AGENT-BASED ETHICAL DECISION-MAKING CONTROL OPTIMIZATION FRAMEWORK FOR SELF-DRIVING VEHICLES

SHEHU YUSHA'U

Ph.D. THESIS **2022**

Agent-Based Ethical Decision-Making Control Optimization Framework for Self-driving Vehicles

SHEHU YUSHA'U

A Thesis Submitted in Partial Fulfilment of the Requirements for the Degree of Doctor of Philosophy

School of Science, Engineering, & Environment
Autonomous Systems and Robotics Research Centre,
University of Salford, Manchester, UK

June 2022

Contents

List of Figures

List of Tables

List of Acronyms

AIA	:	Artificial Intelligence Approach
AIS	:	Abbreviated Injury Scale
AVs	:	Autonomous Vehicles
DLM	:	Dynamic Lag Modelling
EP	:	Ethical Profile
EPAS	:	Electric Power Assisted Steering
EV	:	Ethical Valence
EVD	:	Ethical Vehicle Design
EVT	:	Ethical Valence Theory
FOL	:	First Order Lag
GPS	:	Global Positioning System
HCD	:	Human-Centred Design
IRL	:	Inverse Reinforcement Learning
LBD	:	Life-Based Design
LBM	:	Linear Bicycle Model
LTL	:	Linear Temporal Logic
MDP	:	Markov Decision Process
MPC	:	Model Predictive Control
NHTSA	:	National Highway Traffic Safety Administration
PID	:	Proportional Integral Derivative
PMM	:	Point Mass Model
POMDP	:	Partially Observable Markov Decision Process
PTD	:	Pure Time Delay
QMDP	:	is the fully observable approximation of a POMDP policy and relies on the Q-values to determine actions.
QPs	:	Quadratic Programs
RMS	:	Root Mean Square
RRT	:	Rapid Random Tree
SOL	:	Second Order Lag
VB	:	Vehicle Behaviour
VM	:	Vehicle Model
VSD	:	Value Sensitive Design

Acknowledgments

Firstly, I am extremely grateful to my supervisor, Prof. Samia Nefti-Meziani, for her invaluable advice, continuous support, and patience during my PhD study. Her patience, motivation, immense knowledge, and plentiful experience have encouraged me in all the research and writing of this thesis. The meetings and conversations were vital in inspiring me to think outside the box and from multiple perspectives to form a comprehensive and objective critique. Besides my supervisor, I would like to thank Prof. Steve Davis and Dr. Theodoros Theodoridis for their insightful comments and encouragement, but also for the hard questions that inspired me to widen my research from various perspectives.

I had the pleasure of working with many colleagues, lab mates, and friends whom I hold in high regard, and I would like to express my heartfelt gratitude to everyone who has assisted me for the stimulating discussions, the sleepless nights we spent working together before deadlines, and all the fun we have had over the last four years. I am grateful to all the technical staff for their assistance and support, especially Mr. Andrew Baker.

I owe a lot to my parents, whose constant love and support keep me motivated and confident. My accomplishments and success are because they believed in me. Deepest thanks to my siblings, who keep me grounded, remind me of what is important in life, and are always supportive of my adventures, and this was no exception. To my loving wife, Mrs. Z. A Shehu, for unconditional, unequivocal, and loving support and to my kids (Ilham, Aysar, Affan, Noor, and Nadia), I'm sorry for being even grumpier than normal whilst I wrote this thesis! You have been amazing, and I will now clear all the papers as I promised!

I acknowledge the generous financial support from the Nigerian government through PTDF (as a sponsor) during the research period. I will forever remain grateful for this rare opportunity. Saving the best for last, I am most indebted to Dr. A. J. Abbas for his support and guidance. You taught me the multiple, complex, and rewarding aspects of research.

Dedication

All praise be to Almighty Allah, my creator, my strong pillar, my source of inspiration, wisdom, knowledge and understanding. He has been the source of my strength throughout this journey, and on his wings alone have I soared.

This thesis is dedicated to my loving parents and beloved wife, without whose constant support it wouldn't have been possible. Also, to my little angels, Fatima (Ilham), Ahmad (Aysar), Abdallah (Affan), Aslamiy (Noor) and Zainab (Nadia), who have been affected in every way possible by this quest. Thanks to my caring siblings, whose prayers go beyond measure. My love for you all can never be quantified.

To my late friend Nasiru Lawal Esha, who left a void never to be filled in our lives, though your life was cut short, your memory will live on as long as I do. May Allah (SWT) grant you Jannah Firdaus. Ameen.

Abstract

When driving, humans balance values like safety, legality, and mobility. When sharing the road with humans, an autonomous vehicle will likely use the same values. Incorporating human values into algorithm design is tough for self-driving vehicle engineers. To address this problem, a decision-making algorithm is designed using philosophical concepts translated into mathematical frameworks. Deontological ethics parallels rule-based mathematical concepts, whereas consequentialism parallels cost-based mathematical concepts. The virtue ethics philosophical principle is also used to motivate the different weightings of path tracking, obstacle avoidance, and traffic regulation compliance. The consequences of different design decisions made in a model predictive steering controller are highlighted by simulation results of a self-driven vehicle negotiating an obstructed two-lane route with a double yellow line.

The value-sensitive design (VSD) iterative process is used to formalise the link between human values and technological needs. A modified VSD technique was used to develop a self-driven vehicle speed control algorithm for pedestrian crossings. The VSD iterations model the problem as a partially observable Markov decision process that was used to generate an effective approach for controlling the vehicle's longitudinal acceleration based on the belief of a pedestrian crossing.

