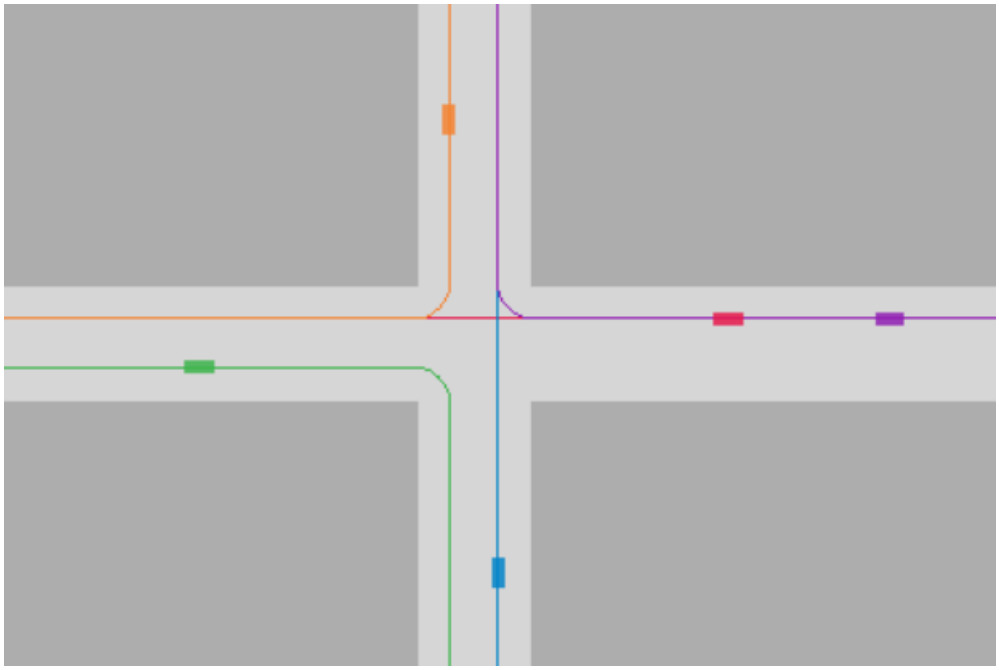


Intersection Navigation for Automated Vehicles

Risk Assessment and Maneuver Coordination

Master's thesis in Computer science and engineering

LARS NIKLASSON

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2018

Master's thesis 2018

Intersection Navigation for Automated Vehicles

Risk Assessment and Maneuver Coordination

LARS NIKLASSON

UNIVERSITY OF
GOTHENBURG

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
Chalmers University of Technology
University of Gothenburg
Gothenburg, Sweden 2018

Intersection Navigation for Automated Vehicles
Risk Assessment and Maneuver Coordination
LARS NIKLASSON

Supervisor: Elad M. Schiller, Computer Science and Engineering
Examiner: Miquel Pericas, Computer Science and Engineering

Master's Thesis 2018
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Traffic simulation of an intersection. Five vehicles with predefined trajectories are approaching the intersection from different directions.

Typeset in LaTeX
Gothenburg, Sweden 2018

Intersection Navigation for Automated Vehicles
Risk Assessment and Maneuver Coordination
LARS NIKLASSON
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This work provides a simulation environment for automated vehicles that drive through an intersection while using vehicle-to-vehicle (V2V) communications. To the end of demonstrating the proposed simulation environment, we provide software for risk assessment, which identifies dangerous situations by comparing what drivers intend to do with what they are expected to do in order to avoid collision. Our software takes into account uncertainties caused by vehicle state noise. Another way in which we demonstrate the proposed simulation environment is via software for coordinating intersection crossing in which vehicles request permissions from each other before deciding to cross, similar to the work of Casimiro and Schiller. The software is designed such that no two vehicles with conflicting maneuvers can be inside the intersection at the same time. Furthermore, we combined these two methods to create a system with multiple safety layers. The idea is that the risk assessment software works as a backup if some deviant driver does not adhere to the software for coordinating intersection crossing. The two methods are connected, such that whenever a vehicle is granted, the priority rules of the risk assessment software are temporarily changed in favor of the granted vehicle. Finally, we use the proposed evaluation environment for conducting experiments in a four-way give-way intersection. During these experiments, we observed that we can combine the risk assessment software with the coordination software. As we explain, the proposed simulation environment can help to measure to what degree collisions are avoided or mitigated, even while taking into account deviant vehicle behaviors, such as unpredictable traveling speeds, significant noise in state measurements, to name a few. We hope that the system can serve as a basis for future research projects.

Keywords: ADAS, automated vehicles, risk assessment, intersection navigation, maneuver coordination, cooperative intersection management, traffic simulation

Acknowledgements

I want to thank my supervisor Elad Schiller for providing support and guidance throughout the project. I also want to thank my colleagues Ibrahim Fayaz and Emelie Ekenstedt for their professional input and their friendship.

Lars Niklasson, Gothenburg, March 2019

Contents

1 Introduction

According to the World Health Organization, road traffic accidents cause around 1.25 million deaths each year [1]. A great majority of these accidents are, at least in part, caused by human errors, such as being distracted or driving too aggressively [2, 3]. What is more, around 20% of all fatal traffic accidents are intersection-related, despite intersections making up only a small part of the total road area [4]. Using active safety systems and cooperative intersection management (CIM), Advanced Driver Assistance Systems (ADAS) and fully autonomous vehicles (AVs) can therefore greatly increase road safety by counteracting or eliminating human driving errors at intersections.

Active safety systems in vehicles use information about other nearby vehicles, such as position, direction and speed, to assess the risk of a given traffic situation. Risk assessment is typically done by predicting vehicles' future motions and calculating the probability of a collision occurring. It can also be done by inferring drivers' intentions and detecting unexpected behaviors. If the risk is high, the system can warn the driver, or suggest the best course of action in order to avoid or mitigate potential accidents.

CIM methods can allow AVs to safely and efficiently navigate through intersections. Using vehicle-to-everything (V2X) communication, vehicles can cooperate with each other and with intersection infrastructure to coordinate the passage of vehicles. The traffic coordination can either be centralized, where a central control unit collects information, makes decisions and instructs each vehicle how to pass through the intersection, or it can be distributed, where decisions are made locally by each vehicle.

Many risk assessment and CIM methods rely on robust V2X communication and do not take into account potential deviant drivers that do not communicate or cooperate [5, 6, 7, 8, 9]. In his 2018 master's thesis [10], Fayaz uses an algorithm by Casimiro et al. [11] for exploring how risk can be avoided when vehicles share information about their intentions via V2X communications. The software integrates a risk assessment component as the basis for the vehicle coordination.

Fayaz's study was exploratory in nature and the software designed was preliminary. This thesis intends to continue this exploration and improve upon the simulation environment used by Fayaz. This simulation environment was provided to Fayaz by the author of this paper, and is part of this work. As such, the contribution of this work does not merely include improvements made to this simulation environment, but the entire development of it. The simulation environment has also been used by Casimiro et al. in a related work [12], where a fault-tolerant negotiation-based intersection crossing protocol is presented.

1.1 Related Work

There are multiple traffic simulators found in the literature, SUMO [13], VEINS [14] and GAZEBO [15] being some of them. SUMO and VEINS can handle high traffic volumes while GAZEBO can simulate complex vehicle dynamics. Hybrid approaches of these also exist [16], as well simpler ROS-based models [17]. In this work, we use a simple ROS-based simulation environment that builds on the work of Andrén et al. [17], and is specifically designed for four-way intersections.

An important part of many risk assessment methods is *intent inference*, i.e. trying to predict what maneuvers vehicles will make. Liebner et al. [18] and Lefèvre et al. [5] all use speed profiles and bayesian statistics to infer driver intentions. Zyner et al. [6] instead use a technique based on Recurrent Neural Networks. In this work, intention is inferred by comparing vehicle's states against common speed profiles and trajectories, similar to [5] and [18]. For example, if a vehicle does not slow down while approaching the intersection, we can infer that it will not turn left or right, since its speed better matches the speed profile for going straight than turning left or right. One difference is that our example compares the vehicle states against the speed profiles directly, which allows us to add additional inference rules. Such rules could, for example, be that if it is physically impossible for a vehicle to stop at the intersection due to driving to fast, we can infer that that vehicle will not stop at the intersection.

The most common way of computing traffic risk is to calculate the probability of a collision occurring. An example of such a method is the work of Ward et al. [19], who trained a SVM with time to collision (TTC) and looming data to predict future collisions between vehicles. An alternative way of computing risk is proposed by Lefèvre et al. [5]. They propose a risk assessment framework where risk is defined as the probability that the intention and expectation of a driver differs. This allows them to detect *unexpected behaviors* such as running a red light or not stopping to give way to other vehicles. In this work, we use the definition of risk proposed by Lefevre et al. [5] Our work examplify how the risk calculation can be performed in such a way as to be able to calculate the expectation *given* some intention. This is useful if we want to know if a specific maneuver is risky or not.

Vehicle-to-everything (V2X) communication is required for cooperative vehicular systems, however wireless communication is prone to failures. Morales Ponce et al. [20, 21], Savic et al. [22] and Casimiro et al. [11] all propose solutions based on distributed communication protocols that allows vehicles to safely navigate traffic in the presence of communication failures. In this work, due to the limited scope of the project, we consider only part of the solution proposed by Casimiro et al. [11]. We have omitted the membership-based cloud solution for the sake of simplicity, which allows us to more easily reason about the safety of our software solution.

A more detailed review of traffic simulators, risk assessment methods and methods for vehicle cooperation in the presence of communication failures appears in Chap-

ter 2.

1.2 Our contribution

This thesis project propose a new simulation environment for automated vehicles that drive through an intersection while using vehicle-to-vehicle (V2V) communications. The simulator allows to consider many vehicles and a variety of traffic scenarios in a four-way give-way intersection.

We demonstrate the usefulness of the simulator by considering software for risk assessment, and software for coordinating intersection crossing. Furthermore, show that the proposed simulation environment can combine these two these software components to create a system with multiple safety layers, as in Casimiro et al. [11]. We use this system to to demonstrate that the proposed simulation environment can help to measure to what degree collisions are avoided or mitigated, even while taking into account deviant vehicle behaviors, such as unpredictable traveling speeds, significant noise in state measurements, to name a few.

The test scenarios for demonstrating this work has been extended significantly from Fayaz's work [10]. We have also observed great improvements both in safety and performance compared to Fayaz's work.

We hope that the system can serve as a basis for future research projects.

2 Background Knowledge and Related Work

In this chapter, we first give related work with regards to traffic simulation. Then we give an overview of how risk assessment methods work. We analyze and discuss a few different risk assessment methods found in the literature. Specifically, we look at three methods for intent inference at intersections, and three methods for general risk computation. We also give an overview of three different general methods for cooperative intersection management. Lastly, we give related work with regards to vehicular cooperation in the presence of communication failures.

2.1 Simulation

SUMO [13] is an open source traffic simulator. It allows simulation of multimodal traffic, e.g. cars, pedestrians and public transport. SUMO is microscopic, which means that each actor is modelled explicitly and has an own route through the traffic network. SUMO has among other things been used to evaluate the performance of traffic lights.

VEINS [14] is a framework built upon SUMO and the event-based network simulator OMNET++ [23], which allows for running vehicular network simulations. VEINS offers models for realistic inter-vehicular communication.

Gazebo [15] is an open-source 3D dynamic robot simulator. Gazebo includes a wide variety of robot models and environments including vehicle models and traffic networks. Gazebo is easily integrated into ROS (Robot Operating System) [24], which is a framework that includes libraries and tools for developing robot applications.

Garzón et al. [16] proposes a tool for realistically simulating automated vehicles in scenarios with very high volume of traffic. They use a hybrid approach, where ROS is used to combine Gazebo and SUMO. Gazebo allows for complex simulation of dynamics and sensors, whereas SUMO can handle a high volume of traffic.

Andrén et al. [17] use ROS to simulate a single autonomous truck. They use a simple kinematic model to describe vehicle motion and a PID controller to follow a planned route. They use the ROS package *rviz* to visualize and interact with the simulation.

2.2 How risk assessment works

To estimate the danger of a given traffic situation, road traffic risk assessment methods usually rely on three processing steps - perception, motion prediction and risk computation. After the risk has been computed, the system can inform the (autonomous) driver.

2.2.1 Perception

First, information about other nearby vehicles has to be obtained. There are two predominant ways to achieve this.

1. Through on-board sensors such as cameras, radar and lidar. By using computer vision techniques, nearby vehicles can be detected and tracked, and their positions and directions can be estimated.
2. Using wireless network or cellular based Vehicle-to-everything (V2X) communication, vehicles can share information about themselves, such as their position and direction. In the case of an AV, maneuver intention could also be shared.

V2X communication could potentially provide more (and more accurate) information than vision-based perception methods, however the technology is not yet widely adopted. The two techniques could also be used together to further improve accuracy and make the system more robust in the presence of failures. [25]

2.2.2 Motion Prediction

Once the safety system has a perception of its surroundings, mathematical models are used to predict the future motions of vehicles, i.e. how the traffic situation will evolve. Several different types of motion prediction models exist. In a survey from 2014 [26], Lefèvre et al. classify existing models into three categories with increasing levels of abstraction:

1. Physics-based models, where the motions of the vehicles only depend on the laws of physics. These work well for, but are also limited to, short-term predictions.
2. Maneuver-based models, where the intended maneuvers of vehicles are also considered. These models are generally better at long-term prediction than the physics-based models.
3. Interaction-aware models, which also consider *other* vehicles' intended maneuvers. These models can make more accurate predictions than the others, but can suffer from high computational complexity. This may limit their usefulness.

If adopting a maneuver-based or an interaction-aware model, then an important part of the motion prediction step is to infer the intentions of other drivers. In the context of intersections, this could mean predicting whether or not a driver will stop to give way to other vehicles. Intention inference could potentially be skipped if vehicles explicitly share intentions using V2X communication.

2.2.3 Risk Computation

Risk can be thought of as the level of danger of a given situation, and is usually computed in one of the following two ways [26].

