the follower may pay less attention to the traffic environment because they trust that if the situation is safe for the leader, then it will be safe for them as well.

Table 5.2 shows results across the four categories, LTAP left-turn-tag-on (**Left-turn**), LTAP straight-tag-on (**Straight**), LTAP both left-turn- and straight-tag-on (**Both**), LTAP reveal (**L-Reveal**), and RT reveal (**R-Reveal**) situations. Out of the 79 OCCs, 49 were **Left-turn**, 11 were **Straight**, 6 were **Both**, 11 were **L-Reveal**, and 2 were **R-Reveal** situations.

We now cover the relationship between the occlusion situation categories covered above and the collision configurations they lead to. The collision configuration, *front-to-front*, occurs when both colliding vehicles collide head-on into each other. A similar configuration is the *angle* collision, which occurs when one of the colliding vehicles collides head-on into the side of the other colliding vehicle. A *sideswipe* collision occurs when the sides of both vehicles' glance off each other. Finally, a *front-to-rear* collision occurs when one of the colliding vehicles collides head-on into the rear end of the other colliding vehicle[2]. Both **Left-turn** and **Straight** situations have a similar distribution over collision configurations, having roughly 75% front-to-front collisions, and 25% angle collisions—and in the case of **Left-turn** situations, a small percentage of sideswipe collisions. This distribution is not surprising, since in both **Left-turn** and **Straight** situations, both vehicles are heading towards each other, so one would expect most collisions to result in either front-to-front or angle collisions.

The difference in how a front-to-front and angle collision occurs comes down to the relative position and speeds of the left-turning vehicle and oncoming straight-through vehicle. Front-to-front collisions occur as the left-turning vehicle is beginning its turn while angle collisions occur when the left-turning vehicle is already deep into its turn causing the left-turning vehicle to be at an angle to the oncoming straight-through vehicle. **Both** situations have a relatively high percentage of sideswipe collisions. Looking at figure 5.4d gives us a clue as to why sideswipe collisions occur more frequently in these situations: they are typically configured with the lead occluding vehicles in adjacent lanes and with the followers moving closely behind.

The final row in Table 5.2 shows the mean impact velocities along with their variance across the different situation categories. While there is a large spread over the impact velocities, the maximum impact velocity (second last row) for **Both** situations is much lower than the other three categories. This is likely because both vehicles are followers and

[2]Refer to Appendix F in [2] for a more detailed overview of the different collision configurations.

as such cannot accelerate to higher speeds through the intersection. Furthermore, **Both** situations have the highest mean occlusion duration, which is caused by simultaneously having two occluding vehicles in front of both colliding vehicles. Therefore, it is clear from these results that **Both** situations represent a distinct situation category apart from **Left-turn** and **Straight** tag-on situations.

Situation category	**Left-turn**	**Straight**	**Both**	**L-Reveal**	**R-Reveal**
Front-to-front Collision	38/49	8/11	3/6	4/11	0/2
Angle Collision	9/49	3/11	1/6	7/11	0/2
Sideswipe Collision	2/49	0/11	2/6	0/11	0/2
Front-to-rear Collision	0/49	0/11	0/6	0/11	2/2
Occlusion duration (s)	0.5 ± 0.2	0.6 ± 0.6	0.9 ± 0.7	0.5 ± 0.2	0.2 ± 0.0
Min. impact v (m/s)	2.1	3.4	3.2	1.3	3.8
Max. impact v (m/s)	25.9	17.5	12.0	22.3	6.5
Mean impact v (m/s)	14.8 ± 37.8	11.2 ± 18.6	8.3 ± 8.7	12.3 ± 34.1	5.2 ± 1.9

Table 5.2: A comparison between the different occlusion situation categories and their collision details. The columns represent left-turn tag-on **Left-turn**, straight tag-on **Straight**, both left-turn and straight tag-on **Both**, left-turn reveal **L-Reveal**, and right-turn reveal **R-Reveal**. Rows 1-4 show the distribution of collision configurations for each situation category. Row 5 shows the mean occlusion duration—the time it takes for both vehicles to become unoccluded from each other—and variance for each situation category. Rows 6-8 show the minimum, maximum, and mean of the impact velocities, where the impact velocity is the relative velocity between both colliding vehicles at the moment of impact.

Unlike the other situation categories, angle collisions rather than front-to-front collisions make up the majority of **L-Reveal** collisions. While it is difficult to say with certainty why **L-Reveal** situations should be so skewed towards angle collisions, the asymmetric nature of occlusions provides a possible explanation. Angle collisions are caused when the left-turning vehicle is in the middle of executing its turn when the oncoming straight-through vehicle collides with it. This could be caused by the left-turning vehicle becoming unoccluded from the straight-through vehicle first. In this case, the left-turning vehicle would continue through its turn, since it is unaware of the occluded straight-through vehicle, while the straight-through vehicle would begin emergency braking since it sees the left-turning vehicle. The result is that the left-turning vehicle would be deep into its turn when the collision occurs.

Indeed, we found that 4 out of the 7 **L-Reveal** situations that resulted in an angle

Situation category	**Left-turn**	**Straight**	**Both**	**L-Reveal**	**R-Reveal**
Front-to-front Col. (S)	17/38	3/8	3/3	0/4	0/0
Front-to-front Col. (L)	5/38	5/8	0/3	2/4	0/0
Front-to-front Col. (R)	0/38	0/8	0/3	0/4	0/0
Angle Col. (S)	4/9	1/3	1/1	4/7	0/0
Angle Col. (L)	4/9	1/3	0/1	1/7	0/0
Angle Col. (R)	0/9	0/3	0/1	0/7	0/0
Sideswipe Col. (S)	1/2	0/0	2/2	0/0	0/0
Sideswipe Col. (L)	0/2	0/0	0/2	0/0	0/0
Sideswipe Col. (R)	0/2	0/0	0/2	0/0	0/0
Front-to-rear Col. (S)	0/0	0/0	0/0	0/0	0/2
Front-to-rear Col. (L)	0/0	0/0	0/0	0/0	0/2
Front-to-rear Col. (R)	0/0	0/0	0/0	0/0	0/2
Total (S)	22/49	4/11	6/6	4/11	0/2
Total (L)	9/49	6/11	0/6	3/11	0/2
Total (R)	0/49	0/11	0/6	0/11	0/2

Table 5.3: Asymmetric occlusion across the different situation categories and decomposed for each collision configuration. The columns represent left-turn tag-on **Left-turn**, straight tag-on **Straight**, both left-turn and straight tag-on **Both**, left-turn reveal **L-Reveal**, and right-turn reveal **R-Reveal**. Here, (S) stands for straight, (L) for left turn, and (R) for right turn. For example, Front-to-front Col. (S) for **Left-turn** is 17/38 meaning that 17/38 of the front-to-front **Left-turn** OCCs resulted in the straight-through vehicle gaining vision of the left-turning vehicle first.

collision had the straight-through vehicle gain vision of the left-turning vehicle first, which supports the theory that asymmetric occlusion plays a role in causing angle collisions. However, this is not a complete explanation since 1 of the 7 angle collisions had the left-turning vehicle gain vision of the straight-through vehicle first and 2 did not have any asymmetric occlusion. This suggests that while asymmetric occlusion can be important in determining collision configuration, other factors such as the initial position and speed of the colliding vehicles still play an important role.

We were only able to generate 2 **R-Reveal** situations, making it the lowest occurring situation category in our results. **R-Reveal** situations are configured such that the occluding vehicle is in the process of completing its left turn while one of the colliding vehicles

moves straight through the intersection and the other colliding vehicle executes a right turn. The OCC is caused by the straight-through vehicle rear ending the right-turning vehicle[3]. Unsurprising, these situations result in the lowest occlusion durations across the five situation categories since there is only a small window when the occluding vehicle can block vision between the colliding vehicles. **R-Reveal** situations also result in the lowest impact velocity since they only resulted in front-to-rear collisions in our experiments. We perform a severity analysis in section 5.5, but even these initial results give AV safety designers insight into which situation categories are more dangerous and thus should be prioritized in the safety validation pipeline.

Table 5.3 provides insight into which situation category experiences the most asymmetric occlusion. From the table both **Left-turn** and **Straight** situations experience asymmetric occlusion across the different collision configurations. For **Left-turn** situations 17 out of the 38 front-to-front collisions had the straight-through vehicle gain vision of the left-turning vehicle first, this suggests that the specific positions of the vehicles in this situation category might bias this outcome. In contrast, the majority (i.e., 5/8) of **Straight** front-to-front collisions had the left-turning vehicle gain vision of the straight-through vehicle first. For **Both** situations in all collision configurations, the straight-through vehicle was able to gain vision of the left-turning vehicle first. The last three rows in table 5.3 show that situation category seems to influence the number and type of asymmetric occlusion. **Left-turn** situations are heavily skewed towards the straight-through vehicle gaining vision first, while **Straight** situations are slightly skewed towards the left-turning vehicle gaining vision first. The only asymmetric occlusion cases for **Both** situations had the straight-through vehicle gaining vision first. For **L-Reveal** the split is almost equal and there was no recorded asymmetric occlusion for **R-Reveal** situations.