An ethical valence that characterises self-driven vehicle decision-making as a mechanism for claim mitigation, in which various road users make varying moral claims on the vehicle's behaviour, and the vehicle must neutralise these claims while making assessments about its surroundings. With self-driving vehicles, the harm produced by an action and the uncertainties connected with it are assessed and accounted for, leading to an ethical implementation that is realistic. Instead of describing how moral concepts need self-driving vehicles to behave, this approach provides a computational approach that may accommodate a variety of moral positions about what morality demands and what road users could expect.

Chapter 1: General Introduction

1.1 Main Overview

Despite the major success of self-driving vehicles in increasing safety, eliminating various sources of errors, especially due to human cognition, and reducing car crashes, there is still no guarantee that accidents will be completely avoided. A self-driving vehicle is expected to behave properly in such situations to make appropriate moral and ethical choices that reduce the cost of human life, possible injury, or damage, and avoid the obstacles with the highest priority. It is generally believed that they will be more secure than human-driven vehicles, which are more likely to detect and stay away from the dangers and impacts of various drivers and pedestrians. Since 94% of accidents are attributed to driver errors, with 31% attributed to drunk drivers and 10% attributed to occupied drivers (Tanelli, Toledo-Moreo et al. 2018), the case for self-driving vehicles for increased security and lives saved is very compelling. Indeed, even the most optimistic expectations concerning self-driving vehicles demonstrate that they can bring significant benefits regarding social expenses of death or damage, as well as expanded comfort and profitability for the individual consumer.

As experts travelling in an area that includes many road users, from pedestrians to cyclists to other drivers, both self-driving and manned vehicles, computer-controlled vehicles consistently interact with those around them. The idea of these communications is a consequence of the programming in the vehicle and the requirements defined by the software engineers. Similarly, as human drivers show the scope of driving styles, self-driving vehicles offer a large screen on which designers can create responses to different driving situations. In any case, the conduct of the vehicle and its control algorithms will eventually be judged not by measurements or test track execution, but rather by the standards and morals of the public in which they operate.

Humans can go from one location to another in a quick and pleasant manner by driving. People have places to go, people they want to see, and activities they want to do. Automobile mobility gives ease, efficiency, and flexibility to people who drive. This urge to travel demonstrates how essential mobility is in the lives of those who drive. However, drivers are not the only ones who desire mobility. The need for mobility of one road user may collide with that of another road user. A pedestrian, for example, could wish to cross the street in a crosswalk while a motorist is driving down the road. This reveals a disagreement over mobility's worth. The term "value" is defined by (Enke 2020), (Miller and Cushman 2018), as "what a person or group of people considers valuable in life". As a result, mobility might be considered a human virtue.

While driving, another issue is safety. Consider driving along a street when a toy ball unexpectedly rolls into the roadway from behind a huge family vehicle parked nearby. The human driver has no way of knowing whether someone will run out to chase the ball. Human drivers, on the other hand, recognise that they just need to stop or slow down to maintain the situation's safety since that person may be someone's child. Drivers who value safety can safeguard other people and property from a two-ton moving car (Ploeg 2017) (Riaz, Jabbar et al. 2018).

Legality is a third value. Thanks to traffic regulations, humans can safely use the road with other drivers and vulnerable road users. Although traffic regulations are defined as rigid restrictions, human drivers frequently test the limits of these restrictions or completely disregard them. Traffic regulations, however, are rarely severely enforced, according to (Krotov and Silva 2018), (Gogarty and Hagger 2008). and can operate a vehicle in the face of legal issues.

As seen by how they manage the circumstances, human drivers traverse the roads by balancing values such as mobility, safety, and legality. Many other values, such as care and respect for others, fairness and reciprocity, respect for authority, trust, and transparency, are also implicated while driving. One value may take precedence over another, or the values may even clash, depending on the situation (Plyley 2018). Human drivers, luckily, have a means of evaluating which values are most important at any given time (Zachko, Golovatyi et al. 2019).

In the history of the ethics of robots, we still must decide whether artificial agents can truly behave morally without free will (Krotov and Silva 2018). In any case, it appears that other road users and society will interpret self-driving vehicles' activities and the needs put forth by their software engineers from a moral standpoint. The control framework that decides the activities of self-driving vehicles is thoroughly investigated if it causes damage, and social acceptance is greatly influenced by the social interactions that shape daily traffic.

Unexpectedly, the interpretation of philosophical developments and ideas and their mathematical counterparts in the control hypothesis is simple. Thus, similarities can be established between the philosophical theory and the use of costs or constraint functions in the control hypothesis. These enable the execution of ethical principles as either a cost or rules for controlling systems with other objectives. Looking at the issue from the mathematical perspective of determining control laws for a vehicle leads to the conclusion of both (Goodall 2014), and (Qian, Fortelle et al. 2016) that a single philosophical concept is unlikely to be sufficient for programming self-driving vehicles. However, in this research work, the idea of

several moral frameworks, as well as parallels between ethics, consequentialism, and a constrained optimization problem as a starting point, will be addressed.

1.2 Rationale

The research problems are formulated as follows:

1) How can decision-making behaviour be adopted to better respond to risks in navigation and exploration?
2) From a consequentialist perspective, how can these principles best be described as a weighting of costs, which form the more absolute rules of deontological ethics?
3) How do vehicles decide where to go next, and how do interactions with other vehicles affect what needs to be done?
4) How do vehicles use the data provided by their sensors to make short-term and long-term decisions?