1. By detecting collisions between predicted future trajectories for the present vehicles. The risk can be estimated based on the probability that these collisions will occur. Sometimes the type and severity of the collisions are also taken into account.
2. By detecting unexpected or unusual behaviours, such as driving very slow or very fast, not slowing down before a stop sign, or driving very close to other vehicles. These behaviours might not result in collisions, but could still be potentially dangerous.

2.3 Driver intent inference at intersections

This section covers three different methods of predicting vehicle maneuvers at urban intersections found in the literature.

2.3.1 Intelligent Driver Model

Liebner et al. [18] use an explicit model called the "Intelligent Driver Model", that represents both car-following and turning behaviour, to infer driver intention. The model includes unique speed profiles for each intended maneuver. The likelihood of an intention corresponds to how well that intentions speed profile matches the observed speed. A simple Bayesian network classifier is used to perform the inference.

The model takes into account uncertainty (noise) in vehicle measurements. The motions of preceding vehicles are considered, but not those of other vehicles in the traffic scene. Classification is done transparently, probabilistically, and in real time. The approach is dependant on the availability of speed profiles for the intersection.

2.3.2 Hidden Markov Model

Lefèvre et al. [5] use a Hidden Markov Model to represent not only driver intention but also driver expectation and vehicle position. Expectation refers to what drivers are expected to do according to the traffic rules and is inferred using priority rules and gap-acceptance models, taking into account other vehicles' intentions and positions. Typical speed profiles and typical vehicle trajectories are used to predict vehicle intention and position. Since most drivers follow the traffic rules, the intention inference is biased towards adhering to the inferred expectation. For example, if the *expectation* of some

driver is to stop, then the *intention* will likely be to stop as well. A particle filter is used to perform the inference.

The model is rather complex as it is used not only for intention inference but also for risk computation, see Section 2.4.1. The fact that expectation, intention, and position inference is intertwined makes the model less flexible. It would for example be difficult to replace the position inference with a potentially more accurate Kalman Filter. Uncertainties in vehicle measurements are taken into account, and the classification is probabilistic. The inference can be computationally heavy due to the use of a particle filter and the consideration of multiple vehicles. The approach is dependant on trained speed profiles and gap-acceptance models, and it also requires robust vehicle-to-vehicle (V2V) communication.

2.3.3 Recurrent Neural Network

Zyner et al. [6] use Long Short Term Memory (LSTM) based Recurrent Neural Networks (RNN) to predict driver intention. The advantage of using such a technique is that correlation between time steps can be modelled. Instead of analyzing observations at each time step individually (as is done in the approaches described in Sections 2.3.1 and 2.3.2), a short sequence of data is considered. The network was trained using the Naturalistic Intersection Driving Dataset [27].

This neural network based model was able to correctly classify driver intention early compared to the methods described in Section 2.3.1 and 2.3.2. The reason for this might be that the complex network is able to recognize subtle clues that models based on speed profiles cannot. The classification is probabilistic, however the model does not take into account uncertainty of measurements. Other vehicles are not considered and the classification is not transparent (i.e. like a black box). Inference can be performed in real time, but is dependant on a pre-trained network.

2.4 Risk computation methods

This section covers three different general ways of computing risk found in the literature.

2.4.1 Detection of unexpected behaviour

As mentioned in Section 2.3.2, Lefèvre et al. [5] consider both the intended and expected behaviour of a driver. Specifically, they compute expected and intended longitudinal motion when approaching an intersection. Two simple behaviours are considered: to go or stop at the intersection. Risk is defined as the probability that intention and

expectation differ. An example of a risky situation would be one where a driver is expected to give way for other vehicles (stop) but still intends to enter the intersection (go).

This approach to risk computation reflects the fact that most accidents are caused by driver error - dangerous situations are caused by drivers acting in unexpected ways. Risk computation models based on detecting unexpected behaviour can therefore be used to avoid many accidents, without having to use resource-intensive motion prediction models.

2.4.2 Occupancy prediction

Reachable set is a term used in occupancy prediction methods that denotes the set of all possible positions that a vehicle can be in during a certain time interval. A reachable set covers an area that a vehicle will not leave within the specified time frame. In occupancy prediction methods such as [8] and [7], the reachable sets for every vehicle in the traffic scene, for some future time horizon, is calculated. An overlap between two sets represents a possible future collision. Occupancy prediction can be used to formally prove that no collisions will occur within the near future. This feature makes the method suitable for trajectory planning since it enables trajectories to be classified as safe or possibly unsafe.

2.4.3 SVM based general collision prediction

Ward et. al [19] used time to collision (TTC) and looming data to train a Support Vector Machine (SVM) to predict future collisions between vehicles. TTC is a metric that is used to predict how much time is left until a collision occurs, if no object changes its trajectory. The planar TTC algorithm used by [19] provided accurate predictions when two vehicles were on a collision course. However, when the vehicles were not on a collision course, the TTC algorithm could not provide an indication that this was the case. Therefore they also used looming data, which represents how much space in ones field of vision an object occupies. An increase in looming rate would indicate that a collision is possible.

The method took into account uncertainties in vehicle measurements, and was also able to deal with communication failures. It was not dependant on road geometry and predefined speed models and trajectories. Thus, this method might be able to predict collisions in situations where other road-geometry based approaches fail. Examples of such situations could be in unmapped or dynamic traffic environments like small country roads or parking lots.

2.5 Cooperative intersection management

In this section, three different types of CIM methods, based on three different ways of intersection modelling, is described. Common assumptions made when designing CIM methods are also stated. The information in this section comes from a 2016 survey made by Chen and Englund [9].

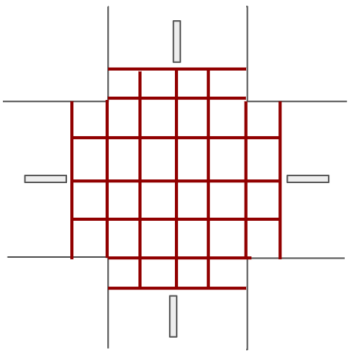


Figure 2.1: An intersection divided into cells.

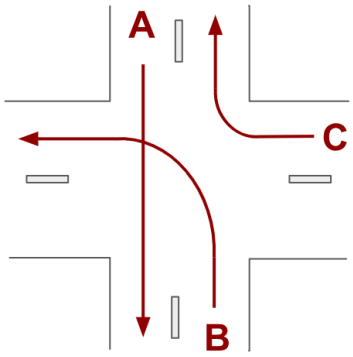


Figure 2.2: Predefined trajectories. A/B are conflicting, while A/C and B/C are not.

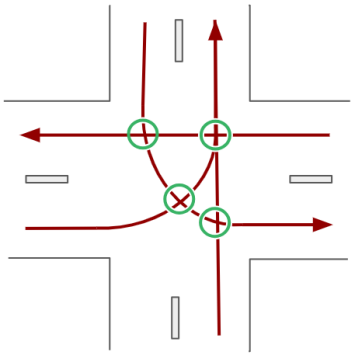


Figure 2.3: Collision regions derived from predefined trajectories.

2.5.1 Space-time based reservation methods

In these methods, the intersection is divided into different cells (see Figure 2.1), where each cell can be reserved by a vehicle for a discrete time interval. To be allowed to cross the intersection, vehicles need to reserve the space-time cells their planned routes pass through. The problem then becomes to schedule the reservations of all the vehicles such that each cell is only occupied by one vehicle at a time, thus guaranteeing that no collisions will occur. Other objectives, such as traffic flow and energy consumption can also be optimized. As both time and space is discretized, discrete optimization methods can be used to solve this problem. Typically, the scheduling calculation is done by a central control unit.

2.5.2 Trajectory planning methods

Here, vehicle trajectories for different turns and directions are predefined, as shown in Figure 2.2. Given two trajectories, it is easy to determine if they are conflicting, i.e. if they cross each other. Non-conflicting trajectories can be scheduled simultaneously while conflicting ones can not. The problem is to schedule trajectories in a non-conflicting way while optimizing other objectives such as traffic flow. To model the

problem in more detail, vehicle control parameters such as velocity and acceleration can also be considered. This problem can be solved using mathematical optimization methods, where the calculations are typically centralized.

2.5.3 Token based reservation methods

These methods combine space discretization and trajectory modelling, and only consider collision regions, as illustrated in Figure 2.3. Assuming that vehicles follow predefined trajectories, the collision regions can easily be identified. This way, only a select number of regions of the intersection needs to be considered. This modelling allows for a token based approach where each of the collision regions is associated with a token. A vehicle is only allowed to occupy a certain collision region if it has acquired the token for that region. When the vehicle has left the region, the token is returned. As a token can only be held by one vehicle at a time, collisions are avoided. These methods are usually decentralized, where vehicles negotiate with each other about who should get to acquire which tokens.

2.5.4 Common assumptions in CIM methods

- In trajectory planning and token based methods, vehicles are assumed to exactly follow pre-defined trajectories, whereas in space-time based methods, vehicle trajectories only need to pass through the cells reserved.
- In trajectory planning and space-time based methods, after a vehicle has been assigned a driving plan, it is assumed that the vehicle will always follow that plan.
- Communication between vehicles and infrastructure does not involve packet loss or latency.
- All vehicles are assumed to be automated and no vulnerable road users such as pedestrians are involved.

2.6 Vehicle cooperation in the presence of communication failures

In [20] and [21], Morales Ponce et al. addresses the problem of how cooperative vehicular systems using V2V communication can achieve a high performance without compromising safety despite communication failures. They present a default strategy that all participants can employ when other participants do not receive complete information on time. The solution, a timed distributed protocol, allows all participants

to either operate cooperatively or autonomously. The solution was tested on vehicular platooning and validated using the network simulator ns-3 [28].

Savic et al. [22] tackle the problem of intersection crossing with autonomous vehicles that cooperate via V2V communication. They propose a distributed algorithm that can handle communication failures while satisfying safety and liveness requirements. The algorithms works by letting the vehicles agree on the order in which they should cross the intersection. They found that crossing delays was only slightly increased, even in the presence of highly correlated communication failures.

Casimiro et al. [11] present a maneuver negotiation protocol where vehicles can ask for permission from other vehicles to perform potentially dangerous maneuvers. The vehicles have access to a membership service which is aware of the locations of all vehicles in the vicinity. V2V and V2X communication is used. The protocol is designed to handle communication failures.

3 Overview

In this chapter, we describe the problem we are trying to solve and our proposed solution. We also list the deviant vehicle behaviors we consider. Lastly, we define the scope of the project.

3.1 Problem description

In this project, we consider a single four-way give-way intersection as is illustrated in Figure 3.1. A number of autonomous vehicles are approaching the intersection from different directions and all the vehicles are connected and can communicate with each other. Each vehicle is able to estimate its own position, orientation and speed, and can share this information with the other vehicles. The problem is to guide each vehicle safely through the intersection while taking into account that vehicles might deviate from normal behavior in a number of ways. These include unpredictable travelling speeds, significant noise in state measurements, communication failures and failing to cooperate. There is no central control unit, so a decentralized solution is required.

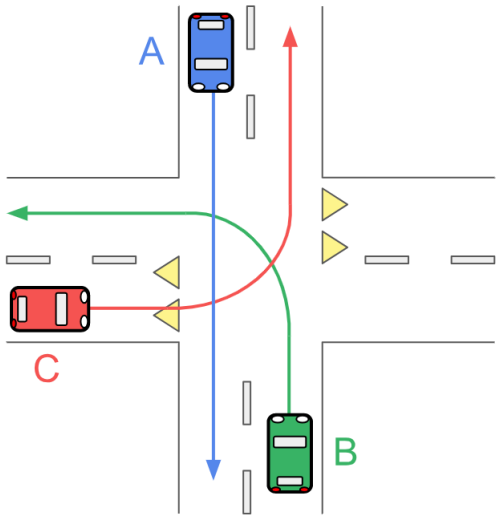


Figure 3.1: A four-way give-way intersection with three vehicles approaching the intersection. The north-south road is the primary road. In this example, vehicle A has higher priority than vehicle B, which has higher priority than vehicle C.

3.2 Proposed solution

We first consider that each vehicle uses a risk assessment software, which is able to detect dangerous situations and can trigger an emergency break in order to avoid collisions. At intersections, vehicle movement is highly dependant on other vehicles. We

therefore propose an interaction-aware risk assessment method where mutual influences between vehicles are considered. We also propose computing risk by detecting unexpected behaviors. This is motivated by the fact that most accidents at intersections are caused by driver errors such as running red lights or failing to yield the right of way.

We also consider using a CIM method to coordinate the vehicles. We propose a maneuver coordination software where vehicles need to ask for permission from other vehicles before they are allowed to perform a maneuver. When deciding whether to grant or deny a given maneuver request, the risk of that maneuver is calculated using the risk assessment method. If the risk is low enough, and certain other conditions are met, the maneuver is granted. Whenever a vehicle is granted, the priority rules of the intersection are temporarily changed in favor of the granted vehicle. This affects the risk assessment software, as it uses priorities to calculate what a vehicle is expected to do.

Unlike time based scheduling approaches, when a vehicle is granted, there is no time interval in which it needs to perform the maneuver. This way we don't need to rely on the assumption that individual vehicles follow time based driving plans.

If all the vehicles would follow the software, safety could be guaranteed, and emergency breaks would not be needed. We do however assume that some malicious drivers might fail to cooperate. In those cases, the emergency break triggered by the risk assessment software could still help avoid collisions. This way, we are not entirely reliant on vehicles following the software to ensure safety.

3.3 Deviation model

We assume vehicles can deviate from normal behavior in the following ways.

1. Selfish behavior.

   In a real setting, you would need to account for the fact that some vehicles might not follow traffic rules, might not use risk assessment, or might not cooperate like everyone else. Someone could for example hack their autonomous vehicle to behave maliciously. This behavior could also be due to some software or hardware failure, or due to some manual driver entering the traffic scene.

   Note that we still assume that all vehicles broadcast their estimated state, and that they will never (intentionally) send false information. We motive this with the fact that in a real setting you would be able to verify and complement state measurements with visual sensors such as cameras and lidar. Dealing with vehicles that intentionally send faulty maneuver coordination messages is outside the scope of this project.