While it is clear that asymmetric occlusion alone does not determine the collision configuration, it nonetheless is an important factor. Thus, understanding the relationship each situation category has to asymmetric occlusion allows us to better understand how OCCs occur.

5.4 Comparison with Real-World Occlusion Situations

We next address the question of how realistic the generated OCCs are. We want the OCCs to be realistic so that if the AV performs well using our situation-based accelerated evaluation method, it will be an indicator that it will perform well in the real world. We

[3]The two **R-Reveal** OCCs can be viewed at https://bit.ly/3ktB1Fq and https://bit.ly/3Dc4gDS.

demonstrate that our method can generate realistic occlusion situations and OCCs by showing that our results closely match real-world occlusion situations and OCCs.

Figure 5.6 show three OCCs, 5.6a-5.6c[4], 5.6d-5.6f[5], and 5.6g-5.6i[6], as well as one risky occlusion situation, 5.6j-5.6l[7], that could have led to a collision if the drivers were not cautious. For the images 5.6a-5.6i, the occluded colliding vehicles are circled in red while the occluding vehicles are circled in blue. We focus first on the first three images 5.6a-5.6c, which describe a left-turn-tag-on situation. We are able to generate this occlusion situation as can be seen in figure 5.5a. This OCC can be broken down as follows, in image 5.6a the occluding vehicle (a truck highlighted in blue) makes a left turn at the intersection, while a straight-through vehicle (highlighted in red) approaches from the other side of the road. The follower vehicle, shown in image 5.6b, is also highlighted in red. Both the follower and the oncoming straight-through vehicle are occluded from each other by the truck. Instead of waiting to gain vision of potential oncoming vehicles, the follower decides to proceed with its left turn and collides in an angle collision with the oncoming straight-through vehicle in image 5.6c.

The sequence of images, 5.6d-5.6f show a LTAP reveal situation. We have also generated this occlusion situation in our results (see figure 5.5f). In images 5.6d and 5.6e we see that a large emergency vehicle, in the process of making a left turn, causes occlusion between a left-turning vehicle on the opposite side of the road and an oncoming straight-through vehicle. Both vehicles take evasive manoeuvres but still end up colliding in image 5.6f.

The OCC shown in images 5.6g-5.6i is actually not covered in our results but can be classified as a reveal situation. The reason why our method was not able to generate this OCC is because the white truck (highlighted in red) drives through the intersection on a red light. We did not include running red lights as valid behaviour in our experiment design. In images 5.6g and 5.6h we see the truck (highlighted in blue) making a right turn and occluding both colliding vehicles (highlighted in red), which are making their way into the intersection. In image 5.6i they collide in an angle collision. While this situation differs from the other situations we generated due to the truck running the red light, we can draw similarities to our results. For example, if the occluding vehicle was instead making a left turn from the opposite side of the road, and the black vehicle was making a right turn instead of proceeding straight through the intersection, the situation would look exactly like the RT reveal situation as shown in figure 5.5g.

[4] https://youtu.be/jDEZ-igoDgw?t=634
[5] https://youtu.be/tDN-mwNSJc8?t=44
[6] https://youtu.be/fQv43M6PcWs?t=71
[7] https://youtu.be/Qk7ejm8MlEc?t=94

Figure 5.6: Real-world occlusion situations. Examples (a)-(i) result in collisions. The red circles in these images represent the colliding vehicles, which are occluded from each other. The blue circle represents the occluding vehicle. The sequence of images, (j)-(l), show a risky occlusion scenario recorded in the real world that could have resulted in a collision if either driver was not cautious. The red circles here represent vehicles that were fully occluded and are a potential dynamic-occlusion risk.

While we only show the one variation of the RT reveal situation in figure 5.4g, this

real-world occlusion situation suggests another variation where the occluding vehicle is not making a left turn in front of the straight-through vehicle but is instead making a right turn in the adjacent lane to the straight-through vehicle.

Finally, images 5.6j-5.6l show another LTAP reveal situation, involving a large transport vehicle causing oncoming left-turn vehicles to be occluded. In images 5.6j and 5.6k, a previously occluded left-turning truck (highlighted in red) decides to make a left turn even though it did not have vision of the oncoming-straight-through vehicle (the camera view is from the oncoming-straight-through vehicle's perspective). A collision is avoided as the following left-turn vehicle (highlighted in red) in image 5.6l decides to wait before proceeding. Given the size of these trucks it is easy to see how occlusion situations can easily occur in the real world. One drawback of our occlusion checking approach, being a 2D raycasting algorithm, is that it treats vehicles as infinitely tall obstacles that rays cannot pass through. However, situations like these translate well into our simulation environment, since the occluding vehicle is tall enough to effectively act as an infinitely tall obstacle that AV sensors and human vision cannot pass through.

From these real-world examples, we can see that our method is able to generate occlusion situations that precisely resemble or at least have similar characteristics to what occurs in the real world. Ultimately, what makes an occlusion situation realistic or not are factors such as how we model occlusion and our assumptions as to how vehicles behave under occlusion. We make simplifying assumptions here such as using 2D raycasting to model vision, as well as assuming that occluded vehicles do not exist from the perspective of other vehicles. Such assumptions could be changed depending on how best to model the traffic environment. For example, instead of using 2D raycasting one could model an AV perception system using front- and side-mounted cameras as well as a top-mounted LiDAR system in a 3D environment. Such a system would be much more costly to simulate but would result in a much more realistic implementation of how an AV sees its environment. Similarly, human vision could be modeled using 3D raycasting. Nonetheless, we see from these examples that these simpler assumptions are in certain circumstances good enough to model the real world. As shown in images 5.6a-5.6c, human drivers sometimes make the mistake of thinking that an occluded oncoming vehicle doesn't exist. Furthermore, occlusion situations like the one shown in images 5.6j-5.6l demonstrate that 2D raycasting can be an appropriate method to model occlusion in the real world.

5.5 Severity Analysis

We perform a severity analysis on the 79 OCCs, in order to evaluate how dangerous OCCs tend to be. We calculate how severe an OCC is by extracting the relative velocity between the colliding vehicles and mapping the value to the ranges shown in table 5.4. The severity class ranges are based on tables 2 and 3 in Krampe and Mirko's injury modelling work [64]. Specifically, table 5.4 provides the injury risk to belted front-seat occupants involved in a front-to-rear collision with a leading vehicle, and also applies to belted occupants seated on the near-side of an angle collision. The ranges have also been cross-checked with the injury model proposed by Jurewicz et al. [52], which estimates the probability of an injury with a severity level above 3, as a function of $\triangle v$, on the AIS injury scale. The AIS scale is shown in table 5.5, which maps each AIS level (1-6) to an injury level, with 1 representing minor injuries and 6 representing maximal injuries. The model proposed by Jurewicz et al. predicts that at roughly 30 km/h the probability of an AIS level 3 or higher injury (MAIS3+) is approximately 10% for frontal and nearside collisions. This roughly corresponds with S2 in our categorization of severity injury level in table 5.4. An impact velocity of 40 km/h results in a roughly 50% chance of a MAIS3+ injury, and roughly corresponds to S3.

Severity Class	**Velocity Range (m/s)**	**Velocity Range (km/h)**
S0	$[0, 5.3]$	$[0, 19]$
S1	$(5.3, 7.7]$	$[19, 28]$
S2	$(7.8, 10.3]$	$[28, 37]$
S3	$(10.3, \infty)$	$[37, \infty]$

Table 5.4: Mapping relative velocity between colliding vehicles to severity class (S0-S3). S0 is the least severe type of collision and S3 is the most severe. The ranges are based on injury models outlined by Krampe and Mirko [64], Jurewicz et al. [52], and the SAE Functional Safety Committee [40].

We note that both Krampe and Mirko's injury severity model [64] as well as the injury model proposed by Jurewicz et al. [52] use $\triangle v$, which measures the change in velocity between both colliding vehicles before and after the collision. Instead of using $\triangle v$, we choose to use the relative velocity at impact between both colliding vehicles, which for vehicles of the same mass, is roughly half the value of $\triangle v$ [18]. The value of $\triangle v$ approaches the relative velocity if one vehicle's mass is much greater than the other's (such as when a small car collides with a large bus). Therefore, using relative velocity we naturally end

up with higher severity measurements when analysing collisions. From a safety validation standpoint, we can do this because what we end up measuring represents an upper bound on OCC severity, and so we can conclude that in the real world the actual severity of these OCCs would be at most as severe as what we measure.