1.3 Thesis Contributions

This thesis bridges the gap between decision-making and ethics by accounting for actuation delays in a steering controller, incorporating traffic regulations relating to lane dividers and crosswalks into problem formulations, mapping normative theories to mathematical concepts found in decision-making algorithms, and connecting human values to engineering specifications using a modified form of a generic design technique.

To achieve the aim of the research work, the following objectives are set:

1) To account for steering actuation delay compensation in model predictive control
2) To integrate lane dividers in model predictive steering control and crosswalks in partially observable speed control
3) To Integrate philosophical principles as mathematical frameworks in self-driven vehicle decision-making planning
4) To develop a modified value sensitive design technique for self-driven vehicle decision-making algorithms.
5) To develop an ethical valence algorithm for a self-driven vehicle

1.4 Thesis Organization

The contributions are spread over the chapters and include ethical considerations for self-driving vehicle decision-making. This thesis is structured into six chapters and summarised as follows.

Chapter One: Provides the main overview and highlights issues related to the thesis, challenges, objectives, contributions to knowledge, and summary of the report.

Chapter Two: It examines some of the fundamental concepts and similar works to demonstrate the motivation and headlines for this research work. Including the theoretical background and how the philosophical concepts of deontology and consequentialism relate to technical design decisions.

Chapter Three: It provides a mechanical formulation governing the response of self-driving vehicles that includes the stages taken for the development of the ethical optimization framework and how a model-based design technique was utilised to achieve the research objectives.

***Chapter Four*:** It employs a modified version of the value-sensitive design (VSD) approach to directly relate human values to engineering specifications through iteration over conceptualization, technical implementation, and empirical analysis. In this case, the revised VSD is utilised to create an ethical decision-making algorithm for safe pedestrian crossing navigation. During the conceptualization phase, the ethical values and stakeholders in the situation are identified. In a closed-loop planning technique, the control algorithm takes the form of a POMDP to account for the scenario's uncertainty.

Chapter Five: It offers an ethical valence technique that provides a computational method that is flexible enough to support a variety of "moral perspectives" on what morality demands and what road users could anticipate, as well as an evaluation tool for the social acceptability of an autonomous vehicle's ethical decision-making.

Chapter Six: It summarises the contributions of the thesis, including preliminary work that explicitly accounts for ethical issues in self-driving vehicle decision-making algorithms. There is more work to be done. As part of the recommendation, emphasis was placed on describing how some of the future work may be implemented. As a result, the findings emphasize that ethical considerations are present throughout the self-driven vehicle stack, not just at the decision-making layer, but also in all aspects of vehicle and system design, such as sensor selection, perception layer design, and even how vehicles are tested and deployed.

Chapter 2: Literature Review

2.1 Introduction

Self-driving vehicles are designed with the expectation of decreasing the rate of accidents by eliminating the root cause: the driver. However, there are circumstances in which an accident is probably imminent, if not inevitable, even for a self-driving vehicle. In such circumstances, a self-driving vehicle is supposed to react appropriately. By colliding with different obstacles, different costs will be incurred depending on the damage and the injury. A self-driving vehicle in an impending accident situation should consider these costs and provide a manoeuvre to avoid the highest priority obstacles. In cases when value conflicts develop, autonomous vehicle motion planning necessitates ethical concerns. However, certain technical issues in motion planning do not include value conflicts since they are concerned with the system's functionality. This chapter focuses on ensuring that the proposed trajectories can be performed by accounting for the actuation level in the decision-layer in a computationally efficient manner.

Because it can account for system restrictions while maximising numerous objectives, model predictive control (MPC) has been demonstrated to perform well in autonomous driving applications. MPC also uses a system model to anticipate how control inputs will affect the plant's future trajectory. MPC also uses a system model to forecast how control inputs will affect the plant's future trajectory. Tracking the reference trajectory by reducing lateral deviation in the cost function and specifying obstacle avoidance as a constraint for the motion planning issue established in the previous chapter is one approach to designing an MPC problem. To execute the specified trajectories, however, an autonomous vehicle must steer effectively.

Today's automated cars are modified production cars. A simple external interface to the steering actuation is provided by an electric power assisted steering (EPAS) system incorporating a motor attached to the steering column. On the other hand, EPAS might cause severe delays and compliance issues in the production steering system. It is critical to model and integrate the delay in the MPC formulation if a system has a non-trivial time delay between actuation requests and fulfilment. Path tracking and passenger comfort might be jeopardised if this isn't done correctly.

Actuation modelling in MPC has been extensively studied in the literature, with state propagation or extra delay states in the model being the most common approaches. (Grüne and Pannek 2017) demonstrate state propagation in simulation for systems with well-known signal

delays, and (Bayerlein, De Kerret et al. 2018) address a $20ms$ signal delay with state propagation using a second order Runge Kutta method with validation on 1:43 scale electric cars. As established by (Tjolleng, Jung et al. 2017), an alternate technique to state propagation for pure time delays involves explicitly modelling the signal delay in the model by appending the system dynamics with delay states. In simulation and on small-scale electric automobiles, these methods show state propagation to be a suitable tool for actuation dynamics with known signal delays.

Although actuation modelling is not a new addition to MPC, I am unaware of any comparisons of different techniques using MPC on a full-scale. In addition to pure time delays, the actuation modelling approaches do not consider actuation dynamics. Various actuation modelling problems formulations are assessed using low-fidelity models as part of the research reported in this thesis. Using first- and second-order identifiable system models, the actuation dynamics are integrated into the system dynamics. All the models improve the system's behaviour.