2. Unpredictable speed when approaching and navigating through the intersection.

   In reality, vehicles do not follow fixed speed profiles. Even if they are autonomous, unforeseen things can happen, such as a vehicle needing to perform an emergency break. In the maneuver coordinatoin software, we *do not* make the assumption that vehicles need to enter and leave the intersection during a certain time interval. Therefore, we need to be able to handle cases where a vehicle is taking either a very long or a very short time to navigate the intersection.

3. Communication messages can be delayed or lost.

   In a real-world setting, we can not always guarantee perfect communication between vehicles. We certainly should not rely on it, as unforeseen failures could happen in any scenario. When taking into account that messages might be lost, a problem that arises is the need to be able to update the risk assessment software with only a few vehicles at a time. Also, in the maneuver coordination software, we need to wait before resending maneuver requests, as packet delays need to be taken into account.

   A vehicle can lose communication temporarily but we assume that, eventually, communication will always return. We make this assumption since, in the maneuver coordination software, a vehicle needs to notify the others when it has left the intersection. In a real scenario however, a vehicle can use other sensors, such as cameras and lidar to check if a vehicle have left the intersection. Instead of introducing more sensors, we can achieve the same effect by assuming that the communication will always return.

4. Significant noise in state measurements.

   The methods used to measure the position, orientation and speed of a vehicle are not perfect, and will always include some amount of noise. Taking into account uncertainty in the estimated state (caused by noise) is important, since we would need to be more careful if the uncertainty is high. We might for example require a bigger gap to the other cars when we are uncertain what our own state is.

See Section 7.2 for how these deviations are realized in the system evaluation.

3.4 Scope and limitations

The project does not involve any physical vehicles, and only simulations are used to evaluate the system. Also, no other road-users such as pedestrians are considered. No vehicle state filter was implemented, instead, we mimic a filter as described in Appendix A. Furthermore, speed profiles, predefined vehicle trajectories and acceptable gap thresholds are not based on any real data, but hand-constructed in a somewhat

realistic way. Assumptions for the maneuver coordination software can be seen in 6.2. Finally, due to time constraints, the evaluation of the system is limited.

4 System Description

In this chapter, we first describe how we model the intersection considered, including defining the priority rules. We then describe how the traffic simulation works, in terms of how the movements of the vehicles are simulated, how they communicate and what software we used to build the system.

4.1 Intersection model

We consider a four-way give-way intersection with single lanes and predefined vehicle trajectories. As there are four different travelling directions (north, south, west, east) and three different "turns" (left, straight, right), there are 12 different predefined trajectories.

Each trajectory has two speed profiles associated with it, one for stopping at the intersection, and one for going (entering the intersection). A few different speed profiles are shown in Figure 4.1.

Conflicting trajectories have different priorities, defined by some simple rules:

1. A trajectory originating from north or south always have higher priority than a trajectory originating from west or east.
2. If two trajectories, A and B, originate from opposing sides while A is a left turn and B is not, then B has higher priority than A.
3. If two trajectories originate from opposing sides and both are left turns, then they have the same priority.

These rules ensure that priorities are non-cyclic, which will be useful to avoid deadlocks in the maneuver coordination.

4.2 Vehicle state

The vehicle state considered consists of four variables, x and y position, orientation (θ) and speed (s). Each vehicle can measure its own state with some level of noise. We assume that each vehicle has some kind of filtering process that is able to output an *estimation* of the vehicle's true state, using a multivariate normal distribution. We also assume that all four state variables are independent. Thus, we model the estimated state as:

$S \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$,

where $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_\theta, \mu_s)^T$ and $\boldsymbol{\Sigma} = diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_s^2)$.

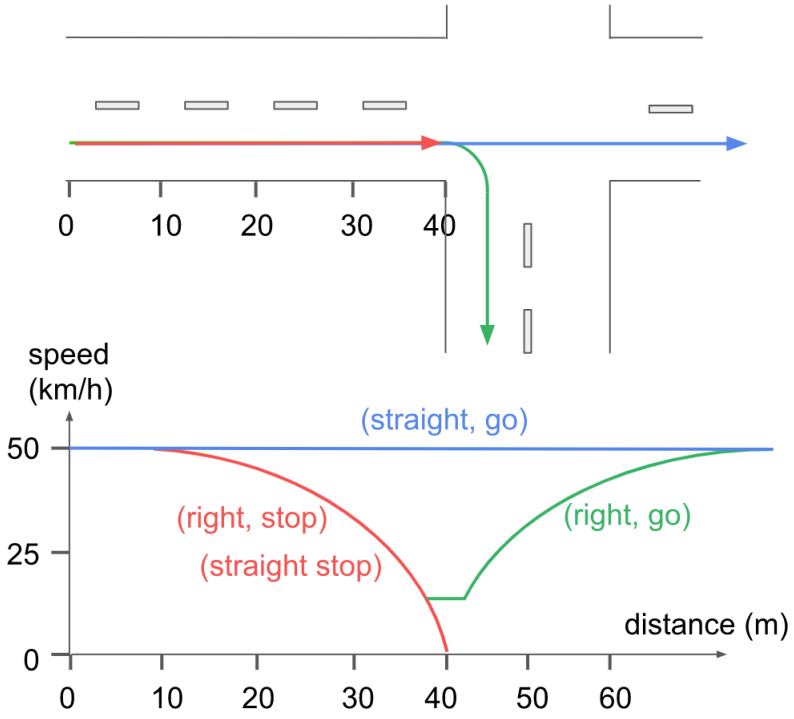


Figure 4.1: Trajectories and corresponding speed profiles for turning right and going straight. The axes on the two figures are aligned. Notice how the stop profiles have a speed of 0 km/h at the edge of the intersection.

A description of how we mimic a filter for the vehicle state is found in Appendix A.

4.3 Movement simulation

Each vehicle is initialized with a starting state, a trajectory and two speed profiles, one for going and one for stopping. Each iteration, a new vehicle state is evolved using a kinematic model. (See Figure 4.2).

The state evolution can be described with the follow equations.

$$\dot{x} = |v| \cdot cos(\theta)$$
$$\dot{y} = |v| \cdot sin(\theta)$$
$$\dot{\theta} = |v| \cdot \frac{tan(\Phi)}{L}$$

A new steering angle and a new speed is needed to evolve the current vehicle state. A simple PID controller is used to output the new steering angle needed to follow the

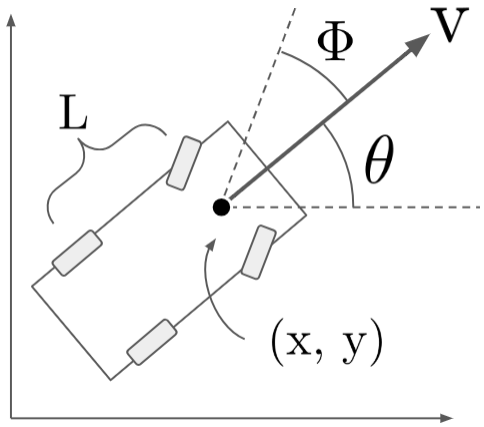


Figure 4.2: The kinematic model. Φ is the steering angle, θ is the orientation relative to the x-axis, and v is the velocity.

predefined trajectory, and the new speed is determined by the speed profile currently used.

The vehicle can switch back and forth between the go and stop profiles depending on the result of the maneuver coordination. The speed can also be influenced by the triggering of an emergency break.

4.4 Software frameworks

The system is designed using the Python version of ROS (Robot Operating System) [24]. Using ROS, different Python processes (nodes) can pass messages to each other on different communication channels. Each node has a main thread that runs continuously, and whenever a message is received, a callback function is invoked that allows the node to process the received data. We also use the ROS package *rviz* for visualizing the simulation.

4.5 Communication

The communication architecture can be seen in Figure 4.3. We consider a simple architecture where each vehicle is controlled by a separate node. Every pair of nodes are connected with two communication channels, one for broadcasting the estimated state and one for sending maneuver coordination messages. The vehicle nodes are also connected to a visualization node which allows the vehicles and the intersection to be drawn to the screen by rviz.

There are three types of messages sent between vehicles:

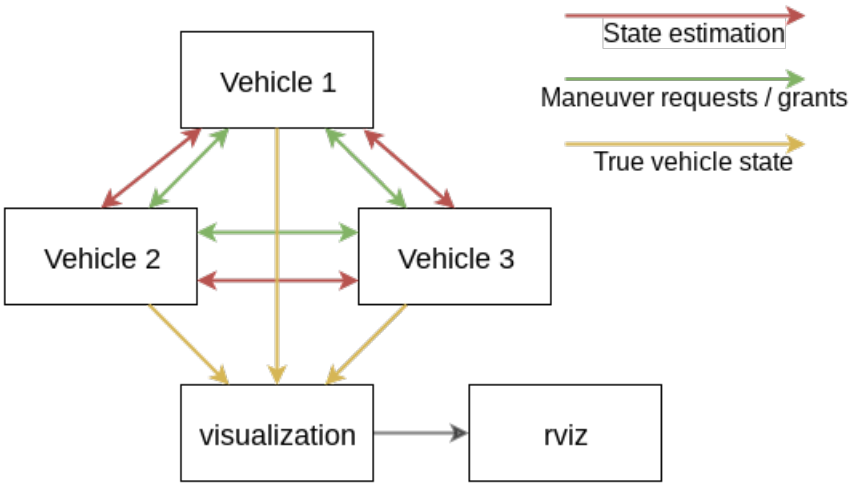


Figure 4.3: The communication architecture in a scenario with three vehicles.

- State estimation: $(\mu_x,\ \mu_y,\ \mu_\theta,\ \mu_s,\ \sigma_x,\ \sigma_y,\ \sigma_\theta,\ \sigma_s,\ id,\ timestamp)$
- Maneuver request: $(type,\ time\_of\_initial\_request,\ id,\ timestamp)$
- Maneuver grant: $(id,\ timestamp)$

The *type* variable represents the requested maneuver and can either be `"left"`, `"straight"` or `"right"`. The time of the initial maneuver request is included so that deadlocks can be avoided in certain edge cases, something that is explained in Chapter 6. We assume that both vehicles have access to a global clock. State messages are sent at a rate of 60 times per second.

5 Risk Assessment Software

This chapter presents the risk assessment software developed for this project. The software design is influenced by the work of Lefèvre et al. [5], which is described in Sections 2.3.2 and 2.4.1. Like them, we use intention and expectation to calculate risk, and time gaps to calculate expectation. However, our intention calculation is quite different, and instead of using particle filters, we model each variable with probability distributions.

5.1 Overview

In each iteration (time step), the software relies on four main processing steps to assess the risk of a given situation. These are:

1. Receiving the estimated state (S) from each vehicle.
2. Inferring the intention (I) of each vehicle.
3. Calculating the expectation (E) of each vehicle.
4. Calculating the risk (R) for each vehicle.

Each step results, for each vehicle, in a new value for one of the four software variables S, I, E and R. The dependencies between these variables are illustrated in Figure 5.1.

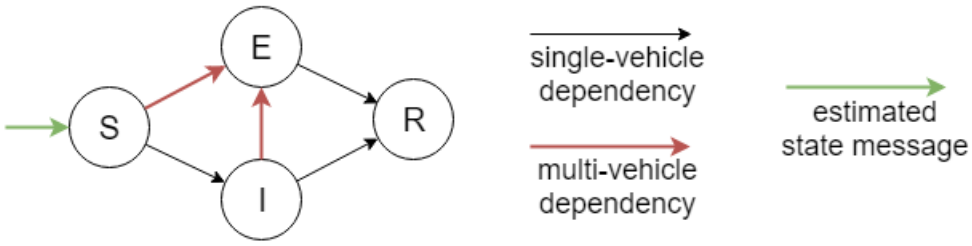


Figure 5.1: Dependencies between the estimated state (S), intention (I), expectation (E), and risk (R) variables. Note that the expectation of a vehicle is derived from all vehicle's estimated states and intentions, while the intention and risk is derived only from that vehicle's own variables.

The following sections describe steps 2-4 in detail.

5.2 Intention inference

Vehicle intention (I) is comprised of a combination of two variables:

- Intention to stop or go at the intersection, $I_s \in \{go, stop\}$
- Intended trajectory, $I_t \in \{left, straight, right\}$

We model the intention as a discrete random variable with a probability for each of the six possible (I_s, I_t) pairs. To calculate these probabilities, we first assign a *likelihood* (L) to each pair, and then divide by a normalizing constant. The likelihood for each pair is a product of the *state likelihood* (L_s) and the *priority road likelihood* (L_{pr}). The state and priority road likelihoods are explained in the following subsections.

5.2.1 State likelihood

The state likelihood L_s for some (I_s, I_t) pair is calculated using:

- The state estimation $S \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ for the vehicle in question. As mentioned in Section 4.2, $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_\theta, \mu_s)^T$ and $\boldsymbol{\Sigma} = diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_s^2)$.
- The trajectory and speed profile corresponding to the (I_s, I_t) pair.

Optimal state

First, μ_x, μ_y, the trajectory and the speed profile are used to derive an *optimal state* $\mathbf{o} = (o_x, o_y, o_\theta, o_s)^T$. This is done in four steps:

1. (o_x, o_y) is the point on the trajectory which is closest to (μ_x, μ_y).
2. o_θ is the angle of the tangent of the trajectory at (o_x, o_y).
3. d_o is the distance travelled (i.e. the length of the trajectory) at (o_x, o_y). The starting point for each trajectory is 117.5 meters from the intersection.
4. o_s is the value of the speed profile at d_o.

An illustration of this can be seen in Figure 5.2.

Error function

Once we have derived the optimal state, we define a function *err* as such:

$err(\boldsymbol{\mu}) := \mathbf{w}(\mathbf{o} - \boldsymbol{\mu})^2$ where $\boldsymbol{w} = (w_x, w_y, w_\theta, w_s)$

The function takes some vehicle state $\boldsymbol{\mu}$ and outputs the *error*, or distance, from that state to the optimal state $\boldsymbol{o}$. The impact that each state variable has on the error is controlled by the weight vector $\boldsymbol{w}$, which was set to $(125, 125, 125, 1)$.