AIS Code	Injury	Example	AIS Prob. of Death (%)
1	Minor	Superficial Laceration	0
2	Moderate	Fractured Sternum	1-2
3	Serious	Open Fracture of Humerus	8-10
4	Severe	Perforated Trachea	5-50
5	Critical	Ruptured Liver with Tissue Loss	5-50
6	Maximum	Total Severance of Aorta	100

Table 5.5: The AIS injury scale [30], which defines a 6 level scale for injury severity ranging from 1 for minor injuries and 6 for injuries that result in death.

The distribution of severity classes across the 79 OCCs is shown in figure 5.7. Roughly 10% were classified as S0, 10% as S1, 13% S2, and 67% as S3. That many of the OCCs were front-to-front collisions with a severity level of S3 is not surprising, given that out of the 79 OCCs, 55 were front-to-front collisions, which frequently result in high impact velocities. However, this does highlight the critical need for AVs to be properly evaluated under these types of occlusion situations.

In addition to the high likelihood of a severe collision, OCCs are dangerous because they do not allow much time for either driver to respond to the imminent collision. Figure 5.8 shows the distribution of durations from the moment both colliding vehicles are no longer occluded to the moment of impact, across the 79 OCCs. The distribution ranges from 0.5 s to 3.0 s with a mean value of 1.65 s. The range of driver response time is typically between 0.8 s to 2.5 s with a mean response time between 1.3 s to 1.5 s [35, 21, 67]. Based on the average response times then, drivers only have between 0.15 s and 0.35 s to decelerate or perform an evasive manoeuvre before a collision. Therefore, proactively identifying and addressing potential occlusion situations *before* they occur is critical for ensuring driver safety.

We also analyze the occlusion duration length to get a sense for how long occlusions generally last. Figure 5.9a shows the range of times it took for both occluded vehicles to gain vision of each other. We see that the distribution is skewed towards lower durations, with a peak at 0.2 s. The mean occlusion duration for both vehicles was 0.57 s. The

longest occlusion duration recorded was 2.6 s, which was a result of a situation where both occluded vehicles were following lead occluding vehicles that were moving slowly and so the occlusion lasted longer. The average occlusion duration being only 0.57 s suggests that the occlusion duration does not need to be long in order for a dangerous occlusion situation to arise.

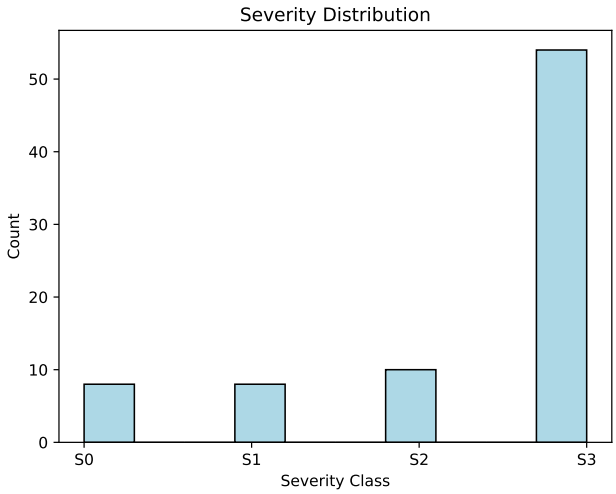


Figure 5.7: Severity distribution across the 79 OCCs.

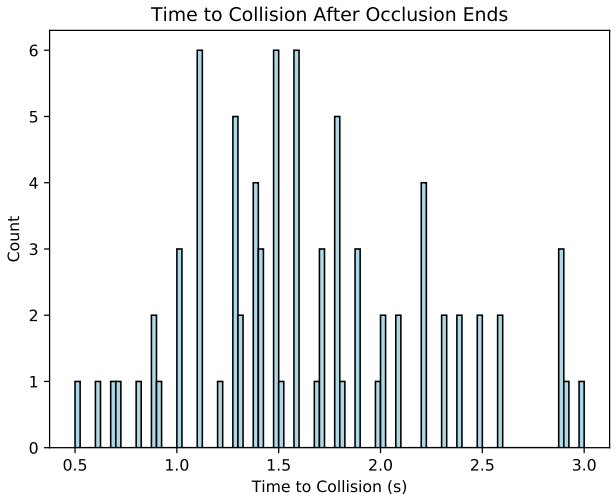
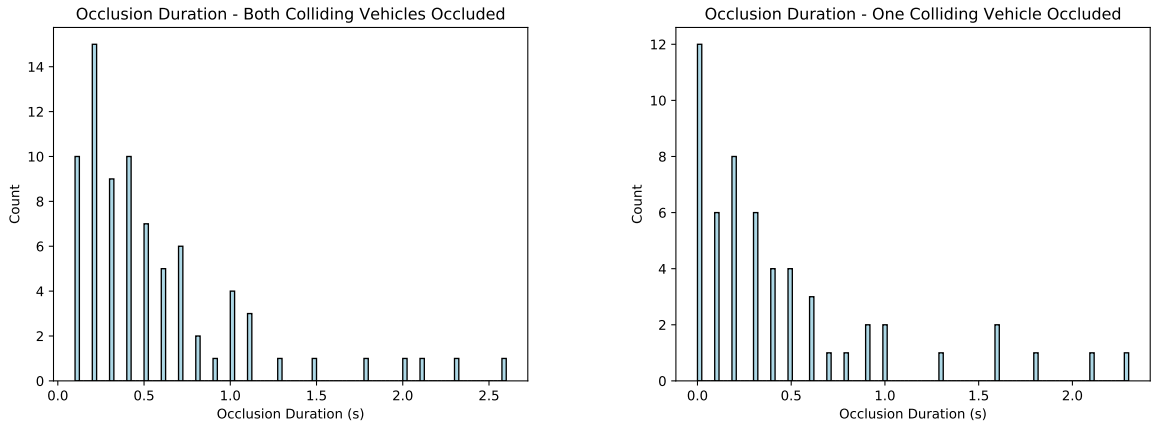


Figure 5.8: The distribution of time to collision after occlusion ends.

Out of the 79 OCCs, 54 had asymmetric occlusion. Figure 5.9b shows the range of times it took from the moment both vehicles were occluded to the moment one of the vehicles became unoccluded from the other. The mean occlusion duration was 0.46 s. In 12 cases the occlusion duration was 0 s meaning one of the colliding vehicles never actually lost vision of the other vehicle. It seems odd that an OCC would occur even though one of the vehicles had vision of the other vehicle the whole time. The explanation is that in our simulation the vehicle with vision incorrectly assumes the other colliding vehicle also has vision of them. This phenomenon highlights the need to not only reason over other driver's preferences, but also to correctly model the world from the other driver's perspective, since they may be make decisions that seem irrational to others but are rational given their perspective of the environment. In other words, even if no vehicle is occluded from the driver personally, in order for them to safely navigate occlusion situations, they must be occlusion-aware.



(a) (b)

Figure 5.9: (a) Shows the range of times it takes for both vehicles to gain vision of each other. (b) If one of the occluded vehicles gained vision of the other occluded vehicle first we record the duration of time it took for the first vehicle to gain vision and include it in this distribution. Both distributions are over the 79 OCCs we generate.

Finally, we present the distribution of lead-follower distances from each tag-on OCC (see figure 5.10). The mean lead-follower distance is 11.1 m. If we assume that the speed limit while traveling through the intersection is 50 km/h and the minimum time gap between vehicles is 2 s, then followers should ideally be around 28 m behind the leader. Of course,

many of the situations included in this distribution are of left-turning followers and leaders in which case the distance gap will naturally be lower. Furthermore, the skew towards low lead-follower distances likely is a result of the lead-follower distances from each tag-on OCC, where in order for the follower to be occluded from oncoming vehicles, it must be tagging-on closely behind the lead vehicle.