2.2 Self-driving Vehicle Architecture

In this section, the relationships between the motion planning module and different modules of a self-driving vehicle are described. Figure 2.1 illustrates the relationships through the design of a self-driving vehicle (Rasouli, Tsotsos et al. 2018). The motion planning module determines the direction of the vehicle to avoid obstacles, comply with traffic rules, follow the desired instructions, and allow passengers a smooth ride (Xu, Zhao et al. 2019). It is believed that the module gets data about the obstacles, roads, vehicles, and directions from alternate modules (Sadat, Casas et al. 2020).

The information on the obstacles incorporates the position, speed, size, and classification of every obstacle. The road data comprises the road profile, the number of paths, and the size of the paths. The vehicle data incorporates the vehicle's position, heading angle, longitudinal speed, lateral speed, yaw rate, and typical tyre force. The obstacles, road, and vehicle data are provided to the motion planning module from the perception and estimation module (Joa, Yi et al. 2019). Additionally, the desired instructions, including the ideal path and speed, are produced in the behavioural module (Schwarting, Alonso-Mora et al. 2018). To estimate its condition in model predictive control, it is assumed that every obstacle travel at the same longitudinal and lateral speeds as its present speeds. The danger connected with the unpredictable behaviour of the obstacles, as well as the estimated inaccuracy of the vehicle and

obstacle circumstances, are considered while establishing the safety margin of the various potential functions.

(Joa, Yi et al. 2019) predicted the MPC using a linear vehicle model that was built to reflect the vehicle's longitudinal, lateral, and yaw movements but not the roll, pitch, or bounce motions. The vehicle's roll, pitch, and bounce motions do not correspond to the vehicle's movement on the path, but rather to the vertical tyre forces (Cheng, Li et al. 2019), which are thought to be accessible for the estimate module's motion planning module. The vehicle's parameters are also thought to be stable. In any event, if their values vary and the estimation module evaluates the parameters, the vehicle model may be adequately updated by the estimated parameters in all circumstances (Gallardo, Romeo et al. 2017).

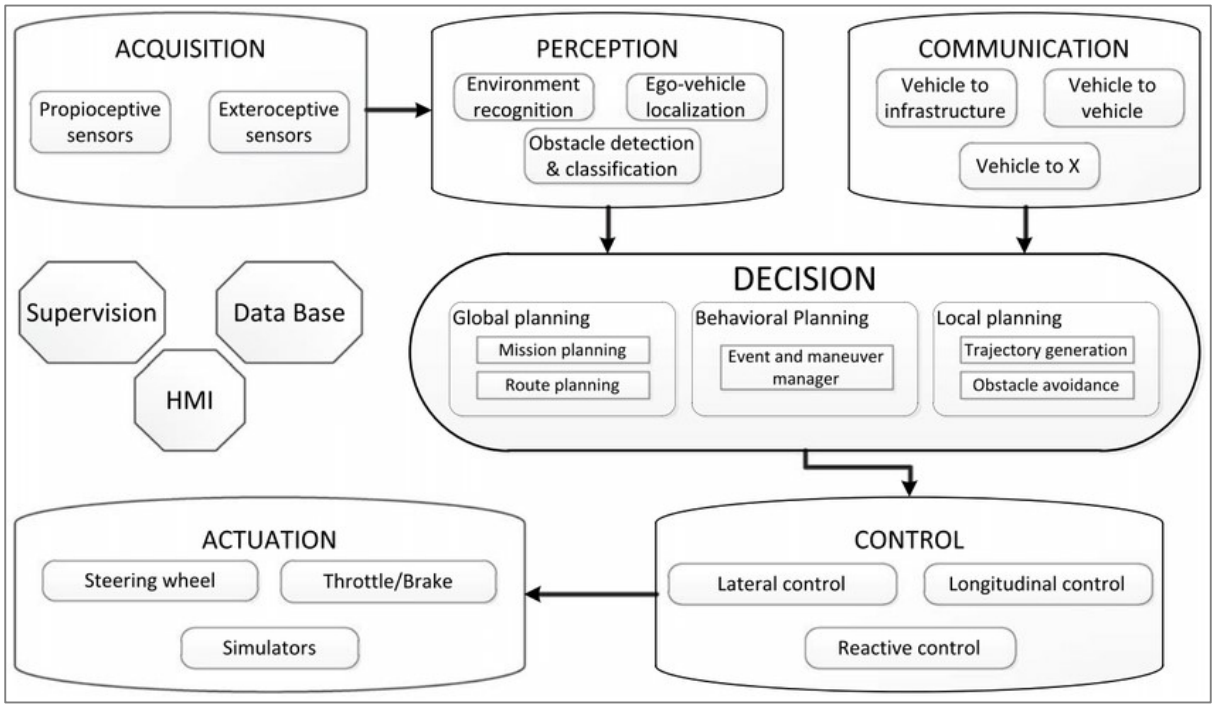

Figure 2.1: Architecture of Self-driving Vehicles

2.2.1 Self-driven Vehicle Motion Planning

Self-driven cars accessing the streets are likely to face comparable circumstances as human drivers, such as sharing the road with other humans and vulnerable road users, transporting human passengers, and unexpectedly confronting objects or people from behind occlusions. A self-driving car will have to traverse the roads while balancing priorities like mobility, safety,

and legality. The issue for autonomous vehicle developers is to build motion planning algorithms that balance these opposing human values.