Expected error

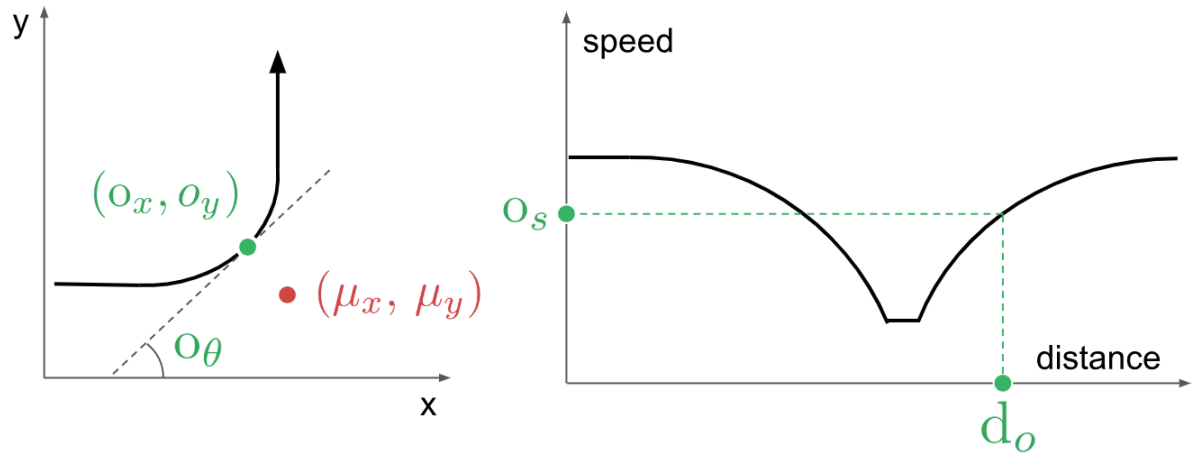


Figure 5.2: Deriving the optimal state $(o_x, o_y, o_\theta, o_s)$ from μ_x, μ_y, the trajectory for $(go, left)$ and the speed profile for $(go, left)$

The idea is to base the state likelihood L_s on the error between the estimated state S and the optimal state $\boldsymbol{o}$. Since S is a random variable, we can not calculate this error directly. Instead, we calculate the *expected* error.

If *pdf* is the probability density function for S, the expected error becomes:

$$\begin{aligned} E[err(S)] &= \int_{-\infty}^{\infty} err(\mathbf{x}) pdf(\mathbf{x}) d\mathbf{x} \\ &= \boldsymbol{w}((\boldsymbol{\mu} - \boldsymbol{o})^2 + \boldsymbol{\sigma}^2) \end{aligned}$$

where $\boldsymbol{\sigma}^2 = (\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_s^2)$.

The integral was solved with the computation software Mathematica [29].

One issue with this approach is that a vehicle driving ususually fast, before the go and stop speed profiles start to diverge, would be seen as equally likely to go or stop at the intersection. To compensate for this, we add another condition to the expected error calculation: If $\mu_s - o_s > 10$ km/h and the (I_s, I_c) pair in question is a "stop" pair, then the expected error is set to ∞.

State likelihood calculation

Finally, we define the state likelihood as $L_s = 1/E[err(S)]$. This means a large error will result in a small likelihood and vice versa.

Notice how the expected error increases when $\boldsymbol{\sigma}^2$ increases. This means that the higher the variance is, the more *even* the intention probability distribution will become. This

makes sense, since more uncertainty in the vehicle state estimations should result in a less precise inference.

5.2.2 Priority road likelihood

We assume that vehicles on the priority road are more likely to continue straight than to turn left or right. Given some (I_s, I_t) pair, the priority road likelihood for some vehicle v is defined as:

$$L_{pr} = \begin{cases} r & \text{if } I_t = straight \text{ and } v\text{'s origin is north or south.} \\ 1 & \text{otherwise} \end{cases}$$

where r is some positive constant greater than 1. The value of r was set to 9. This will bias the intention probability towards continuing straight for the vehicles on the priority road.

5.3 Expectation calculation

This section describes how the *expectation* for each vehicle is calculated.

5.3.1 Model

Given some intended trajectory I_t, the expectation E of a vehicle can, just like the I_s variable, either be *stop* or *go*. For each intended trajectory I_t, we model the expectation as a probability $P(E = go \mid I_t)$. $P(E = stop \mid I_t)$ can then easily be calculated as $1 - P(E = go \mid I_t)$.

Thus, for each vehicle, we need to calculate three values, $P(E = go \mid left)$, $P(E = go \mid straight)$ and $P(E = go \mid right)$.

5.3.2 Equation for calculating expectation distribution

The expectation to go for some vehicle A given some intended trajectory I_t^A is calculated using the following equation.

$$
\begin{aligned}
&P(E_s^A = go \mid I_t^A) = \\
&\quad \min_{B \in Q}(\\
&\qquad P(E_s^A = go \mid I_t^A, I_t^B = left) \times P(I_t^B = left) + \\
&\qquad P(E_s^A = go \mid I_t^A, I_t^B = straight) \times P(I_t^B = straight) + \\
&\qquad P(E_s^A = go \mid I_t^A, I_t^B = right) \times P(I_t^B = right) \\
&\quad)
\end{aligned}
$$

where Q is the set of all other vehicles in the scene. If Q is empty then the expectation to go is set to 1. Note that the expectation is calculated with respect to one vehicle at a time. The minimum value is then chosen as the final expectation. The intention calculations are described in Section 5.2.

The expectation to go for A given two intended trajectories I_t^A and I_t^B is given by

$$
P(E_s^A = go \mid I_t^A, I_t^B) = \begin{cases} 1 & \text{if } B \text{ has granted } A \\ 1 & \text{if } I_t^A \text{ and } I_t^B \text{ do not intersect} \\ 1 & \text{if } I_t^A \text{ has higher priority than } I_t^B \\ P(G < -g_1) + P(G > g_2) & \text{otherwise} \end{cases}
$$

where G is the *gap* between A and B, and g_1 and g_2 are positive constants. g_1 was set to 1.0 and g_2 was set to 1.5. Ideally though, these values should not be constant, but vary depending on the combination of conflicting trajectories.

Thus, A is expected to go if B has granted it, if there are no conflicts between A and B, or if A has the priority over B. Examples of the last two cases can be seen in Figures 5.3 and 5.4. Note also that A is expected to go if the absolute gap is large enough to provide a safe margin.

The gap is defined as $G = T_p^B - T_p^A$, which represents the difference in estimated arrival times for A and B to the point p, the point where I_t^A and I_t^B intersect. The arrival times, and thus the gap, is modelled using normal distributions. An example of a gap calculation is illustrated in Figures 5.5 and 5.6. The arrival time estimation is described in Appendix B.

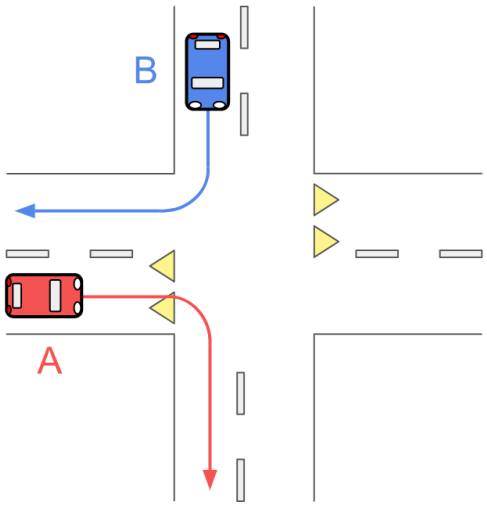


Figure 5.3: A and B's intended trajectories do not intersect.

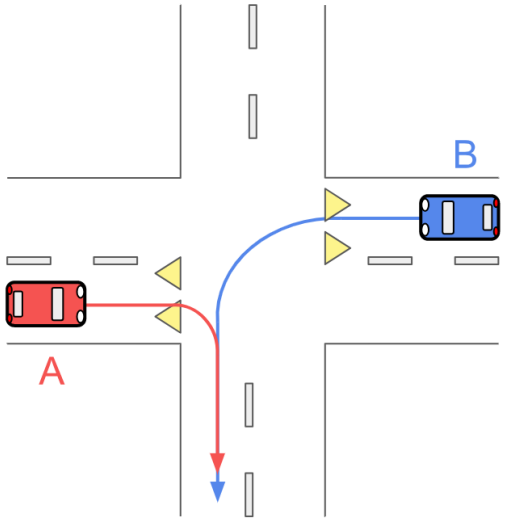


Figure 5.4: A's intended trajectory has higher priority than B's.

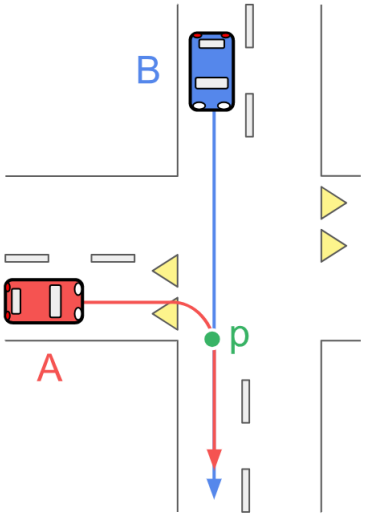


Figure 5.5: B's intended trajectory has higher priority than A's. The trajectories intersect at point p.

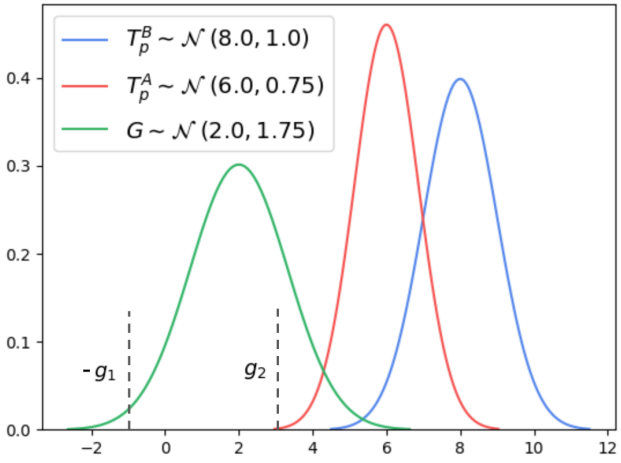


Figure 5.6: Estimated arrival times to p, and the resulting gap. Typical values for $-g_1$ and g_2 are also shown.

5.4 Risk calculation

A vehicle is considered risky if it is expected to stop but intends to go. The risk for some vehicle v is calculated as follows.

$$
\begin{aligned}
R_v &= P(E_s^v = stop \wedge I_s^v = go) \\
&= P(E_s^v = stop \wedge I_s^v = go \mid I_t^v = left) + \\
&\quad P(E_s^v = stop \wedge I_s^v = go \mid I_t^v = straight) + \\
&\quad P(E_s^v = stop \wedge I_s^v = go \mid I_t^v = right) \\
&= P(E_s^v = stop \mid I_t^v = left) \times P(I_s^v = go \mid I_t^v = left) + \\
&\quad P(E_s^v = stop \mid I_t^v = left) \times P(I_s^v = go \mid I_t^v = straight) + \\
&\quad P(E_s^v = stop \mid I_t^v = left) \times P(I_s^v = go \mid I_t^v = right)
\end{aligned}
$$

All these terms are calculated as described in Sections 5.2 and 5.3. If the risk exceeds some threshold ϵ, an emergency break is triggered. The value of ϵ was set to 0.55. The emergency break causes the vehicle to slow down with a constant acceleration a. The value of a was set to -15 m/s^2.

6 Maneuver Coordination Software

This chapter presents the decentralized coordination software used to ensure that vehicles can safely navigate through the intersection.

6.1 High-level description of the software

High-level pseudocode for the software is presented below. The main parts of the code is explained in following subsections. For a more complete pseudo-code, see Appendix C.

```
1  granted <- False
2  grant_list <- Empty list
3  max_transmission_delay <- Some positive constant
4  main_loop()

5  function tryManeuver():
6      ask_list <- List of vehicles with higher priority than ego
                   vehicle (with respect to its intended maneuver),
                   that have not yet left the intersection

7      if ask_list is empty:
8          granted <- True
9          return

10     grant_inbox <- Empty list
11     Send a REQUEST message to all vehicles in ask_list
12     Start a timer, which in 2 * max_transmission_delay seconds will
       invoke tryManeuver

13 upon received REQUEST message:
14     if message is older than max_transmission_delay:
15         return

16     if grant_list contains the sender id:
17         Remove the sender id from grant_list

18     c <- Ego vehicle's intended maneuver do not conflict with the
           requested maneuver
19     if c:
20         Reply with a GRANT message
21     else:
22         l <- Both the requested maneuver and the ego vehicle's
```

```
                intended maneuver are left turns and ask_list contains
                sender id
23          a <- The sender arrived to the request line earlier than the
                ego vehicle

24          h <- Ego vehicle has time to stop before the intersection
25           g <- The gap to perform the requested maneuver, with respect
                to the ego vehicle, is large enough
26          if l and a or h and g:
27              Reply with a GRANT message
28              Add the sender id to grant_list

29  upon received GRANT message:
30      if message is older than max_transmission_delay:
31          return
32      Add the sender id to grant_inbox
33      if ask_list and grant_inbox contain the same ids:
34          Cancel timer
35          granted <- True

36  upon received STATE message:
37      if the sender has left the intersection:
38          if grant_list contains the sender id:
39              Remove the sender id from grant_list

40  function main_loop():
41      if ego vehicle has passed request line:
42          tryManeuver()
43      if granted and grant_list is empty:
44          Use the go speed profile
45      else:
46          Use the stop speed profile
47      Evolve vehicle state
48      Send a STATE message to the other vehicles
49      main_loop()
```

6.1.1 Deciding to go or stop

A vehicle's decision to go or stop at the intersection is determined by a boolean variable **granted** and a list **grant_list** (lines 1-2). **granted** needs to be true and **grant_list** needs to be empty for the vehicle to be allowed to enter the intersection (lines 43-46).

granted is set to true if there is no one to ask permission from (lines 7-8), or if all the asked vehicles have granted the ego vehicle (lines 33, 35). If the ego vehicle grants some other vehicle, then that vehicle is added to **grant_list** (unless their trajectories

are non-conflicting) (lines 27-28). Vehicles are removed from **`grant_list`** when the ego vehicle gets information that they have left the intersection (lines 36-39).