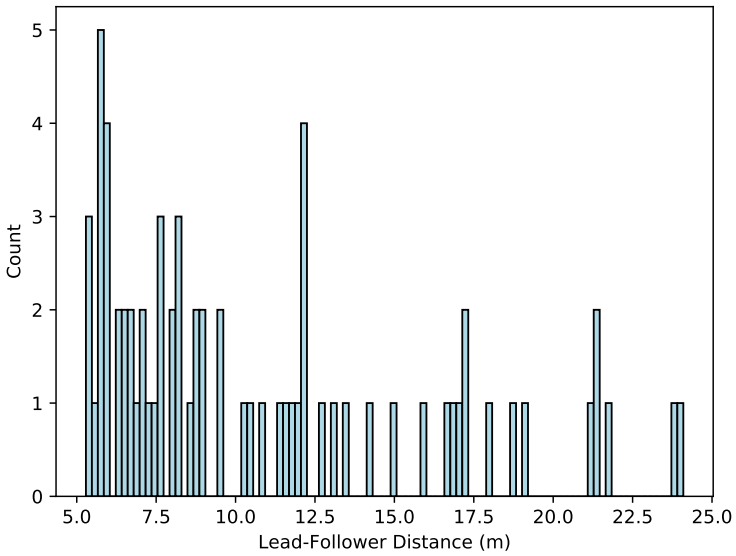


Figure 5.10: Lead-follower distances based on the initial positions of the lead and follower vehicles across each tag-on OCC.

5.6 Use Cases

In this section we outline the two primary use cases we foresee for our proposed method. The first use case is as an offline situation-based accelerated testing method for strategic planners where multiple planners can be tested in tandem in traffic environments with dynamic occlusion. We have already outlined this method in chapter 4. The second use case is as an online occlusion-aware predictive reasoning engine that works in real-time to predict how other agents will behave in situations with and without occlusion.

Using the method presented in chapter 4 as an offline situation-based accelerated testing method comes naturally from the workflow. We note that our approach is also planner-agnostic meaning that it can be extended to any type of strategic planner with just the knowledge of the solution concept. A more sophisticated strategic planner could reason

over the possibility of occluded vehicles or about other drivers' perspectives of the driving environment. We also note that using a low-fidelity 2D environment for simulation is actually an advantage rather than a drawback. This is because we see our method as simply the first step in a larger safety validation pipeline. Using our method, failure cases can be quickly generated in low-fidelity without having to incur the computational costs associated with running high-fidelity simulation. We envision the next step in the safety validation pipeline to be to recreate the occlusion situations that lead to OCCs in a high-fidelity environment with realistic vehicle models, where we can verify if these occlusion situations lead to OCCs in that environment too. A similar idea of bootstrapping AV failure examples in low-fidelity and then translating them into a high-fidelity environment is proposed by Koren [58]. While some of the candidate failure cases generated in low fidelity may end up being spurious errors in high fidelity, the idea is to generate a large enough pool of candidates that at least some candidates translate into genuine failures or near-failures in high fidelity.

Our second proposed use case is to use our hypergame-based simulation and DOR estimation methods as a real-time occlusion-aware predictive reasoning engine. This idea relies upon the assumption that the subject AV, using traffic cameras and other nearby AVs on the road, will be able to construct a near occlusion-free view of their driving environment using vehicle-to-X (V2X) communication. However, AVs will still need to reason about how human drivers—who do not have access to such an omnispective state—will behave in occlusion situations. Assuming the subject AV has access to each human driver's world view (for example by simulating the human driver's view of the environment), the AV can act as an occlusion-aware agent, using our game-theoretic framework to predict what human drivers (i.e., occlusion-naive agents) will do even in the presence of occlusion.

Chapter 6

Threats to Validity

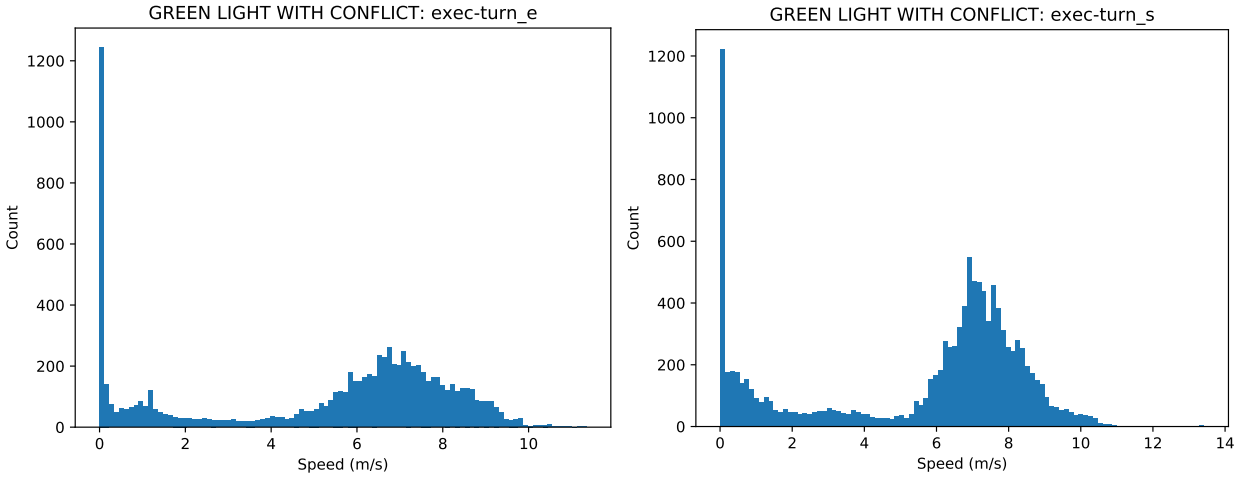
In this chapter we identify threats to the validity of our findings. These threats cover issues that may have affected our results including the number and diversity of our results, and also cover the possibility that some OCCs may be spurious errors when checked in higher fidelity simulation. We split the threats into two categories, internal threats, which deals with aspects of our implementation that could have impacted our results, and external threats, which deals with how ideas we did not implement could have impacted our results.

6.1 Internal Threats

These are threats to the validity of our results that emerge from the particular way we implemented our method. In order to cover a wide range of driving behaviours, we generate trajectories with varying degrees of lateral offset. However, this lateral offset can be quite large at times, so much so that the vehicle can end up crossing the lane boundary into the lane running adjacent in the opposite direction. This is likely how some of the sideswipe collisions in our results occur. Sideswipe collisions do occur in the real-world, so whether or not trajectories with a large lateral offset are realistic is something we still need to investigate. Ultimately, the challenge is making sure that the OCCs we generate in low-fidelity translate into OCCs in high-fidelity. Another factor that may prevent this from happening is that we only use emergency braking to avoid collisions. In contrast, a study done by Scanlon et al. [96] found that most drivers perform a combination of steering and braking to avoid a collision in an intersection environment. Thus, our simulation step should ideally incorporate evasive manoeuvres with steering in addition to braking.

Another limitation of our driver behaviour modelling stems from our definition of the safety utility as the minimum distance gap between two trajectories. Driving in an intersection frequently forces vehicles to drive close to each other, but these situations are also not necessarily dangerous. Indeed, vehicles drive adjacent to each other at high speeds through intersections, highways, and other types of public roads every second of every day, and these situations are not considered dangerous. For example, the minimum gap may occur in the middle of two vehicles executing their trajectories, but since the trajectories do not intersect there is no actual conflict between them. So while the safety utility would be extremely low, say -0.97, there would not actually be any threat of a collision. However, since the safety utility is lower than 0, we use it to represent the total utility (see eq 3.1). As a result, in reality this pair of trajectories may be completely safe but the vehicles do not choose to use them because they appear dangerous using our safety utility definition. A potential fix would be to record pairs of trajectories where there is a collision and give these pairs low safety utilities, while giving pairs of trajectories where there is no collision a high safety utility score.

In addition to what has already been discussed, another limitation of our work centres around modelling each lanelet's conditional speed distributions in the intersection. Our goal was to split the speed data into conditional distributions that represent distinct driver behaviour, however, these conditional distributions did not always capture a single mode of driving behaviour. For example, figures 6.1a and 6.1b show that the conditional speed distribution for green light with conflict for lanelets `exec-turn_e` and `exec-turn_s` follow a bimodal distribution comprised of an exponential decay distribution and normal distribution. These bimodal distributions suggest that the "green light with conflict" condition can be further decomposed. What appear to be happening here is that these distributions are capturing both wait and go turning behaviours, i.e., i) waiting for the oncoming vehicle to pass before making their turn, or ii) quickly making the turn before the oncoming vehicle gets too close. Thus, the exponential distribution near 0 m/s represents this waiting behaviour while the normal distribution centred around 7 m/s likely represents this go behaviour. Since we were not able to capture a single driving behaviour in these conditional distributions we did not use them to sample speeds. Instead, we manually set each SOV's initial speed to 8 m/s for go manoeuvres and 2 m/s for stop manoeuvres.



(a) (b)

Figure 6.1: Bimodality in conditional speed distributions.