There are several definitions of motion planning in the literature. Motion planning is defined in this thesis as an algorithm that plans lateral and longitudinal motion in terms of a specified reference trajectory using a mix of steering and acceleration instructions. The reference is a hypothetical path that defines the autonomous vehicle's planned locations and speeds, but it is not guaranteed to be obstacle-free. If an autonomous car is navigating a two-lane highway and encounters a barrier or a pedestrian, the car will need to use a motion planner to decide whether to follow the reference or prevent crashes. Motion planning is appropriate for motion planning in situations like these. Model predictive control (MPC) is an excellent choice for a motion planner because it solves for a series of control inputs by maximising a cost function according to a set of constraints in a receding way along a prediction horizon (Berntorp, Hoang et al. 2019). The cost function, restrictions, and weights must all be determined by the MPC optimization problem designer for the vehicle to achieve goals such as mobility, safety, and legality. Other motion planning approaches could be a suitable fit, and the designers of such algorithms will have to think about the same things.

The most recent studies (Yurtsever, Lambert et al. 2020); (Claussmann, Revilloud et al. 2019)) give a comprehensive summary of self-driving vehicle motion planning. In summary, most conventional methods for determining the are based on one of the principles below. Grid planners (Ghazal, Said et al. 2021), fundamental elements directed to the street, whose major benefit is their simplicity and efficiency, especially in road scenarios, are examples of the discretization of the entrance space with collision verification. The major benefit of random planning, such as quick search in random trees (Cai, Luo et al. 2018), is the probabilistic exploration of huge state spaces while retaining a high degree of calculation. Finally, the limited optimization and control of the receding horizon (Deori, Garatti et al. 2018), which is primarily used for trajectory tracking but can now also calculate trajectories without colliding with other road users, as described by (Brown and Gerdes 2019), who developed a nonlinear predictive control model and applied it to safe navigation in an intelligent vehicle, this was made feasible by recent improvements in nonlinear forced optimization solvers. The major benefits of forced optimization are the path regularity and the direct coding of the vehicle model in the trajectory planning. Constrained optimization only converges to an optimal local path for the vehicle if the problem is not convex.

A set of regulations apply to self-driving cars. These guidelines put constraints on the motion planner, which must always be adhered to. They must, nevertheless, be broken in specific

instances. Like Traditional motion planning methods may be utilised to identify the route with the lowest cost if the traffic regulations are incorporated into the cost function (Althoff, Koschi et al. 2017). It may also describe the rules as logical functions and utilise automated control synthesis to implement them. To generate a discrete model of a robot system and arrive at an objective state, (Singh, Chen et al. 2018), and (Li, Zhang et al. 2017) have developed a technique of movement synthesis that only breaks the rules with the lowest priority for the shortest time feasible. Despite its promise, automated control synthesis has problems when applied to non-deterministic systems and settings, as well as continuous dynamic models like self-driving cars. Similarly, (Vasile, Tumova et al. 2017) looked at the issue of road network violations of minimal limits related to integrated planning and routing. They used syntactically safe linear time logic formulae to define the expected behaviour of the vehicle, as well as a motion planner based on RRT formulae to find the shortest route with the fewest trajectory violations for a single vehicle and trip. Routing for a minimum violation linked to fleet management and vehicle pooling is still an outstanding issue that must be addressed to ensure effective transportation with the least delays.

2.2.1.1 Safety and Mobility

Engineers already make judgments that affect the values of mobility and safety while developing autonomous vehicle motion planning algorithms. (Mohseni, Frisk et al. 2020) investigates the creation of two MPC optimization problems for conducting a constant speed double-lane change manoeuvre along a collision-free reference trajectory at the vehicle's handling limitations. According to Falcone et al., the cost function includes heading deviation, lateral deviation, yaw rate deviation, and steering effort, whereas the constraints include the vehicle model and actuator restrictions. In most systems, the weight placed on heading deviation is higher than in other states. The weight on steering effort is smaller than the weight on heading deviation in the nonlinear MPC formulation, but it is higher than the weight on lateral deviation. The steering effort penalty is two orders of magnitude larger than heading deviation in the linear time-varying formulation, which also includes a limitation on slip angle. Since the authors do not specify how mobility and safety are captured in their algorithms, I interpret the formulations to mean that mobility is achieved by following the desired speed, while safety is achieved by vehicle models and system constraints since the reference trajectory is obstacle-free. (Demirel, Ghadimi et al. 2017) build a limited optimization problem and solve it in a receding manner. Following the speed limit, smoothing acceleration, jerking, and

attenuating extreme yaw rates are all part of the cost function. The driving corridor, as well as the steering geometry and tyre friction restrictions, are all limitations. The publication does not provide the weights used in the tests. The authors also didn't explain how their algorithm incorporates mobility and safety clearly. These formulations lead me to believe that, in addition to occupant comfort, mobility is achieved by adhering to the speed limit wherever feasible, while safety is achieved by avoiding obstructions and road edges.