6.1.2 Requesting permission

Whenever a vehicle crosses a request line (see Figure 6.2), it starts the request process (lines 41-42). The first step is to determine which vehicles to ask for permission (line 6). Those are vehicles that have higher priority than the asking vehicle and have not yet left the intersection. Some examples of this are seen in Figure 6.1. If there is no one to ask permission from, the vehicle is automatically granted (lines 7-8). Otherwise, maneuver requests are sent out to the other vehicles (line 11), and a timer is started (line 12). Whenever the timer runs out, the request process is repeated. If a vehicle has been granted by everyone it asked, the timer is cancelled and **`granted`** is set to true (lines 33-35).

Request and grant messages that take longer than **`max_transmission_delay`** seconds to arrive are ignored (lines 14-15, 32-33). Since we wait **`2 * max_transmission_delay`** seconds to send new requests (line 12), there is no risk that we confuse different requests with each other. We assume that all vehicles have access to a global clock.

6.1.3 Granting permission

There are three cases when a vehicle (A) can decide to grant permission to another vehicle (B).

1. A's intended trajectory does not conflict with B's (lines 18-20). (See Figure 6.3)
2. The gap (see Section B) from B to A is large enough, and A has time to stop at the intersection (lines 24-27). (See Figure 6.2) A gap is considered large enough if $P(G > g) > \epsilon$ where G is the estimated gap, g is a positive constant and c is a value between 0 and 1. g was set to 2.5 seconds and ϵ was set to 0.8.
3. A and B have the same priority, and B arrived at the request line earlier than A (lines 22-23, 26-27). This only happens when both vehicles intend to turn left and are approaching the intersection from opposite sides. (See Figure 6.4)

6.2 Correctness

The software ensures safety and is free from deadlocks and starvation under the following assumptions.

1. All vehicles know the origin (north, south, etc.) of all the other vehicles in the traffic scene.

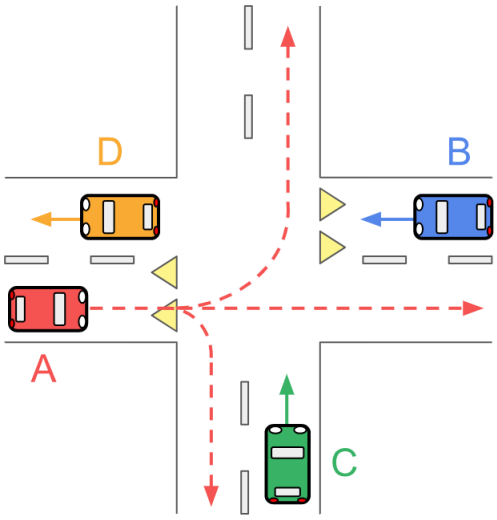


Figure 6.1: Deciding who A should ask permission from. If A intends to turn left it needs to ask B and C. C might be turning right, but A does not know this. If going straight ahead, A only needs to ask C. If turning right, A does not need to ask anyone, and is automatically granted. A never needs to ask D, since D has already left the intersection.

2. The number of vehicles are fixed, i.e. no new vehicles will arrive.
3. Communication range is infinite.
4. Communication can only be lost temporarily, i.e. it will eventually return.

6.2.1 Safety

Safety (no risk for collision) can be guaranteed if no two vehicles with conflicting trajectories will be inside the intersection simultaneously. Let us first demonstrate that this is the case for traffic scenarios with only two vehicles, A and B.

Assume that A has higher or equal priority than B. This means that B will have to ask A for permission before entering the intersection. Because of assumption 1, B is aware of this. B will not enter the intersection unless it is granted, or if A leaves the intersection. If A grants B, A will add B to its grant list. A will not enter the intersection until its grant list is empty, which will only happen once B has crossed and left the intersection. A will not grant B if A does not have time to stop.

Adding more vehicles does not affect safety, since a vehicle needs to be uniformly granted by *all* the asked vehicles to be allowed to enter the intersection.

Delayed or lost messages do not affect safety. Let us look at each message type and see what the effects of delayed or lost messages are. As mentioned, messages delayed

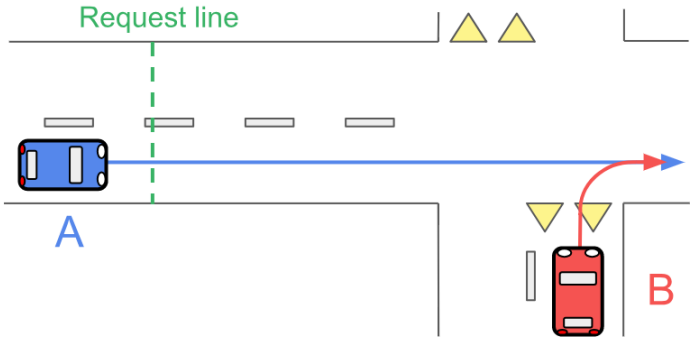


Figure 6.2: The estimated gap from B to A is large, and A has time to stop, so A will grant B.

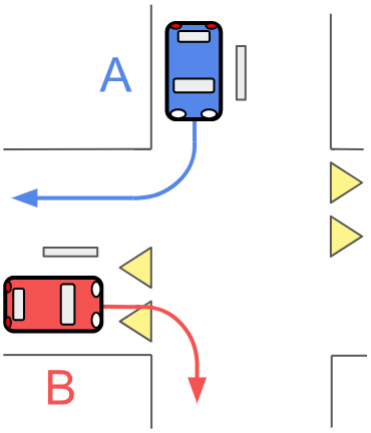


Figure 6.3: A and B's intended trajectories do not intersect, so A will grant B.

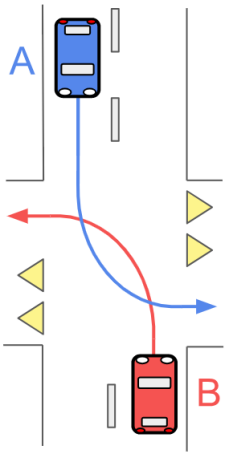


Figure 6.4: Both A and B intends to turn left, and are asking each other for permission. A will grant B, since B arrived earlier at the request line (not seen in the figure).

more than `max_transmission_delay` seconds are ignored, and can thus be considered lost.

State messages

Because of assumptions 1 and 2, `ask_list` will only shrink with time. If a state message is lost or delayed, a vehicle that has already left the intersection might still being asked for permission. This might lower throughput, but will not affect safety.

Request messages

If a request message is delayed, the gap estimation needs to be done with "old" information, and the result might be inaccurate. However, safety does not depend on the accuracy of the gap estimation. In fact, the gap condition (line 25) is only used for throughput purposes. Even if the estimated gap is completely inaccurate, safety will be ensured by the condition that the granting vehicle has time to stop before the intersection (line 24).

If a request message is lost, the vehicle will not be granted, the timer will run out, and it will simply try again.

Grant messages

Once a vehicle A sends a grant message to another vehicle B, A will immediately regulate its speed according to the stop speed profile. This means message delay has no affect on safety. If the granting message is lost, B does not know it was granted, however A will still stop. B's timer will run out and new requests will be sent out. A will then remove B from its grant list (lines 16-17), and redo the grant process. This time, the gap estimation may be different. However, none of this affects safety.

6.2.2 Deadlock

A deadlock can occur if there is a circular relationship between vehicles and who they need to ask permission from. The only such relationship that exists in the considered intersection context arise in the special case where two vehicles arrive from opposing sides, and both intend to turn left. In this case, both vehicles need to ask each other for permission.

A deadlock would occur if both of these vehicles grant each other, or if no one grants the other one. Whoever is granted depends on each vehicle's arrival time to the request line. This information is included in the request message, so both vehicles make their decisions using the same information. If the vehicles have exactly the same arrival times, the vehicle ids are used to break the tie. This means that exactly one of the vehicles is guaranteed to be granted.

If one of the vehicles have not yet arrived at the request line, the asked vehicle will be granted. If communication temporarily fails, none of the vehicles will enter the intersection. However, as stated in assumption 4, communication will eventually return.

6.2.3 Starvation

Starvation can occur if some vehicle is never allowed to enter the intersection. This would mean that **`granted`** is never set to true or **`grant_list`** is never emptied.

As mentioned, **ask_list** will only shrink with time. Even if a vehicle is never granted, the vehicles with higher priority will eventually leave the intersection. The edge case with two left turning vehicles was covered in Section 6.2.2. Because of assumptions 3 and 4, the vehicle will eventually notice that it has been granted, or that the other vehicles have left. Either way, **granted** will be set to true.

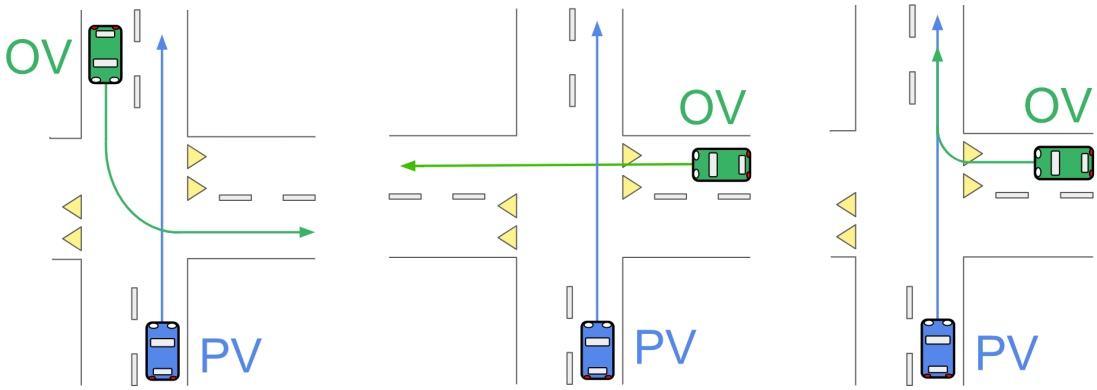
There are two ways in which **grant_list** could keep from emptying. Either a vehicle in continuously re-granted and the messages keep getting lost, or a granted vehicle loses communication and the state messages signalling that it has left the intersection keep getting lost. However, because of assumptions 3 and 4, **grant_list** will eventually be emptied.

7 Evaluation Strategy

This section describes how our solution described in Chapters 4-6 was evaluated. The results are presented in Chapter 8.

7.1 Test scenarios

We consider the three test scenarios illustrated below. These scenarios are some of the most frequent and severe collision scenarios at road traffic intersections in Europe [30]. In all scenarios, there is a Priority Vehicle (PV), which drives on the priority road, and an Other Vehicle (OV), which is performing a potentially dangerous maneuver.



Test scenario 1:
Left turn across path

Test scenario 2:
Crossing path

Test scenario 3:
Merging right

To create different instances of these scenarios, we vary the starting position of the OV while keeping the starting position of the PV constant. We divide these instances into three categories based on what happens if both vehicles follow the go speed profiles and do not stop at the intersection:

- **Collision cases**, where the vehicles collide.
- **Semi-dangerous cases**, where the vehicles do not collide but the gap between the OV and the PV is between -1.5 and 2 seconds. These gap limits were chosen by observing simulations with different gaps and intuitively labelling these dangerous or non-dangerous.
- **Non-dangerous cases**, where the gap is less than -1.5 or greater than 2 seconds.

We consider 30 different instances of each test scenario, 10 of which are collision cases, 10 are semi-dangerous and 10 are non-dangerous.

7.2 Deviations

In each scenario instance, the vehicles can vary their behavior in nine different ways:

1. **Normal behavior. (`normal`)**
   Both vehicles follow their speed profiles without deviating.
2. **OV ignores instructions and acts selfishly. (`OV_selfish`)**
   The OV does not trigger any emergency breaks, does not send any maneuver coordination messages and does not stop to give way at the intersection.
3. **Both vehicles drive fast. (`both_fast`)**
   Starting from 30 meter before the intersection, both vehicles increase their speeds with 15 km/h, i.e. they will start driving 15 km/h faster than the normal speed profile.
4. **PV drives slow, OV drives fast. (`PVslow_OVfast`)**
   Starting from 30 meters before the intersection, The PV decreases its speed with 10 km/h, while the OV increases its speed with 15 km/h. The PV will not drive slower than 12 km/h.
5. **OV drives slow, PV drives fast. (`PVfast_OVslow`)**
   Starting from 30 meters before the intersection, The OV decreases its speed with 10 km/h, while the PV increases its speed with 15 km/h. The OV will not drive slower than 12 km/h.
6. **Communication lost 40 meters before intersection. (`com_loss_40`)**
   All communication between the vehicles are lost when the OV is 40 meters from the intersection. Communication is returned when either vehicle has travelled 30 meters after leaving the intersection.
7. **Communication lost 20 meters before intersection. (`com_loss_20`)**
   All communication between the vehicles are lost when the OV is 20 meters from the intersection. Communication is returned when either vehicle has travelled 30 meters after leaving the intersection.
8. **Communication lost inside intersection. (`com_loss_inside`)**
   All communication between the vehicles are lost when the OV has travelled 5 meters into the intersection. Communication is returned when either vehicle has travelled 30 meters after leaving the intersection.
9. **Significant noise. (`noise`)**
   The noise in the in state measurements is determined by a noise vector $\boldsymbol{Z} = (z_x, z_y, z_\theta, z_s)^T$, as described in Appendix A. The vehicle state consists of x and y positions, orientation (θ) and speed (s). Normally, $\boldsymbol{Z}$ is set to $(0.2, 0.2, 0.04, 0.1)^T$, but it is here set to $(1.0, 1.0, 0.2, 0.5)^T$ (for both vehicles).

Motivations for the deviations can be seen in Section 3.3.