Finally, we note that we treat sideswipe collisions as front-to-front collisions for the purposes of computing the relative velocity between both colliding vehicles, as well as computing severity of the collision. This is because it is difficult to say how the relative velocity in a sideswipe collision translates into our S0-S3 severity level otherwise. The best case assumption is that a sideswipe collision results in no injuries because both colliding vehicles lightly glance off each other. However, from a safety point of view we would rather have an upper bound on the potential severity of sideswipe collisions and so therefore we instead make the worst-case assumption, treating sideswipe collisions as front-to-front collisions. Thus, our results represent an upper bound on the actual severity of OCCs in the real world.

6.2 External Threats

While these can be considered threats, they are more so future avenues of research for how we can improve the implementation of our method in the future. The first limitation we highlight involves the notion that we base each vehicle's behaviour in simulation solely on the Nash equilibrium manoeuvres with the maximum combined utility values. However, vehicles in the real world likely base their decisions not only off of the Nash equilibrium but also on the rules of the road. If there is a conflict between the Nash equilibrium

and rule-based manoeuvres, they will likely choose the rule-based manoeuvre since that is the safer choice. For example, a left-turning vehicle may find that its Nash equilibrium manoeuvre is to execute its left turn (they cannot see that there is an occluded oncoming vehicle), whereas the rule-based manoeuvre dictates that they should wait to make sure the way is clear before making their left-turn. Nash-equilibrium-based decision making and rule-based decision making represent two different driver behaviours. Most drivers use both decision frameworks, with rule-based decisions having priority over the former. However, it is clear that drivers do not always prioritize rule-based manoeuvres as can be seen from the real-world OCCs we cover in figure 5.6. Nonetheless, future work should incorporate this feature when modelling driver behaviour.

Some OCCs that occur in our simulation may be avoidable in the real world for another reason, while occlusion is always present at the beginning of each occlusion situation we generate, our current implementation fails to account for the fact that as vehicles approach the intersection from afar they may see other vehicles that may become occluded in the future. Therefore, it is possible that in some of occlusion situations we simulate, vehicles would know of the existence of the occluded vehicles and would therefore anticipate their presence. With that being said, there are occlusion situations, such as in figures 5.6j-5.6l where there is no opportunity to gain sight of potentially occluded vehicles before entering the intersection. Rather than simulating forward in time from the start of the occlusion situation, one solution to this problem would be to instead simulate backwards in time and check at each previous timestep which vehicles are visible to each other. This would then inform the behaviour of the vehicles in the forward simulation. This can be viewed as a future research endeavour, building off of what we have proposed here. Where in this work we have shown a variety of occlusion situations that lead to OCCs, the next endeavour is to look into the prerequisite conditions that allow occlusion situations to form in the first place.

Finally, we highlight a challenge that does not only pertain to this work but to all simulation-based approaches when going from low-fidelity to high-fidelity. Namely, certain failure modes only emerge in a high-fidelity environment. For example, failures in the system's perception system due to specific lighting or weather. However, we want to emphasize that our workflow, incorporating occlusion situation generation and evaluation of strategic planners using hypergames is the key takeaway of this work, rather than our specific implementation. As computers become more capable of handling higher and higher computational loads, what is considered high-fidelity today will likely become low-fidelity in 10-20 years.

Chapter 7

Conclusion

In this work we have proposed a situation-based accelerated testing method for strategic planners. Our proposed method includes i) extracting partial traffic scenes from naturalistic data, ii) injecting occluding vehicles into extracted scenes to create realistic occlusion situations, and iii) efficiently identifying high-risk occlusion situations using a novel DOR measure.

Our results provide us with rich insight into the nature of dynamic occlusion at intersections. Along with characterizing where OCCs tend to occur in the intersection, we also provide a taxonomy for dynamic occlusion at intersections, categorizing occlusion situations at the high level into LTAP or RT situations and at a lower level into tag-on or reveal situations. A severity analysis of our results reveals that OCCs tend to lead to high-severity collisions with little time for drivers to react and perform an evasive manoeuvre.

Strategic planners are a powerful tool to perform multi-agent decision making, and have the potential to significantly improve the way AVs make decisions in complex multi-agent traffic environments. Unfortunately there are little to no resources for how to properly evaluate strategic planners so that they are safe for public use. In this work, we sought to address this problem by focusing on the subtask of safety validation under conditions of dynamic occlusion in an intersection environment. Additionally, to the best of our knowledge, this is the first work to incorporate the theory of hypergames into game-theoretic occlusion-aware traffic modelling. Our hope is that this thesis can be a stepping stone for future work in this area, with the ultimate goal of creating safe, reliable, and comfortable AV technology for everyone.

References

[1] Houssam Abbas, Matthew O'Kelly, Alena Rodionova, and Rahul Mangharam. Safe at any speed: A simulation-based test harness for autonomous vehicles. In *International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems*, pages 94–106. Springer, 2017.

[2] National Highway Traffic Safety Administration et al. Mmucc guideline: Model minimum uniform crash criteria fourth edition.(report no. dot hs 811 631). *Washington, DC: National Highway Traffic Safety Administration*, 2012.

[3] Amir Al Jawahiri. *Spline-based Trajectory Generation for Autonomous Truck-Trailer Vehicles in Low Speed Highway Scenarios*. PhD thesis, Delft University of Technology, 2018.

[4] Vassili Alexiadis, James Colyar, John Halkias, Rob Hranac, and Gene McHale. The next generation simulation program. *Institute of Transportation Engineers. ITE Journal*, 74(8):22, 2004.

[5] Matthias Althoff and Sebastian Lutz. Automatic generation of safety-critical test scenarios for collision avoidance of road vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1326–1333. IEEE, 2018.

[6] John Amanatides, Andrew Woo, et al. A fast voxel traversal algorithm for ray tracing. In *Eurographics*, volume 87, pages 3–10, 1987.

[7] Rabah Amir and Isabel Grilo. Stackelberg versus cournot equilibrium. *Games and Economic Behavior*, 26(1):1–21, 1999.

[8] Michał Antkiewicz, Maximilian Kahn, Michael Ala, Krzysztof Czarnecki, Paul Wells, Atul Acharya, and Sven Beiker. Modes of automated driving system scenario testing: Experience report and recommendations. *SAE International Journal of Advances and Current Practices in Mobility*, 2(2020-01-1204):2248–2266, 2020.

[9] Claudine Badue, Rânik Guidolini, Raphael Vivacqua Carneiro, Pedro Azevedo, Vinicius B Cardoso, Avelino Forechi, Luan Jesus, Rodrigo Berriel, Thiago M Paixao, Filipe Mutz, et al. Self-driving cars: A survey. *Expert Systems with Applications*, 165:113816, 2021.

[10] Kaushik Basu. Stackelberg equilibrium in oligopoly: an explanation based on managerial incentives. *Economics Letters*, 49(4):459–464, 1995.

[11] Peter G Bennett. Toward a theory of hypergames. *Omega*, 5(6):749–751, 1977.

[12] Peter G Bennett. Bidders and dispenser: manipulative hypergames in a multinational context. *European Journal of Operational Research*, 4(5):293–306, 1980.

[13] Peter G Bennett and Malcolm R Dando. Complex strategic analysis: a hypergame study of the fall of france. *Journal of the Operational Research Society*, 30(1):23–32, 1979.

[14] Peter G Bennett and Chris S Huxham. Hypergames and what they do: A 'soft or'approach. *Journal of the Operational Research Society*, 33(1):41–50, 1982.

[15] PG Bennett and MR Dando. Fall gelb and other games: a hypergame perspective of the fall of france, 1940. *Journal of the Conflict Research Society*, 1(2):1–33, 1977.

[16] PG Bennett, MR Dando, and RG Sharp. Using hypergames to model difficult social issues: an approach to the case of soccer hooliganism. *Journal of the Operational Research Society*, 31(7):621–635, 1980.

[17] PG Bennett, Chris S Huxham, and MR Dando. Shipping in crisis: A trial run for 'live'application of the hypergame approach. *Omega*, 9(6):579–594, 1981.

[18] George M. Bonnett. Stiffness coefficients - energy and damage. https://bit.ly/3lWJSjY, 2001.

[19] Maxime Bouton, Alireza Nakhaei, Kikuo Fujimura, and Mykel J Kochenderfer. Scalable decision making with sensor occlusions for autonomous driving. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 2076–2081. IEEE, 2018.

[20] Sebastian Brechtel, Tobias Gindele, and Rüdiger Dillmann. Probabilistic decision-making under uncertainty for autonomous driving using continuous pomdps. In *17th international IEEE conference on intelligent transportation systems (ITSC)*, pages 392–399. IEEE, 2014.