Instead of solving a limited optimization problem, (Tuncali and Fainekos 2019) construct reference trajectories using rapidly exploring random trees (RRT) and samples from the control space. To randomly sample viable, smooth paths in a congested environment, a generative closed-loop vehicle model with limitations is employed. (Li, Xiong et al. 2019) employ the closed-loop RRT to avoid using a motion planner since the reference trajectory it provides accounts for obstructions as well as the closed-loop speed and steering controllers. Acceleration and steering restrictions are some of the constraints. The closed-loop controllers have gains to choose from, even if there is no cost function. Using various safety systems to overrule the planned trajectory, (Li, Xiong et al. 2019) emphasise safety over mobility considerations, in my opinion. (Meghjani, Luo et al. 2019) explicitly account for mobility as well as safety by penalising collisions and rewarding the vehicle for completing manoeuvres. Although restrictions are not explicitly considered while optimising a POMDP model's policy, Bouton et al. employ a discrete state space to restrict the maximum speed of the manoeuvres and a discrete action space to restrict the change in acceleration instructions. Engineers have included ideals of safety and mobility into the design of motion planning algorithms in several other instances.

2.2.1.2 Legality, Mobility, and Safety

Most autonomous vehicle decision-making algorithms include mobility and safety, but legality is rarely mentioned, if at all (Taeihagh and Lim 2019) and (Lăzăroiu, Machová et al. 2020). To create viable pathways that maintained inside lane markers and considered halting positions at traffic lights and stop signs, all competitors used some variation of a finite state machine. The motion planners were subjected to a speed constraint. The implicit value conflict between mobility, safety, and legality was mitigated by partitioning the decision problem into hierarchical decision structures, where the top-level algorithms considered only the legal rules and bounded the feasible paths for the motion planners to consider only mobility and safety. Only during error recovery, such as when a sensor fails, would there be value conflicts with legality. If the legal criteria were too stringent to provide a viable path, they were trimmed until

an obstacle-free viable path was found. Essentially, unless a fail-safe mode was activated to follow the vehicle code first the hybrid decision architecture proposed by (Schwarting, Alonso-Mora et al. 2018), which considers traffic rules in the mission and reference planners and safety, smoothness, and efficiency in the lower-level behavioural planner (Ahrens 2020), demonstrates that this paradigm of "trying to be legal first".

Another option is to create a controller that enforces adherence to a rule set, such as the traffic code, rather than separating the value dispute in the decision architecture. To create an obedient controller, (Bharadwaj, Carr et al. 2021) use linear temporal logic (LTL) to describe the traffic code. The autonomous car can drive around an impediment on the road and a double yellow line in simulations since the logic rules enable passing a double yellow after the vehicle has come to a complete stop first. (Datta 2019) extend this work by constructing dynamically viable trajectories that follow the rule set using a sampling-based technique known as optimum rapidly exploring random trees (RRT). It is difficult to define a rule set that is accessible, that is, one that does not excessively confine the vehicle's movements. To address this issue, Reyes Castro et al. allowed and chose rule prioritization.

On the other hand, learning techniques that learn from human drivers to balance mobility, safety, and legality are on the other end of the spectrum. Cost function learning using expert demonstrations is used by (Morris, Zhou et al. 2020). The planning issue is expressed as a Markov decision process (MDP), which develops a "perception-to-cost" mapping without explicitly providing any attributes. This strategy obscures the decision-making process and lacks the openness required to comprehend the consequences of mobility, safety, and legality. (Wulfmeier, Wang et al. 2016) use a similar strategy to develop a "perception-to-cost" map, but they describe the characteristics explicitly. Using maximum entropy inverse reinforcement learning, driving style is parameterized in a cost function. To understand how drivers negotiate a blocked route with a double yellow line (Morris, Zhou et al. 2020), specify the characteristics. The autonomous car, with these implementations, represents the demonstrator's mobility, safety, and legality values. As evidenced by the literature cited above, there are several approaches to creating a decision-making algorithm. Some strategies are based on limitations, while others are based on costs, and still others allow for a mixture of the two. The choice of a cost, a constraint, or a weight may appear arbitrary in the general formulation of a motion planning algorithm. How these decisions relate to human values is even less obvious. When constructing motion planning algorithms, an analysis of ethics and human values may help engineers choose what should be a cost or restriction, as well as the weights.

2.2.2 Self-driving Vehicle Behavioural Motion Planning

Most of the above-mentioned work presupposes that a prediction of other road users' future routes is available. Real-life traffic scenarios, on the other hand, entail complicated interactions among several road users (Xu, Zhao et al. 2019) and (Schwarting, Pierson et al. 2019). Dealing with complicated disturbances and modelling interactions with other road users is an issue for self-driving cars that has yet to be overcome(Sadat, Casas et al. 2020). To achieve human-level dependability and react safely even in complicated urban circumstances, self-driving vehicles that operate in complex, dynamic environments require techniques that generalise to unforeseen situations and reason quickly. Accurate perception is necessary for well-informed judgments.

Even though most of the approaches mentioned above summarise perception outside of planning, perception is crucial for self-driving cars. As a result, a brief explanation of the current state of perception is given, followed by a discussion of the end-to-end methods for integrated perception and planning, which produce command inputs for the vehicle directly from sensory data and are typically based on machine learning (Stower 2019). Self-driven Collaboration and supporting choices are required in vehicles with human driving behaviour. Other drivers' intentions must be deduced and included in a planning framework that allows for smart and helpful decision-making without the requirement for vehicle-to-vehicle contact. If self-driving cars must be able to discern the intentions of other road users, they must also allow others to do so. Without the necessity for explicit communication, this results in dependencies and interactions depending on the scene and the behaviour exhibited.