7.3 Evaluating the risk assessment software

To evaluate the risk assessment software, we measure how early it can detect dangerous situations and how many times it classified a semi-dangerous or non-dangerous situation as risky. Here, the vehicles will follow the go speed profiles without breaking.

In the **collision cases** (see Section 7.1), we measure the time when the risk assessment software classifies the situation as dangerous, $T_{detection}$, and the time when the collision takes place, $T_{collision}$. We calculate Time-To-Collision (TTC) as $TTC = T_{collision} - T_{detection}$.

In the other cases we measure the rate of false alarms, i.e. the percentage of cases "wrongly" classified as dangerous. (Some of the semi-dangerous cases are quite dangerous and applying an emergency breaks in these cases would be appropriate.)

With 30 instances of each of the three test scenarios, and nine different deviations, we get 810 combinations. We ran each test case combination three times with different random seeds. Thus, for this evaluation, we ran 2430 tests.

7.4 Evaluating the combination of risk assessment and maneuver coordination

To assess which advantages can be gained by using risk assessment and maneuver coordination together, compared to using each component separately, we consider the following three different ways of controlling both vehicles:

- **(RA)** Using the risk assessment software to trigger emergency breaks.
- **(MC)** Using the maneuver coordination software to determine whether to stop or go at the intersection.
- **(RA+MC)** Using both risk assessment and maneuver coordination together.

We call each of these a *control mode*. We use the following criteria to compare the different control modes:

- Number of collisions.
- Severity of each collision, measured by the square of the relative speed of the colliding vehicles.
- Number of times emergency breaks were triggered. (EBs)
- Travelling time (TT). For each vehicle we measure the time elapsed from the start of the simulation until it reached its destination. We then add these individual times together.

- Priority violations (PVs), i.e. situations where vehicles on the priority road were forced to slow down. We measure both the number of priority violations and the total time lost to vehicles on the priority road due to slowing down.
- Number of times the OV was granted.

With 30 instances of each of the three test scenarios, nine different deviations, and three control modes, we get 2430 combinations. We ran each test case combination three times with different random seeds. Thus, for this evaluation, we ran 7290 tests.

8 Results

In this section we present the results from our evaluation of the system, as described in Chapter 7. We found that by combining the risk assessment software with the maneuver coordination software, collisions were either completely avoided or greatly mitigated, even while taking into account deviant vehicle behaviors such as selfish behavior, unpredictable travelling speeds, communication failures and significant noise in state measurements. However, the evaluation was limited, and further testing would be needed to make any strong claims about the safety of the system.

8.1 Evaluation of the risk assessment software

Here, we present the results of the evaluation of the risk assessment software. We first go through which results we expected and then we present the actual results. Lastly, we point out the differences between the expected and actual results.

8.1.1 Expected results

Detection of dangerous situations

For the deviations with intact communication (all deviations except **`com_loss_40`**, **`com_loss_20`** and **`com_loss_inside`**), we expect to detect all collisions before they occur, as any situation where the vehicles are just about to collide should always be classified as dangerous. For the deviations with lost communication (**`com_loss_40`**, **`com_loss_20`** and **`com_loss_inside`**), we expect that the OV (Other Vehicle) can use the old arrival time estimation of the PV (Priority Vehicle) to infer that performing the maneuver would be dangerous. However, this would require the arrival time estimation to be somewhat accurate, which is unclear.

False alarms

We expect no false alarms in non-dangerous situations, unless the deviations are speed related (**`both_fast`**, **`PVslow_OVfast`** and **`PVfast_OVslow`**). With these deviations it is unclear if non-dangerous situations would cause false alarms, since the arrival time estimation (see Appendix B) is based on vehicles following the standard speed profile. For the same reasons, we expect false alarms in semi-dangerous situations to be more common with these deviations.

Comparing TTC for different deviations within each scenario

Before predicting the TTC values of the different tests, it is important to note that the speed profiles for turning left and right slow down before the intersection, while the speed profile for going straight use constant speed. (See Figure 4.1)

In scenario 1 (left turn across path), when the OV starts slowing down, we do not know whether it will stop at the intersection, turn right or turn left. Once the stop and go speed profiles for the right and left turns start to diverge, we can infer that the OV will not stop. However, we still do not know if it will turn right or left. At this point, the risk is about 0.5 in the collision scenarios, but we do not want trigger the emergency break yet. This is why we set the risk threshold to 0.55. Once the OV starts to turn, we can infer the left turn and then trigger the emergency break (although in this test we do not actually slow down the vehicles). This applies to all deviations in scenario 1. The difference in TTC should then mainly depend on the speed of the OV between the detection point (the edge of the intersection) and the collision point. This implies that **`both_fast`** and **`PVslow_OVfast`** would have a low TTC, that **`PVfast_OVslow`** would have a high TTC, and that the other deviations would be in between.

In scenario 2 (crossing path), the OV drives with constant speed without slowing down. We can therefore detect quite early that the vehicle will not stop. Unlike in scenario 1 (left turn across path), all the possible maneuvers that the OV can make would be dangerous. We therefore do not have to wait to see if the OV intends to turn right, turn left or go straight before triggering the emergency break. We would expect the lowest TTC in **`PVfast_OVslow`** since with this deviation, the OV is slowing down and it becomes more difficult to infer if it will stop or go. On the other hand, we would expect the highest TTC with **`both_fast`** and **`PVslow_OVfast`**, where the OV drives extra fast. There, the software can infer quite early that the OV will not stop because of the extra conditional added in the state likelihood calculation. (See Section 5.2.1).

In scenario 3 (merging right), we can only tell if the vehicle will stop or go once the speed profiles for the right and left turns start to diverge, which happens only a short distance outside of the intersection. We also expect **`both_fast`** and **`PVslow_OVfast`** to have a longer TTC than the other deviations for the same reasons as in scenario 2.

Comparing average TTC for different deviations

In all test scenarios, we expect a slightly lower TTC for **`noise`**, as it will take a bit longer time to infer the correct intention of the OV. For **`com_loss_40`** and **`com_loss_20`**, we expect that only the OV detects the dangerous situation. The PV would remain neutral to whether or not the OV will do something unexpected and therefore not detect the risky situation (the risk will not go higher than 0.5). However, we expect that in these deviations, TTC (detected only by the OV) will be similar to TTC in **`normal`**, since TTC here depends on when the OV can infer its own dangerous intended maneuver. An old arrival time estimation can be used, so there should not be any difference in TTC between **`normal`**, **`com_loss_40`**, **`com_loss_20`** and **`com_loss_inside`**. In

com_loss_inside, both vehicles should detect the dangerous situation, since by the time communication is lost, the OV has already violated the priority rules.

Comparing average TTC for different scenarios

The TTC values depend on the speed of the vehicles and how long the distances between the detection points and the collision points are. In scenario 1 (left turn across path), The OV slows down and the distance is quite long. In scenario 2 (crossing path), the OV does not slow down, and the distance is quite long. In scenario 3 (merging right), the OV slows down and the distance is very short. We therefore expect scenario 1 to have the highest TTC. It is unclear if scenario 2 or 3 will have the lowest TTC.

8.1.2 Observed results

Figure 8.1 shows the percentage of detected collisions as a function of the time remaining before the collision, for the three test scenarios with the different deviations. (**OV_selfish** is left out since it would be identical to **normal**.)

Detection of dangerous situations and false alarms

No collisions were missed, and there were no false alarms in any of the non-dangerous scenarios. 29% of the semi-dangerous cases were classified as risky. False alarms in semi-dangerous cases were most common with **both_fast**, **PVslow_OVfast** and **PVfast_OVslow**.

Comparing TTC for different deviations within each scenario

In scenario 1 (left turn across path), **both_fast** and **PVslow_OVfast** had the lowest TTC and **PVfast_OVslow** had the highest TTC. In scenario 2 (crossing path), the opposite was true, i.e. **PVfast_OVslow** had the lowest TTC and **both_fast** and **PVslow_OVfast** had the highest TTC. In scenario 3 (merging right), **both_fast** and **PVslow_OVfast** had the highest TTC.

Comparing average TTC for different deviations and for different scenarios

On average, TTC was highest in scenario 1 and lowest in scenario 3. For **noise**, a lower TTC was observed compared with **normal** in all three scenarios. **com_loss_40** and **com_loss_20** had a slightly lower TTC than **normal** on average. **normal** and **com_loss_inside** were almost identical.

In **com_loss_40** and **com_loss_20**, only the OV detected that the situations were dangerous.

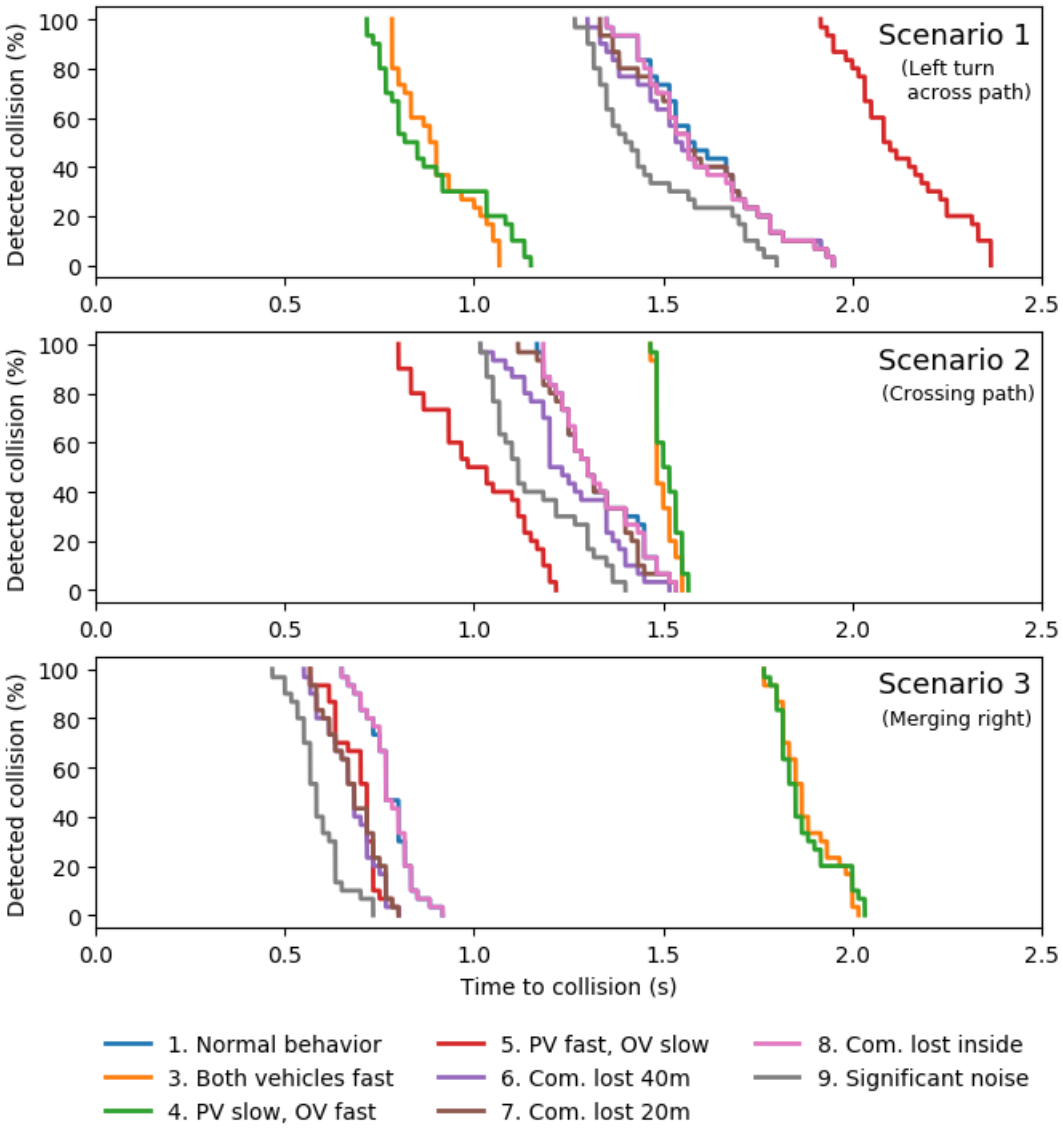


Figure 8.1: Comparison of how early the risk assessment software was able to detect collisions for different deviations in scenarios 1-3.

8.1.3 Discrepancy between expected and observed results

`com_loss_40` and **`com_loss_20`** were expected to have the same TTC values as **`normal`**, but they were observed to be slightly lower. One possible reason for this is that the standard deviation used in the arrival time estimation was too high. Before the communication was lost, when the PV was still a ways away from the intersection, the OV estimated the arrival time for the PV. If that estimation was too forgiving (too high standard deviation in the normal distribution), the probability that the gap was between -1.0

and 1.5 (the gap limits used for the expectation calculation, as described in 5.3.2) could have decreased, due to the normal distribution giving a significant probability that the gap was actually less than -1.0 or greater than 1.5 seconds.

8.2 Evaluation of the combination of risk assessment and maneuver coordination

Here, we present the results of the evaluation of combining the risk assessment software with the maneuver coordination software. We first go through which results we expected and then we present the actual results. Lastly, we point out the differences between the expected and actual results.

8.2.1 Expected results

Collisions

We expect RA+MC (risk assessment and maneuver coordination) to result in fewer collisions than either of RA or MC alone. MC should have collisions in every *collision case* with the deviation **`OV_selfish`**, which would amount to exactly 90 collisions. It is unclear if RA or MC will have the most collisions. This would depends on how often the risk assessment software can detect the dangerous situations early enough so that both vehicles can slow down in time.

RA+MC and MC should have 0 collisions in all deviations other than **`OV_selfish`**, since in these cases the maneuver coordination software should ensure safety. We expect most collisions for RA and RA+MC to occur in scenario 3 (merging right), as the distance from the detection point to the collision point is very short (see discussion in Section 8.1.1).

We expect that RA will have some collisions in deviations other than **`OV_selfish`**. We expect these to primarily come from **`both_fast`**, **`PVslow_OVfast`** and **`noise`**. In **`both_fast`**, **`PVslow_OVfast`**, the OV is driving fast, so it will be harder to slow down in time. In **`noise`**, the time of detection might be delayed by the higher noise.