[21] Nancy L Broen and Dean P Chiang. Braking response times for 100 drivers in the avoidance of an unexpected obstacle as measured in a driving simulator. In *Proceedings of the Human Factors and Ergonomics Society Annual Meeting*, volume 40, pages 900–904. SAGE Publications Sage CA: Los Angeles, CA, 1996.

[22] James Bucklew. *Introduction to rare event simulation*. Springer Science & Business Media, 2004.

[23] Rory Cellan-Jones. Uber in fatal crash had safety flaws say us investigators. *BBC*, 2019.

[24] Turner-Fairbank Highway Research Center. Intersection safety. https://bit.ly/3ASqeux, August 2021.

[25] Yun Chen, Frieda Rong, Shivam Duggal, Shenlong Wang, Xinchen Yan, Sivabalan Manivasagam, Shangjie Xue, Ersin Yumer, and Raquel Urtasun. Geosim: Realistic video simulation via geometry-aware composition for self-driving. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7230–7240, 2021.

[26] Eun-Ha Choi. Crash factors in intersection-related crashes: An on-scene perspective. Technical report, National Highway Traffic Safety Administration, 2010.

[27] Niraj Chokshi. Tesla autopilot system found probably at fault in 2018 crash. *The New York Times*, 2020.

[28] Edmund M Clarke, Thomas A Henzinger, Helmut Veith, Roderick Bloem, et al. *Handbook of model checking*, volume 10. Springer, 2018.

[29] SAE On-Road Automated Vehicle Standards Committee et al. Taxonomy and definitions for terms related to on-road motor vehicle automated driving systems. *SAE Standard J*, 3016:1–16, 2014.

[30] Wayne S Copes, Howard R Champion, William J Sacco, Mary M Lawnick, Donald S Gann, Thomas Gennarelli, Ellen MacKenzie, and Steven Schwaitzberg. Progress in characterizing anatomic injury. *The Journal of trauma*, 30(10):1200–1207, 1990.

[31] Anthony Corso, Peter Du, Katherine Driggs-Campbell, and Mykel J Kochenderfer. Adaptive stress testing with reward augmentation for autonomous vehicle validation. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 163–168. IEEE, 2019.

[32] Anthony Corso, Ritchie Lee, and Mykel J Kochenderfer. Scalable autonomous vehicle safety validation through dynamic programming and scene decomposition. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6. IEEE, 2020.

[33] Anthony Corso, Robert J Moss, Mark Koren, Ritchie Lee, and Mykel J Kochenderfer. A survey of algorithms for black-box safety validation. *arXiv preprint arXiv:2005.02979*, 2020.

[34] Florian Damerow, Tim Puphal, Yuda Li, and Julian Eggert. Risk-based driver assistance for approaching intersections of limited visibility. In *2017 IEEE International Conference on Vehicular Electronics and Safety (ICVES)*, pages 178–184. IEEE, 2017.

[35] Paweł Droździel, Sławomir Tarkowski, Iwona Rybicka, and Rafał Wrona. Drivers' reaction time research in the conditions in the real traffic. *Open Engineering*, 10(1):35–47, 2020.

[36] Jaime F Fisac, Eli Bronstein, Elis Stefansson, Dorsa Sadigh, S Shankar Sastry, and Anca D Dragan. Hierarchical game-theoretic planning for autonomous vehicles. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 9590–9596. IEEE, 2019.

[37] Melvin Fitting. *First-order logic and automated theorem proving*. Springer Science & Business Media, 2012.

[38] Niall M Fraser and Keith W Hipel. Solving complex conflicts. *IEEE Transactions on Systems, Man, and Cybernetics*, 9(12):805–816, 1979.

[39] Niall M Fraser and Keith W Hipel. Conflict analysis and bargaining. In *Proceedings of the International Conference on Cybernetics and Society*, pages 225–229. IEEE Systems, Man, and Cybernetics Society, 1980.

[40] Functional Safety Committee. Considerations for ISO 26262 ASIL Hazard Classification. Technical report, SAE International, 2018.

[41] Philipp Geiger and Christoph-Nikolas Straehle. Learning game-theoretic models of multiagent trajectories using implicit layers. *arXiv preprint arXiv:2008.07303*, 2020.

[42] Philipp Geiger and Christoph-Nikolas Straehle. Learning game-theoretic models of multiagent trajectories using implicit layers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35:4950–4958, May 2021.

[43] Philipp Geiger and Christoph-Nikolas Straehle. Learning game-theoretic models of multiagent trajectories using implicit layers. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 4950–4958, 2021.

[44] MO Giesen and PG Bennett. Aristotle's fallacy: A hypergame in the oil shipping business.... *Omega*, 7(4):309–320, 1979.

[45] Ian Graham, Fiona O'Doherty, Alan McKinnon, and Lynne Baxter. Hypergame analysis of the stability of relationships between computerbased logistics systems. *International Journal of Production Economics*, 26(1-3):303–310, 1992.

[46] Marc Green. " how long does it take to stop?" methodological analysis of driver perception-brake times. *Transportation human factors*, 2(3):195–216, 2000.

[47] Lorne Greenspan, BARRY A McLELLAN, and Helen Greig. Abbreviated injury scale and injury severity score: A scoring chart. *The Journal of trauma*, 25(1):60–64, 1985.

[48] Hamisai Hamandawana, Raban Chanda, and Frank Eckardt. Hypergame analysis and hydroconflicts in the okavango drainage basin. *Water international*, 32(4):538–557, 2007.

[49] James Thomas House and George Cybenko. Hypergame theory applied to cyber attack and defense. In *Sensors, and Command, Control, Communications, and Intelligence (C3I) Technologies for Homeland Security and Homeland Defense IX*, volume 7666, page 766604. International Society for Optics and Photonics, 2010.

[50] Constantin Hubmann, Nils Quetschlich, Jens Schulz, Julian Bernhard, Daniel Althoff, and Christoph Stiller. A pomdp maneuver planner for occlusions in urban scenarios. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 2172–2179. IEEE, 2019.

[51] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. Navigating occluded intersections with autonomous vehicles using deep reinforcement learning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2034–2039. IEEE, 2018.

[52] Chris Jurewicz, Amir Sobhani, Jeremy Woolley, Jeff Dutschke, and Bruce Corben. Exploration of vehicle impact speed–injury severity relationships for application in safer road design. *Transportation research procedia*, 14:4247–4256, 2016.

[53] Nidhi Kalra and Susan M Paddock. Driving to safety: How many miles of driving would it take to demonstrate autonomous vehicle reliability? *Transportation Research Part A: Policy and Practice*, 94:182–193, 2016.

[54] Danial Kamran, Carlos Fernandez Lopez, Martin Lauer, and Christoph Stiller. Risk-aware high-level decisions for automated driving at occluded intersections with reinforcement learning. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pages 1205–1212. IEEE, 2020.

[55] Joost-Pieter Katoen. The probabilistic model checking landscape. In *Proceedings of the 31st Annual ACM/IEEE Symposium on Logic in Computer Science*, pages 31–45, 2016.

[56] Alonzo Kelly and Bryan Nagy. Reactive nonholonomic trajectory generation via parametric optimal control. *The International Journal of Robotics Research*, 22(7-8):583–601, 2003.

[57] Carlo Kopp. Shannon, hypergames and information warfare. *Journal of Information Warfare*, 2(2):108–118, 2003.

[58] Mark Koren. *Approximate Methods for Validating Autonomous Systems in Simulation*. PhD thesis, Stanford University, 2021.

[59] Mark Koren, Saud Alsaif, Ritchie Lee, and Mykel J Kochenderfer. Adaptive stress testing for autonomous vehicles. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1–7. IEEE, 2018.

[60] Markus Koschi and Matthias Althoff. Set-based prediction of traffic participants considering occlusions and traffic rules. *IEEE Transactions on Intelligent Vehicles*, 6(2):249–265, 2020.

[61] Markus Koschi, Christian Pek, Sebastian Maierhofer, and Matthias Althoff. Computationally efficient safety falsification of adaptive cruise control systems. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2879–2886. IEEE, 2019.

[62] Nicholas S Kovach. *A temporal framework for hypergame analysis of cyber physical systems in contested environments*. PhD thesis, Airforce Institute of Technology, 2016.

[63] Nicholas S Kovach, Alan S Gibson, and Gary B Lamont. Hypergame theory: a model for conflict, misperception, and deception. *Game Theory*, 2015, 2015.

[64] Jonas Krampe and Mirko Junge. Injury severity for hazard & risk analyses: calculation of iso 26262 s-parameter values from real-world crash data. *Accident Analysis & Prevention*, 138:105321, 2020.

[65] Maxime Leclerc, Brahim Chaib-Draa, et al. Hypergame analysis in e-commerce: A preliminary report. Technical report, CIRANO, 2002.