2.2.3 Decision-making for Self-driving Vehicles

Radar, computer vision, Lidar, sonar, GPS, odometry, and inertial measurement units are all used in a self-driven decision-making system to manage the vehicle, destination, and knowledge of its surroundings (Arslan, Berntorp et al. 2017). Sensory data is interpreted by the control systems module to determine the best navigation pathways, obstructions, and signs. These perceptions, along with previous knowledge about the road, traffic regulations, vehicle statuses, and sensor models, are used to select values for the vehicle's regulated inputs that guide its movement (Huang, Ding et al. 2019). The perception module uses the previously collected data to acquire a sense of the vehicle's and its environment's dynamic circumstances; the estimations are then used by the decision module to manage the vehicle to meet the driving

goals. As shown in the decision module of figure 2.1 above, the decision-making system of a self-driving automobile is divided into three components. A path via the road network is designed during the planning stage. After that, a behavioural layer calculates a local driving task that will get the automobile to its destination while adhering to the laws of the road (Luo, Cao et al. 2019). After that, a motion planning layer chooses a continuous path in the environment to carry out a local navigation job. The faults in the scheduled motion execution in the control module are then reactively corrected by the control system.

Self-driving decision-making is also classified as deterministic or stochastic (Sakib 2020). When there is confined or restricted access to the information that is anticipated to determine the probable outcome, this is referred to as a stochastic event. In this case, some outcomes are more likely than others, and the probability of a particular outcome may not be known due to constraints on the best way to determine the likelihood of a particular outcome while decisions are made based on known data, and genuine issues frequently include some obscure parameters.

2.2.4 Control Framework for Self-driving Vehicles

Various control algorithms, ranging from classical control (PID) (Yoon, Shin et al. 2009) to advanced controllers, including back-stepping control, sliding mode control (Manenti 2011), fuzzy logic, and model-based predictive control, have been proposed in the literature for such unpredictable systems (Gray, Gao et al. 2013) and (Goodall 2014). Also, it can systematically manage the system's nonlinearities, uncertainties, and systematic constraints and can optimise the current stage by considering the future stages and trajectories. Predictability distinguishes model-based control from other control systems. The main issue with using MPC is the computational burden of dealing with real optimization, especially for nonlinear models. Since the optimization in non-linear models is no longer convex, it is necessary to meet the requirements of stability and numerical solution. Non-linear models are linearized around an operating point to best meet computer demand (Carvalho, Lefévre et al. 2015). At this point, the linearized MPC problem is converted into specific formats to implement some fast and well-developed convex optimization solvers.

Most reviews suggest using hierarchical control structures (Cunningham, Galceran et al. 2015) and (Du and Tan 2015) with a high-level path planner to create dynamically accessible trajectories that evade all obstacles and a low-level tracking controller to control the vehicle to follow the reference trajectories. Control strategies based on Model Predictive Control (MPC)

(Kong, Pfeiffer et al. 2015) have received increased consideration because of their capacity to explore the state space utilising gradient information. MPC starts iteratively to formulate and solve problems of optimal control with a limited horizon, which are generally solved by nonlinear optimization techniques. MPC begins by iteratively formulating and solving optimum control problems with a restricted horizon, which are often addressed using nonlinear optimization techniques. Each optimization offers an optimal control route for the specified prediction horizon, as well as an optimal system trajectory thanks to the MPC's predictive capacity. This unique characteristic of MPC is ideal for decision-making planning as well as self-driving vehicle tracking control (Zhang, Sprinkle et al. 2015).

The continuous component is handled by MPC-based non-linear motion planners presented in previous work (Paden, Čáp et al. 2016), but the discrete component is not. The major issue is that MPC-based non-linear optimization methods are reliant on continuous, gradient-based optimization methods, which are incapable of dealing with logical restrictions (Houjie, Zhuping et al. 2016). Furthermore, gradient-based optimization may be represented as a local optimum, corresponding to a single manoeuvre option, whereas the global optimum requires the exploration of many manoeuvre alternatives. Some approaches for estimating non-differentiable constraints using differentiable non-linear functions have been developed to address these difficulties (Nilsson, Brännström et al. 2016). Approximations like this, on the other hand, add to the computing load. To adjust to diverse local optima, researchers (Qian, Fortelle et al. 2016), (Yi, Gottschling et al. 2016) and (Garip, Karayel et al. 2017) recommend picking the optimal manoeuvre choice. They do not, however, guarantee global optimality, and designing efficient heuristics in a complicated driving environment is a significant issue.

A kinematic model of the car can be used to control the vehicle at low speeds. Proportional integral derivative control (PID), feedback linearization, or predictive control of the model can all be used to monitor a specified reference path (Wang, Ayalew et al. 2018). For high-speed operation or forceful manoeuvres, however, the entire dynamic model of the vehicle, including tyre forces, must be employed. Forward control, also known as non-linear control, model predictive control, or feedback, stabilises the vehicle's behaviour while it travels the set route. Even in autonomous races, good tracking performance was achieved using these models and car controls. These control techniques are based on a vehicle model that must be predictable (Joa, Yi et al. 2019). For framework identification, there are strategies based on optimization and learning-based approaches (Brazell, Bayeh et al. 2019). The technology used is determined by the amount and type of data available, as well as an understanding of the system's dynamics and control technique. Identifying the online model (Chen, Fang et al. 2019) will enhance the

performance of autonomous cars when road and vehicle conditions change over time. Machine learning technologies have a lot of potential for generating models out of massive amounts of data (Rahman, Xie et al. 2019).