The severity of the collisions should be much higher for MC than RA or RA+MC, since in MC the vehicles do not slow down to mitigate the collisions.

Emergency breaks

We expect RA to result in much more emergency breaks than RA+MC. RA should have emergency breaks in all collision cases and some semi-dangerous cases, whereas

RA+MC should only have emergency breaks in **`OV_selfish`**. MC should have 0 emergency breaks.

Travel time

The difference in travel time depends on the time added by emergency breaks and the time added by having to wait for the other vehicle to exit the intersection. It is unclear which of these will have the largest impact on travel time. If emergency breaks have a higher impact, then RA should have the highest travel time. Otherwise, RA+MC should have the highest travel time. In any case, RA+MC should have a higher travel time than MC, due to having more emergency breaks and equal waiting time.

We expect the deviations with communication failures to result in an increased travel time, since the vehicles has to wait for communication to return to enter the intersection.

Priority violations and grants

Priority violations can result from two things: emergency breaks triggered by the PV, and the PV having to wait for the OV the exit the intersection. We expect the second case to be quite rare, so overall RA should have the most priority violations, and therefore also the highest priority violation time.

The number of grants in MC and RA+MC should be about the same.

8.2.2 Observed results

Summarized evaluation results are presented below. More detailed results that distinguish between different deviations can be seen in Appendix D.

The kinetic energy involved in a collision is a key determinant for the severity of the collision and the injuries caused [31]. The kinetic energy for an object is proportional to its mass and to the square of its speed. In our tests, we assume all vehicles have the same mass, however, the speed at the time of collision varies. We can therefore compare the severity of our collisions by comparing the speed at the time of collision. Figure 8.2 shows a comparison of the severity of collisions for the different control modes.

Control mode	Nr collisions	EBs	TT (s)	PVs	PV time (s)	Nr grants
RA	35	1755	73349	827	4297	0
MC	90	0	69192	52	52	164
RA+MC	28	116	69765	180	631	168

Collisions

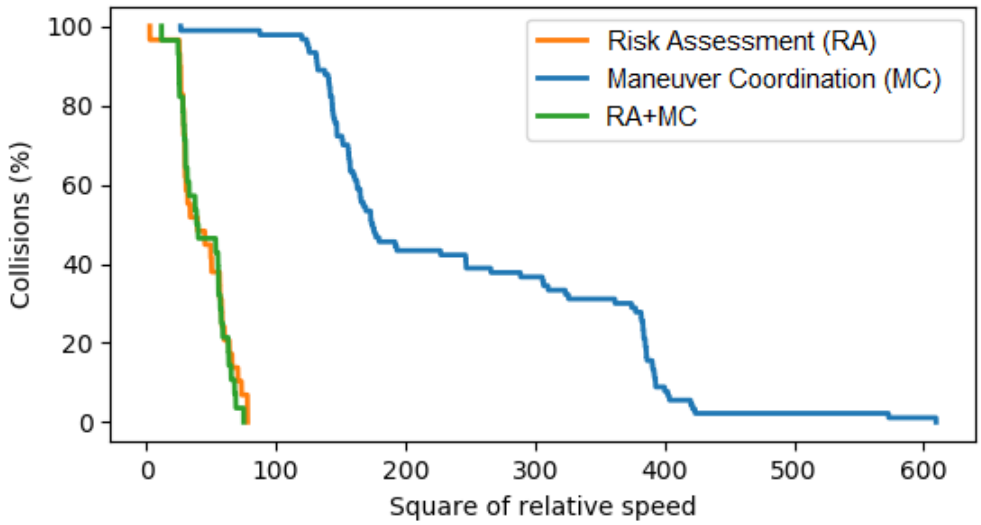


Figure 8.2: Comparison of the severity of collisions for the different control modes. The graph shows the percentage of collisions as a function of the square of the relative speed of the two vehicles at the time of the collision. Note that around 30% of the collisions for MC have a value greater than 380 on the x-axis. That means 30% of the collisions for MC are at least four times more severe than any collision for RA and RA+MC.

MC resulted in the highest number of collisions, and these were much more severe than the collisions for RA and RA+MC. All of the collisions for MC and MC+RA occurred with deviation **`OV_selfish`**, where the OV ignores the coordination software. In all the other deviations, safety was guaranteed by the maneuver coordination software.

All of the collisions for RA and MC+RA came from test scenario 3 (merging right), which is not surprising given that it had the lowest TTC values reported. In the other test scenarios, the risk assessment software was able to detect the dangerous situation early enough to avoid collisions.

The 7 extra collisions in RA compared to MC+RA came from **`both_fast`**, **`PVslow_OVfast`** and **`noise`**.

As expected, RA+MC were almost identical to MC in all deviations except **`OV_selfish`**. There, the backup emergency break helped avoid 62 collisions, and vastly reduced the severity of the collisions that did occur.

Emergency breaks, travel time and priority violations

Much fewer emergency breaks were triggered in MC+RA than in RA. It seems that the

travel time and priority violations heavily depend on the emergency breaks, since RA had the highest values in all these categories.

None of the priority violations in MC were due to emergency breaks, so they must have been caused by the OV taking a long time to exit the intersection. Unsurprisingly, most of them were from **PVfast_OVslow**, where the OV drives slowly through the intersection.

Communication failures

The communication failures did not cause any collisions. With RA, the PV was able to use old information to trigger the emergency break in time. With MC and RA+MC, safety was guaranteed by the maneuver coordination software. However, the vehicles had to wait for communication to return before being allowed to enter the intersection, which created longer travelling times and more priority violations.

8.2.3 Discrepancy between expected and observed results

There were no unexpected results.

8.3 Other tests

We only performed rigorous tests in scenarios with two vehicles, however, the system has been developed to work with any number of vehicles. Informal test runs of scenarios with up to 15 vehicles have been conducted with no apparent loss in performance or safety issues.

Furthermore, the runtime of the risk assessment software developed in this project (NEW) was compared to the one used by Fayaz [10] (OLD), which is an implementation of the work of Lefèvre et al. [5] The runtime of the update step of both software were compared. In the update step, intention, expectation and risk is calculated. With two vehicles, NEW was 20 times faster than OLD, and with four vehicles it was 50 times faster. 400 particles were used in the particle filter in OLD.

9 Discussion and Conclusion

In this section, we discuss the test results from Chapter 8, make some conclusions about the work as a whole, and discuss future work.

9.1 Discussion of test results

The evaluation does not consider different combinations of deviations and each test was only run three times. This limits the claims we can make about the safety of the system. However, the preliminary results are promising.

The risk assessment software was able to reliably detect dangerous situations early enough such that almost all collisions could be avoided. This was true even with significant noise, unpredictable speeds and lost communication. When there was a non-cooperating vehicle, more collisions were observed, however the impact of the collisions was greatly reduced by slowing down. The software was also found to be significantly more efficient than the software used by Fayaz [10]. This would suggest that using intentions and expectations could be an effective way of performing risk assessment in traffic intersections, while still being able to take into account deviations in vehicle behavior. However, more testing would be needed to assert this claim with greater confidence.

The software is also extensible, in the sense that processing steps 1-3 of the software (see Section 5.1) are separable. One could for example quite easily use a different method of inferring the vehicle intentions, while not having to change much about the state estimation or the expectation calculation.

We reasoned about the correctness of the maneuver coordination software in Section 6.2. In the tests where both vehicles used the coordination software, no collisions, no deadlocks and no starvation occurred, which supports the reasoned conclusion. However, more rigorous testing or a formal proof would be needed to make a strong claim about the correctness of the coordination software.

The best results were observed when we combined the risk assessment software with the maneuver coordination software. This was expected, but it gives us confidence that the implementation, which includes the change of priority in the risk assessment software whenever a vehicle is granted, is correct.

9.2 Conclusion

We have seen how risk in traffic intersections can be assessed by modelling risk as the probability that intention and expectation differ. An efficient and extendable software

has been proposed. The software is able to take into account packet loss and uncertainties (noise) in vehicle measurements.

Next, we have proposed a decentralized CIM method for allowing vehicles to safely navigate through an intersection. The method is a maneuver coordination software where vehicles need to ask for permission from other vehicles before they are allowed to perform a maneuver. The software is designed to ensure that no two vehicles with conflicting maneuvers will be inside the intersection at the same time. It takes into account packet loss and packet delay, and is designed to be free from deadlocks and starvation. We reasoned about the correctness of the software.

Furthermore, we combined the risk assessment software with the maneuver coordination software, to create a safer system. The idea is that the risk assessment software works as a backup if some deviant driver does not adhere to the coordination software. The two methods are connected, such that whenever a vehicle is granted, the priority rules of the risk assessment software are temporarily changed in favor of the granted vehicle.

Finally, we have developed an evaluation environment using ROS [24], and carried out preliminary testing of the system. The tests consider three different scenarios, where the vehicles can deviate from normal behavior in a number of different ways, including unpredictable travelling speeds, significant noise in state measurements, communication failures and selfish behavior (i.e. failing to cooperate). Preliminary testing has shown promising results - by combining the risk assessment software with the CIM method, collisions were either completely avoided or greatly mitigated, even while taking into account the deviations listed above. However, as mentioned, the evaluation of the system was limited, and further testing would be needed to make any strong claims about its safety.

9.3 Future work

In the project, we made the assumption that all vehicles know of every other vehicle at the start of the simulation. This allowed us to simplify the implementation and we could more easily reason about the correctness of the maneuver coordination software. One could however imagine that all vehicles are instead connected to some centralized cloud. The cloud's role could be to enable individual vehicles to locate each other, acting as a communication coordinator.

One can also consider an edge computer located somewhere close to the intersection. This computer could perform real-time computation to free up the computation powers of the individual vehicles. The edge computer could for example perform the risk assessment for all the vehicles, instead of the current distributed solution. Another idea is that the edge computer could have the license to dynamically change the priorities of the intersection. This could for example be used to maximize throughput or energy efficiency, or to allow some emergency vehicle to get top priority.

References

[1] World Health Organization (WHO), "Global status report on road safety 2015." http://www.who.int/violence_injury_prevention/road_safety_status/2015/en/.

[2] J. R. Treat, N. Tumbas, S. McDonald, D. Shinar, R. Hume, R. Mayer, R. Stansifer, and N. Castellan, "Tri-level study of the causes of traffic accidents: final report. executive summary.," 1979.

[3] U. D. of Transportation, "Critical reasons for crashes investigated in the national motor vehicle crash causation survey." https://crashstats.nhtsa.dot.gov/Api/Public/ViewPublication/812115.

[4] D. T. S. B. Facts, "Urban areas," *European Commission Directorate-General for Mobility and Transport*, 2012.

[5] S. Lefevre, C. Laugier, and J. Ibañez-Guzmán, "Intention-aware risk estimation for general traffic situations, and application to intersection safety," 2013.

[6] A. Zyner, S. Worrall, J. Ward, and E. Nebot, "Long short term memory for driver intent prediction," pp. 1484–1489, IEEE, 2017.

[7] M. Koschi and M. Althoff, "Interaction-aware occupancy prediction of road vehicles," pp. 1–8, IEEE, 2017.

[8] M. Althoff and S. Magdici, "Set-based prediction of traffic participants on arbitrary road networks," *IEEE Transactions on Intelligent Vehicles*, vol. 1, no. 2, pp. 187–202, 2016.

[9] L. Chen and C. Englund, "Cooperative intersection management: a survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 570–586, 2016.

[10] I. Fayaz, "Reliable Virtual Blinker: Risk Analysis of Dynamic Traffic Priorities," Master's thesis, Chalmers University of Technology, 2018.

[11] A. Casimiro and E. M. Schiller, "Membership-based maneuver negotiation in safety-critical vehicular systems," tech. rep., Chalmers University of Technology, Department of Computer Science and Engineering, 2018.

[12] A. Casimiro, E. Ekenstedt, and E. M. Schiller, "Membership-based manoeuvre negotiation in autonomous and safety-critical vehicular systems," *arXiv preprint arXiv:1906.04703*, 2019.

[13] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. WieBner, "Microscopic traffic simulation using sumo," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pp. 2575–2582, Nov 2018.

[14] C. Sommer, R. German, and F. Dressler, "Bidirectionally Coupled Network and Road Traffic Simulation for Improved IVC Analysis," *IEEE Transactions on Mobile Computing*, vol. 10, pp. 3–15, January 2011.

[15] N. Koenig and A. Howard, "Design and use paradigms for gazebo, an open-source multi-robot simulator," in *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154 vol.3, Sep. 2004.

[16] M. Garzón and A. Spalanzani, "An hybrid simulation tool for autonomous cars in very high traffic scenarios," in *2018 15th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, pp. 803–808, IEEE, 2018.

[17] N. Andren, A. Gil Martin, K. Hoogendijk, L. Niklasson, F. Sandblom, and F. Slottner Seholm, "Predictive control for autonomous articulated vehicles," 2017.

[18] M. Liebner, M. Baumann, F. Klanner, and C. Stiller, "Driver intent inference at urban intersections using the intelligent driver model," in *Intelligent Vehicles Symposium (IV), 2012 IEEE*, pp. 1162–1167, IEEE, 2012.

[19] J. Ward, G. Agamennoni, S. Worrall, and E. Nebot, "Vehicle collision probability calculation for general traffic scenarios under uncertainty," in *2014 IEEE Intelligent Vehicles Symposium Proceedings*, pp. 986–992, June 2014.

[20] O. M. Ponce, E. M. Schiller, and P. Falcone, "Cooperation with disagreement correction in the presence of communication failures," in *17th International IEEE Conference on Intelligent Transportation Systems, ITSC 2014, Qingdao, China, October 8-11, 2014*, pp. 1105–1110, 2014.

[21] O. Morales-Ponce, E. Schiller, and P. Falcone, "How to stop disagreeing and start cooperatingin the presence of asymmetric packet loss," *Sensors*, vol. 18, no. 4, p. 1287, 2018.