[66] Minchul Lee, Myoungho Sunwoo, and Kichun Jo. Collision risk assessment of occluded vehicle based on the motion predictions using the precise road map. *Robotics and Autonomous Systems*, 106:179–191, 2018.

[67] Neil D Lerner. Brake perception-reaction times of older and younger drivers. In *Proceedings of the human factors and ergonomics society annual meeting*, volume 37, pages 206–210. SAGE Publications Sage CA: Los Angeles, CA, 1993.

[68] Ben Lewis-Evans. *Testing models of driver behaviour*. University Library Groningen][Host], 2012.

[69] Changjian Li and Krzysztof Czarnecki. Urban driving with multi-objective deep reinforcement learning. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pages 359–367, 2019.

[70] Lening Li, Haoxiang Ma, Abhishek N Kulkarni, and Jie Fu. Dynamic hypergames for synthesis of deceptive strategies with temporal logic objectives. *arXiv preprint arXiv:2007.15726*, 2020.

[71] Nan Li, Ilya Kolmanovsky, Anouck Girard, and Yildiray Yildiz. Game theoretic modeling of vehicle interactions at unsignalized intersections and application to autonomous vehicle control. In *2018 Annual American Control Conference (ACC)*, pages 3215–3220. IEEE, 2018.

[72] Sisi Li, Nan Li, Anouck Girard, and Ilya Kolmanovsky. Decision making in dynamic and interactive environments based on cognitive hierarchy theory, bayesian inference, and predictive control. In *2019 IEEE 58th Conference on Decision and Control (CDC)*, pages 2181–2187. IEEE, 2019.

[73] Qin Lin, Sicco Verwer, and John Dolan. Learning a safety verifiable adaptive cruise controller from human driving data. *arXiv preprint arXiv:1910.13526*, 2019.

[74] Xiao Lin, Jiucai Zhang, Jin Shang, Yi Wang, Hongkai Yu, and Xiaoli Zhang. Decision making through occluded intersections for autonomous driving. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2449–2455. IEEE, 2019.

[75] Sarah M Loos, André Platzer, and Ligia Nistor. Adaptive cruise control: Hybrid, distributed, and now formally verified. In *International Symposium on Formal Methods*, pages 42–56. Springer, 2011.

[76] Oded Maler and Dejan Nickovic. Monitoring temporal properties of continuous signals. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems*, pages 152–166. Springer, 2004.

[77] Stephen G McGill, Guy Rosman, Teddy Ort, Alyssa Pierson, Igor Gilitschenski, Brandon Araki, Luke Fletcher, Sertac Karaman, Daniela Rus, and John J Leonard. Probabilistic risk metrics for navigating occluded intersections. *IEEE Robotics and Automation Letters*, 4(4):4322–4329, 2019.

[78] Umberto Michieli and Leonardo Badia. Game theoretic analysis of road user safety scenarios involving autonomous vehicles. In *2018 IEEE 29th Annual International Symposium on Personal, Indoor and Mobile Radio Communications (PIMRC)*, pages 1377–1381. IEEE, 2018.

[79] John A Michon. A critical view of driver behavior models: what do we know, what should we do? In *Human behavior and traffic safety*, pages 485–524. Springer, 1985.

[80] Patiphon Narksri, Eijiro Takeuchi, Yoshiki Ninomiya, and Kazuya Takeda. Deadlock-free planner for occluded intersections using estimated visibility of hidden vehicles. *Electronics*, 10(4):411, 2021.

[81] Justin Norden, Matthew O'Kelly, and Aman Sinha. Efficient black-box assessment of autonomous vehicle safety. *arXiv preprint arXiv:1912.03618*, 2019.

[82] Santi Novani and Kyoichi Kijima. Symbiotic hypergame analysis of value co-creation process in service system. In *2010 7th International Conference on Service Systems and Service Management*, pages 1–5. IEEE, 2010.

[83] Bureau of Transportation Statistics. Motor vehicle fatalities, vehicle-miles, and associated rates by highway functional system. https://bit.ly/3kNOY1t, 2019.

[84] Norio Okada, Keith W Hipel, and Yoshiharu Oka. Hypergame analysis of the lake biwa conflict. *Water Resources Research*, 21(7):917–926, 1985.

[85] Matthew O'Kelly, Aman Sinha, Hongseok Namkoong, Russ Tedrake, and John C Duchi. Scalable end-to-end autonomous vehicle testing via rare-event simulation. *Advances in Neural Information Processing Systems*, 31:9827–9838, 2018.

[86] OpenAI. CarRacing-v0. https://bit.ly/3mqgqC6, August 2021.

[87] Piotr F Orzechowski, Annika Meyer, and Martin Lauer. Tackling occlusions & limited sensor range with set-based safety verification. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 1729–1736. IEEE, 2018.

[88] Kareem Othman. Public acceptance and perception of autonomous vehicles: a comprehensive review. *AI and Ethics*, pages 1–33, 2021.

[89] André Platzer and Jan-David Quesel. Keymaera: A hybrid theorem prover for hybrid systems (system description). In *International Joint Conference on Automated Reasoning*, pages 171–178. Springer, 2008.

[90] Sasinee Pruekprasert, Xiaoyi Zhang, Jérémy Dubut, Chao Huang, and Masako Kishida. Decision making for autonomous vehicles at unsignalized intersection in presence of malicious vehicles. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2299–2304. IEEE, 2019.

[91] Xin Qin, Nikos Aréchiga, Andrew Best, and Jyotirmoy Deshmukh. Automatic testing and falsification with dynamically constrained reinforcement learning. *arXiv preprint arXiv:1910.13645*, 2019.

[92] Stefan Riedmaier, Thomas Ponn, Dieter Ludwig, Bernhard Schick, and Frank Diermeyer. Survey on scenario-based safety assessment of automated vehicles. *IEEE access*, 8:87456–87477, 2020.

[93] AK Said and DA Hartley. A hypergame approach to crisis decision-making: The 1973 middle east war. *Journal of the Operational Research Society*, 33(10):937–948, 1982.

[94] Atrisha Sarkar and Krzysztof 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)*, pages 6407–6414. IEEE, 2019.

[95] Atrisha Sarkar and Krzysztof Czarnecki. Solution concepts in hierarchical games under bounded rationality with applications to autonomous driving. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 5698–5708, 2021.

[96] John M Scanlon, Kristofer D Kusano, and Hampton C Gabler. Analysis of driver evasive maneuvering prior to intersection crashes using event data recorders. *Traffic injury prevention*, 16(sup2):S182–S189, 2015.

[97] Markus Schratter, Maxime Bouton, Mykel J Kochenderfer, and Daniel Watzenig. Pedestrian collision avoidance system for scenarios with occlusions. In *2019 IEEE Intelligent Vehicles Symposium (IV)*, pages 1054–1060. IEEE, 2019.

[98] Johann M Schumann. *Automated theorem proving in software engineering*. Springer Science & Business Media, 2001.

[99] Wilko Schwarting, Alyssa Pierson, Javier Alonso-Mora, Sertac Karaman, and Daniela Rus. Social behavior for autonomous vehicles. *Proceedings of the National Academy of Sciences*, 116(50):24972–24978, 2019.

[100] Shai Shalev-Shwartz, Shaked Shammah, and Amnon Shashua. On a formal model of safe and scalable self-driving cars. *arXiv preprint arXiv:1708.06374*, 2017.

[101] Michael C Shupe, William M Wright, Keith W Hipel, and Niall M Fraser. Nationalization of the suez canal: a hypergame analysis. *Journal of Conflict Resolution*, 24(3):477–493, 1980.

[102] Aman Sinha, Matthew O'Kelly, Russ Tedrake, and John C Duchi. Neural bridge sampling for evaluating safety-critical autonomous systems. *Advances in Neural Information Processing Systems*, 33, 2020.

[103] Nigel W Stokes and Keith W Hipel. Conflict analysis of an export credit trade dispute. *Omega*, 11(4):365–376, 1983.

[104] Liting Sun, Mu Cai, Wei Zhan, and Masayoshi Tomizuka. A game-theoretic strategy-aware interaction algorithm with validation on real traffic data. In *IROS*, pages 11038–11044, 2020.

[105] Sarah M Thornton, Francis E Lewis, Vivian Zhang, Mykel J Kochenderfer, and J Christian Gerdes. Value sensitive design for autonomous vehicle motion planning. In *2018 IEEE Intelligent Vehicles Symposium (IV)*, pages 1157–1162. IEEE, 2018.

[106] Ran Tian, Sisi Li, Nan Li, Ilya Kolmanovsky, Anouck Girard, and Yildiray Yildiz. Adaptive game-theoretic decision making for autonomous vehicle control at roundabouts. In *2018 IEEE Conference on Decision and Control (CDC)*, pages 321–326. IEEE, 2018.