[22] V. Savic, E. M. Schiller, and M. Papatriantafilou, "Distributed algorithm for collision avoidance at road intersections in the presence of communication failures," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1005–1012, IEEE, 2017.

[23] A. Varga and R. Hornig, "An overview of the omnet++ simulation environment," in *Proceedings of the 1st International Conference on Simulation Tools and Techniques for Communications, Networks and Systems & Workshops*, Simutools '08, (ICST, Brussels, Belgium, Belgium), pp. 60:1–60:10, ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[24] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, and R. Wheeler, "Ros: an open-source robot operating system," in *ICRA Workshop on Open Source Software*, 2009.

[25] S. Eckelmann, T. Trautmann, H. Ußler, B. Reichelt, and O. Michler, "V2v-communication, lidar system and positioning sensors for future fusion algorithms in connected vehicles," *Transportation Research Procedia*, vol. 27, pp. 69–76, 2017.

[26] S. Lefèvre, D. Vasquez, and C. Laugier, "A survey on motion prediction and risk assessment for intelligent vehicles," *Robomech Journal*, vol. 1, no. 1, p. 1, 2014.

[27] A. Bender, J. R. Ward, S. Worrall, and E. M. Nebot, "Predicting driver intent from models of naturalistic driving," in *Intelligent Transportation Systems (ITSC), 2015 IEEE 18th International Conference on*, pp. 1609–1615, IEEE, 2015.

[28] G. F. Riley and T. R. Henderson, "The ns-3 network simulator," in *Modeling and tools for network simulation*, pp. 15–34, Springer, 2010.

[29] W. R. Inc., "Mathematica, Version 11.3." Champaign, IL, 2018.

[30] A. Molinero Martinez, H. Evdorides, C. L. Naing, A. Kirk, J. Tecl, J. Barrios, M. Simon, V. Phan, and T. Hermitte, "Accident causation and pre-accidental driving situations. part 3. summary report," 01 2008.

[31] D. Khorasani-Zavareh, M. Bigdeli, S. Saadat, and R. Mohammadi, "Kinetic energy management in road traffic injury prevention: a call for action," *Journal of injury and violence research*, vol. 7, no. 1, p. 36, 2015.

A Mimicking a filter

In this section, the "filtering" process is described. The new state estimation $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$ is derived as follows, where $\boldsymbol{Z} = (z_x, z_y, z_\theta, z_s)^T$ is the predefined noise level for each vehicle.

1. Let $\boldsymbol{X}$ be the true value of the state variable.
2. Let $\boldsymbol{Y}$ be a sample from $\mathcal{N}(0, diag(\boldsymbol{Z}))$.
3. Set $\boldsymbol{\mu} = \boldsymbol{X} + \boldsymbol{Y}/3$
4. Set $\boldsymbol{\Sigma} = \boldsymbol{Y}/2$

This way, the estimated mean lies somewhere between the true value and the raw sample. The estimated standard deviation is chosen so that the true value is less than a standard deviation away from the estimated mean.

B Arrival time estimation

If a vehicle is following a speed profile perfectly, we can accurately calculate the time of arrival to some point on its predefined trajectory. We model a speed profile as a piecewise function f with distance as input and speed as output.

Depending on the specific turn, f is comprised of function segments that take two forms, $f(x) = k$ and $f(x) = \sqrt{ax^2 + bx}$. These represent constant speed and constant acceleration. The time t between two distances d_1 and d_2 is calculated as $t = \int_{d_1}^{d_2} \frac{1}{f} dx$.

The estimated arrival time for a vehicle to some point p is modeled as a normal distribution $A \sim \mathcal{N}(\mu_t, \sigma_t^2)$. A depends on the estimated vehicle state $S \sim \mathcal{N}(\mu, \boldsymbol{\Sigma})$, where $\boldsymbol{\mu} = (\mu_x, \mu_y, \mu_\theta, \mu_s)^T$ and $\boldsymbol{\Sigma} = diag(\sigma_x^2, \sigma_y^2, \sigma_\theta^2, \sigma_s^2)$.

The estimation process is described in the following steps, and illustrated in Figures B.1-B.3.

1. Consider the ellipse constructed by a one standard deviation contour line around (μ_x, μ_y). Choose the points p_1 and p_2 on the ellipse, which are furthest and closest to p, respectively. Find the distances d_1 and d_2 at these points. The distances are measured from a point 117.5 meters before the intersection.
2. Let $s_1 = \mu_s - \sigma_s$ and $s_2 = \mu_s + \sigma_s$.
3. Let $k_1 = c(d_p - d_1)$ and $k_2 = c(d_p - d_2)$, where c is some positive constant, and d_p is the distance at p. k_1 and k_2 will increase the further away the vehicle is from p.

4. Construct two new speed profiles SP_1 and SP_2 by adding constants to the standard speed profile SP. The new speed profiles will pass through the points $(d_1, s_1 - k_1)$ and $(d_2, s_2 + k_2)$.
5. Let $t_1 = \int_{d_1}^{d_p} \frac{1}{SP_1}$ and $t_2 = \int_{d_2}^{d_p} \frac{1}{SP_2}$.
6. Let $\mu_t = (t_1 + t_2)/2$ and $\sigma_t = t_2 - \mu_t$

As described, we predict two different arrival times, t_1 and t_2, where one is early and one is late. We then define A based on these times. The further away the vehicle is from p, and the bigger $\mathbf{\Sigma}$ is, the larger the difference between t_1 and t_2, and thus also σ_t becomes. If this difference increases, so does σ_t.

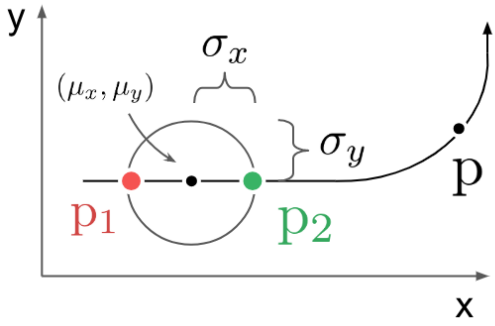


Figure B.1: Finding p_1 and p_2 on the contour line around μ_x and μ_y.

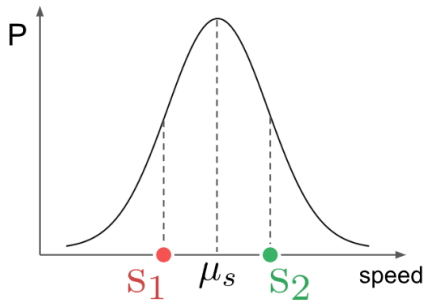


Figure B.2: s_1 and s_2 are the speed values one standard deviation from the mean speed μ_s.

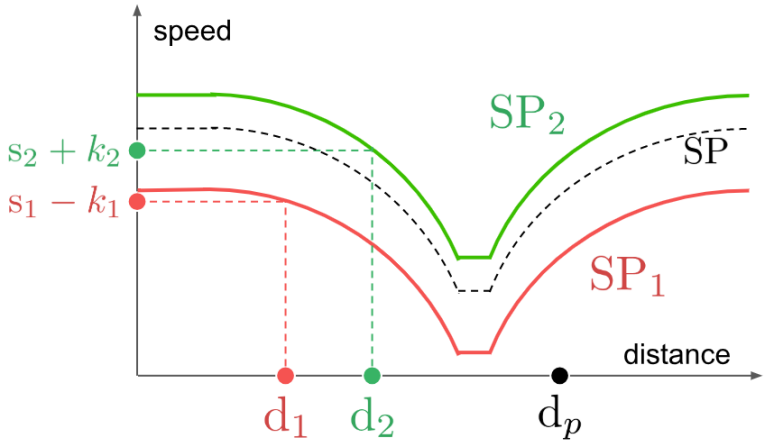


Figure B.3: The new speed profiles SP_1 and SP_2.

C Detailed pseudocode for the maneuver coordination software

The code is python-esque since that is the language the real code was written in.

```
# ---- global variables ----

# whether or not ego vehicle has been (fully) granted yet
granted = False

max_transmission_delay = X #in seconds
ego_id = Y

# temporary list of vehicles that has granted ego vehicle
grant_inbox = []

# temporary list of vehicles ego vehicle has to get a grant from
ask_list = []

grant_list = [] # list of vehicles that ego vehicle has granted
timer = null
ego_turn = Z # left/straight/right
request_line_arrival_time = infinity

# ego vehicle initialized with the starting states and
# travelling directions for every vehicle in the simulation

# dictionary with vehicle id as key. state = (x,y,theta,speed)
vehicle_states = getInitialStatesOfAllVehicles()

# dictionary with vehicle id as key. td = north / south / west /east
travelling_directions = getInitialTravellingDirections()

ego_state = vehicle_states[ego_id]
ego_td = travelling_directions[ego_id]

vehicle_ids = getCarIDs() #list of all vehicles in simulation

def hasLeftIntersection(state, td):
    # check if vehicle has left the intersection

def doesCoursesIntersect(td1, turn1, td2, turn2):
    # check if the 2 courses, described by each (td, turn) pair,
    # intersect
```

```
def hasPrio(td1, turn1, td2):
    # use priority rules to say whether or not vehicle 1
    # has priority over vehicle 2

def getVehiclesToAsk():
    list = []
    for ask_id in vehicle_ids:
        if ask_id == ego_id:
            continue
        ask_state = vehicle_states[ask_id]
        ask_td = travelling_directions[ask_id]
        if not hasLeftIntersection(ask_state, ask_td):
            if not hasPrio(ego_td, ego_turn, ask_td):
                list.add(ask_id)
    return list

def conflictsWithEgoTurn(other_id, other_turn):
    other_td = travelling_directions[other_id]

    return doesCoursesIntersect(ego_td, ego_turn, other_td, other_turn)

def hasTimeToStop():
    # check if ego_state has time to stop before the intersection,
    # using some reasonable deacceleration.
    # if ego_state has already entered and/or left, returns False

def isGapLargeEnough(other_id, other_turn):
    # check if gap for the vehicle with other_id,
    # taking the turn other_turn is larger than 5 sec.
    # gap is only calculated with respect to ego_vehicle
    # use vehicle_states for this

def getTime():
    #get current time in seconds

def tryManeuver():
    ask_list = getVehiclesToAsk(ego_turn)
    if ask_list == []
        granted = True
        return

    grant_inbox = []
    msg = GET(ego_id, ego_turn, getTime(), request_line_arrival_time)
    for vehicle_id in currentAsk:
        send(msg, vehicle_id)
```

```
    timer = Timer(2 * max_transmission_delay, tryManeuver)
    timer.start()


# runs when receiving a GET/GRANT message
def messageCallback(message):
    now = getTime()
    if now - message.time < max_transmission_delay:
        if message.type == GET:

            if message.id in grantlist:
                    grantlist.remove(message.id)

            c = conflictsWithEgoTurn(message.id, message.turn)

            if not c:
                grant_message = GRANT(ego_id, getTime())
                send(grant_message, message.id)
            else:
                l = ego_turn == "left" and message.turn == "left"
                                       and message.id in ask_list
                a = message.arrival_time < request_line_arrival_time

                h = hasTimeToStop()
                g = isGapLargeEnough(message.id, message.turn)
                if h and g or a and l:
                    grant_message = GRANT(ego_id, getTime())
                    send(grant_message, message.id)
                    grant_list.add(message.id)


        if message.type == GRANT:
            grant_inbox.add(message.id)
            if ask_list == sorted(grant_inbox):
                timer.cancel()
                granted = True


# runs when receiving a state message
def stateCallback(message):
    updateVehicleStates(message) # update vehicle_states dictionary

    id_ = message.id
    if hasLeftIntersection(id_):
        if id_ in grantlist:
            grantlist.remove(id_)
```

```
def main_loop():

    if passed maneuverLine:
        request_line_arrival_time = getTime()
        tryManeuver() #this only runs once

    go = granted and grant_list == []

    # move vehicle according to PID, speed model, go, etc.
    updateEgoState()

    for id_ in vehicle_ids:
        send(ego_state, id_) # send this to everyone, including yourself.
```

D Detailed test results

Here, we present the test results from 7.4, while distinguishing between different deviations. Each cell has three values, representing each control mode. (RA / MC / RA+MC)

Deviation	**normal**	**OV_selfish**	**both_fast**
Nr collisions	0 / 0 / 0	28 / 90 / 28	3 / 0 / 0
EBs	226 / 0 / 0	115 / 0 / 116	246 / 0 / 0
Avg. travel time	3,42 / 3,16 / 3,16	3,2 / 2,96 / 3,19	3,26 / 2,91 / 2,91
PVs	113 / 0 / 0	115 / 0 / 111	123 / 0 / 0
PV time	579 / 0 / 0	588 / 0 / 567	687 / 0 / 0
Nr. grants	0 / 18 / 18	0 / 0 / 0	0 / 9 / 18

Deviation	**PVslow_OVfast**	**PVfast_OVslow**	**com_loss_40**
Nr collisions	3 / 0 / 0	0 / 0 / 0	0 / 0 / 0
EBs	252 / 0 / 0	222 / 0 / 0	111 / 0 / 0
Avg. travel time	3,68 / 3,41 / 3,41	3,32 / 2,94 / 2,96	3,18 / 3,38 / 3,38
PVs	126 / 0 / 0	111 / 36 / 42	0 / 0 / 0
PV time	610 / 0 / 0	618 / 43 / 47	0 / 0 / 0
Nr. grants	0 / 0 / 0	0 / 83 / 78	0 / 0 / 0

Deviation	`com_loss_20`	`com_loss_inside`	`noise`
Nr collisions	0 / 0 / 0	0 / 0 / 0	1 / 0 / 0
EBs	117 / 0 / 0	228 / 0 / 0	238 / 0 / 0
Avg. TT (s)	3,2 / 3,33 / 3,33	3,43 / 3,17 / 3,17	3,46 / 3,16 / 3,16
PVs	6 / 9 / 15	114 / 6 / 12	119 / 0 / 0
PV time	21 / 5 / 9	584 / 4 / 8	610 / 0 / 0
Nr. grants	0 / 18 / 18	0 / 18 / 18	0 / 18 / 18