[107] Ran Tian, Liting Sun, Masayoshi Tomizuka, and David Isele. Anytime game-theoretic planning with active reasoning about humans' latent states for human-centered robots. *arXiv preprint arXiv:2109.12490*, 2021.

[108] Hugo H Van der Molen and Anton MT Bötticher. A hierarchical risk model for traffic participants. *Ergonomics*, 31(4):537–555, 1988.

[109] M Wang, KW Hipel, and NM Fraser. Hypergame analysis of the falkland islands crisis. In *tech. rep.. Department of Systems Design Engineering*. University of Waterloo, 1987.

[110] Muhong Wang, Keith W Hipel, and Niall M Fraser. Modeling misperceptions in games. *Behavioral Science*, 33(3):207–223, 1988.

[111] Jackie Wattles. Tesla on autopilot crashed when the driver's hands were not detected on the wheel. *CNN*, 2019.

[112] William M Wright, Michael C Shupe, Niall M Fraser, and Keith W Hipel. A conflict analysis of the suez canal invasion of 1956. *Conflict Management and Peace Science*, 5(1):27–40, 1980.

[113] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. Occlusion-aware risk assessment for autonomous driving in urban environments. *IEEE Robotics and Automation Letters*, 4(2):2235–2241, 2019.

[114] Zixu Zhang and Jaime F Fisac. Safe occlusion-aware autonomous driving via game-theoretic active perception. *arXiv preprint arXiv:2105.08169*, 2021.

[115] Ding Zhao, Henry Lam, Huei Peng, Shan Bao, David J LeBlanc, Kazutoshi Nobukawa, and Christopher S Pan. Accelerated evaluation of automated vehicles safety in lane-change scenarios based on importance sampling techniques. *IEEE transactions on intelligent transportation systems*, 18(3):595–607, 2016.

APPENDICES

Appendix A

Available Manoeuvres in the Traffic Game

Outlined in table A.1 are the set of manoeuvres each vehicle can use in our traffic game simulation. The manoeuvres can be grouped into two broad categories, go manoeuvres, such as proceed-turn, track-speed, follow-lead, and follow-lead-into-intersection, and stop manoeuvres, such as wait-for-oncoming, decelerate-to-stop, and wait-for-lead-to-cross. Go manoeuvres allow the vehicle to make progress along its global path towards its goal location. Stop manoeuvres allow vehicles to avoid a dangerous situation by yielding to oncoming traffic. Stop manoeuvres can also be used to avoid rear ending lead vehicles.

The description column outlines the manoeuvres each vehicle can use. In general, followers can use follow and non-follow go manoeuvres. The primary difference is the speed target for follow manoeuvres is set to the lead vehicle's initial speed, whereas, non-follow go manoeuvres set the target speed to the intersection's speed limit.

Available Manoeuvres	Description
Wait-for-oncoming (W)	This applies to left- and right-turning vehicles. Wait for an oncoming vehicle to clear the intersection.
Proceed-turn (P)	This applies to left- and right-turning vehicles. Execute a left-turn or right-turn.
Track-speed (T)	This applies to vehicles driving straight through the intersection. Drive at the speed limit straight through the intersection.
Follow-lead (F)	This applies to vehicles driving straight through the intersection and have a lead vehicle. Follow a lead vehicle straight through the intersection, matching the lead vehicle's speed.
Decelerate-to-stop (D)	This applies to all vehicles. Decelerate to a stop in order to let an oncoming straight-through, left-turning, or right-turning vehicle to pass.
Wait-for-lead-to-cross (W)	This applies to left- and right-turning vehicles with a lead vehicle. Wait for the lead vehicle to finish executing its turn.
Follow-lead-into-intersection (F)	This applies to left- and right-turning vehicles with a lead vehicle. Follow the lead vehicle into the intersection while the lead vehicle executes its turn.

Table A.1: The set of available manoeuvres to vehicles in the traffic game.

Appendix B

Extracting Speed Distributions from the Dataset

Tables B.1 and B.2 show the conditional speed distributions used for each lanelet. The lanelets are also grouped based on their function in the intersection. For example, we use the same five conditional speed distributions for the preparatory left-turn lanelets, `prep-turn_[x]` where `[x]` can be one of `n`, `s`, `e`, or `w`. Since vehicles in this lane are making left turns we split the speed distribution into cases where there is an oncoming straight-through vehicle and cases where there is not, since this will affect the speed of the left-turning vehicle. We also distinguish cases where the left-turning vehicle has a green or yellow light, since this can affect the urgency of the turn. A final distinction is made for cases where there is a dedicated green light since even if there a conflicting vehicle in such cases the left-turning vehicle can freely proceed without having to yield. The left-turn lanelet, `exec-turn_[x]`, in which vehicles execute their left turn, directly proceeds `prep-turn_[x]`. Therefore, we use the same conditional distributions for this lanelet category. However, we do not create a dedicated green light distribution because of a lack of data. Nonetheless, the four distributions we are able to create provide adequate coverage of the factors (i.e., traffic light state and presence of a conflicting vehicle) that can affect the SOV's initial speed and allow for realistic speed sampling in this lanelet category.

The exit lanelets, `ln_[x]_-1` and `ln_[x]_-2`, are the regions that vehicles take to exit the intersection and so the primary factor affecting speed in these lanelets is the traffic light state. Similarly, the right-turn lanelets `rt_prep-turn_[x]` and `rt_exec-turn_[x]`, and the straight-through lanelets, `l_[x]_[y]_l` and `l_[x]_[y]_r` (where `[y]` is the connecting direction from `[x]`) only use green and yellow light distributions since all conflicting vehicles

have a red light and therefore should not affect the SOV's speed. Note that `l_[x]_[y]_l` refers to the *left* straight-through lanelet while `l_[x]_[y]_r` refers to the right straight-through lanelet.

The entrance lanelets for the straight-through lanes are denoted `ln_[x]_2` and `ln_[x]_3` for the right and left straight-through lanes respectively. For example, `ln_w_2` precedes `l_w_e_r`, while `ln_w_3` precedes `l_w_e_l`. Instead of just using a green light and yellow light distribution, these lanelets divide the green light speed distributions based on 5 s delay, the idea being that if the light has just changed to green most vehicles stopped at the intersection will take roughly that long to start moving (with vehicles in the front moving more quickly than vehicles in the back of the line).

Finally, the most involved lanelet category is the entrance lanelet to each left-turn lane, `ln_[x]_1`, which directly precedes `prep-turn_[x]`. We use six distributions to model this lanelet. The first two being the dedicated green light with and without a delay, which is included for the same reasons as the previous lanelet category. The other four conditional distributions are the same as `prep-turn_[x]` and `exec-turn_[x]` and are included for the same reasons. The full list of conditional speed distribution images can be found at https://bit.ly/31NBwnb.

Lanelets	Conditional Speed Distributions
prep-turn_n prep-turn_s prep-turn_e prep-turn_w	Dedicated green light, green light with conflict, green light without conflict, yellow light with conflict, yellow light without conflict.
exec-turn_n exec-turn_s exec-turn_e exec-turn_w	Green light with conflict, green light without conflict, yellow light with conflict, yellow light without conflict.
rt_prep-turn_s rt_prep-turn_w	Green light, yellow light.
rt_exec-turn_s rt_exec-turn_w	Green light, yellow light.
ln_n_-1 ln_s_-1 ln_e_-1 ln_w_-1	Green light, yellow light.
ln_n_-2 ln_s_-2 ln_e_-2 ln_w_-2	Green light, yellow light.
ln_n_2 ln_s_2 ln_e_2 ln_w_2	Green light with delay, green light without delay, yellow light.
ln_n_3 ln_s_3 ln_e_3 ln_w_3	Green light with delay, green light without delay, yellow light.

Table B.1: The right column shows the groupings of lanelets and the left column shows the conditional speed distributions created for each lanelet grouping. For example, for each exit lanelet, `ln_n_-1`, `ln_s_-1`, `ln_e_-1`, and `ln_w_-1`, a green light and yellow light speed distribution was extracted.

Lanelets	Conditional Speed Distributions
l_s_n_l l_s_n_r l_n_s_l l_n_s_r l_w_e_l l_w_e_r l_e_w_l l_e_w_r	Green light, yellow light.
ln_s_4 ln_w_4	Green light, yellow light.
ln_n_1 ln_s_1 ln_e_1 ln_w_1	Dedicated green light with delay, dedicated green light without delay, green light with conflict, green light without conflict, yellow light with conflict, yellow light without conflict.

Table B.2: Continuation of table B.1.