According to (Faulhaber, Dittmer et al. 2019), who suggested threat measures based on dynamic vehicle limitations, the most reasonable method to incorporate human input into the output of the safety system is to linearly mix the two. There, human input was blended with a calculated path depending on the threat's intensity. For example, feedback can be used to achieve shared control (Huang, Ding et al. 2019). Human input may also be directly integrated into an optimization framework in a slightly aggressive manner. The goal is to close the gap between the self-driving system's strategy and the objectives of the driver. The present steering and acceleration inputs, in their most basic form, reveal the driver's purpose. To compute the safe inputs for the common control, (Pan, Xiang et al. 2020) devised a restricted convex optimization. The method, however, was confined to a one-step viewpoint. One of the most common assumptions for intelligent cars is to consider the vehicle's set speed and simply improve it using the steering angle, which simplifies the optimization issue. For example, (Son, Oh et al. 2020) reduced the angular difference between the steering wheel and the artificial control input necessary to maintain safe trajectories. On the other hand, (Yurtsever, Lambert et al. 2020), specified vehicle stability and environmental circumstances to offer safe steering controls in a discrete environment, taking into consideration the vehicle's constant speed and resolving a convex optimization. It is now feasible to optimise concurrently via the steering angle and the speed or the accelerator pedal input (Di and Shi 2021) to achieve minimum intervention thanks to the advancement of rapid nonlinear optimizers.

2.2.5 Learning-Based Approaches

Game-theoretic methods, probabilistic approaches, and partially observable Markov decision processes, such socially compliant driving, are all focused on the framework and model for human-driven vehicle interactions. Consequently, interactions may be exhibited by indirect control of the other vehicle, like an untrained framework, using the data-driven techniques described in (Nar, Ratliff et al. 2017). Based on the features of expert demonstrations, the suggested method learns the reward function from the IRL. Staying on course, preventing accidents, monitoring progress, and limiting effort costs are some of the goals of manually created functions such as cost conditions. Other cars' behaviour is based on a two-player game in which the other vehicle optimises its own reward in response to the self-driven vehicle's

control route. In such situation, the human driver has no choice but to be selfish. The impacts of self-driving car behaviour on human activities can be exploited using this method. The belief status code is associated with one of two discrete cost functions that describe the driver's behaviour, such as attentive or distracted driving. Exploration-exploitation compensation (Peysakhovich and Naecker 2017), unlike comparable POMDP formulations, is not yet processed and is merely coded by a linear combination of goals in the reward function. The weights of the incentive function, on the other hand, can be determined by a human driver iteratively picking a favourite route from a collection of two entrant pathways (Jie, P. L et al. 2018). Without a set of expert routes and predetermined labels, the vehicle may learn the reward function in this way. (Zhang, Chen et al. 2018) demonstrated an enhanced type of algorithm and its effectiveness by constructing pathways that resemble those of humans in parking lots, with less demonstration necessary during learning.

Finally, (Del Giudice and Crespi 2018) and (Terziyan, Gryshko et al. 2018) shown the efficacy of generative learning through conflicting imitation, which has been used to the optimization of recurrent methods. As previously mentioned, a method to learning guidelines based on expert demonstrations is reconstructing the expert's cost function using IRL, then deriving a model of this cost function with better learning (Xu, Dherbomez et al. 2018). Because this method is inherently sluggish, problematic generative learning offers a framework for deriving models from data directly. The method replicates developing human driver behaviour, such as path-changing, while remaining valid over time (Son, Oh et al. 2020).

2.3 Philosophical Principles

Academics, researchers, journalists, and philosophers have been debating moral norms for decades. This section examines these criteria as they pertain to vehicle behaviour. A specialist programmes a self-driving car, and the programming follows a set of decision-making and control logic. While morality and control logic cannot be compared, there are ethical frameworks that are suited for mathematical systems.

The importance of safety in autonomous vehicle motion planning extends beyond the steering actuation system's predictable control. Engineers face new problems while designing control algorithms for driverless cars. Control systems have always had desired requirements and performance criteria for which programmers created control algorithms. The capacity to drive safely and seamlessly in traffic is the ultimate anticipated performance outcome for fully autonomous cars. Because traffic conditions involving people are difficult to quantify, setting

criteria for obtaining this desired outcome is difficult as well. Driving in traffic necessitates that the vehicle adheres to social road-behaviour norms. Expectations such as avoiding collisions and obeying traffic regulations go beyond technological specifications to touch on long-standing and formally defined moral concerns in philosophy. The link between philosophical frameworks and mathematical frameworks used in programming autonomous cars is explored in this chapter to resolve value conflicts.

Accident prevention is essentially driven by the notion of protecting life and preventing harm. (Wang, Vasilakos et al. 2012) defined care (and its opposite, damage) as one of the basic concepts for moral reasoning. Another moral underpinning for a vehicle's compliance with traffic regulations is its level of respect for authority. In addition, interactions with other road users should be based on fairness and reciprocity, which is another of moral foundations (Wang, Vasilakos et al. 2012). The fact that these social expectations for autonomous cars are so well aligned with ethical standards in philosophy implies that philosophy might be a valuable tool for translating such expectations into specifications. (Purves, Jenkins et al. 2015) use social justice in the design of a traffic control framework for automated cars, which is in the same domain as engineering ethics but with distinct applications. (Arkin 2016), as well as (Lin 2016), argue that ethics should be considered throughout the engineering process. (Miller, Wolf et al. 2017) synthesise operations research ethics theories and apply them to ethical decision-making robots. (Shaoshan, Li et al. 2017) also argues that ethical considerations are crucial in the development of autonomous cars.

Both (Guanetti, Kim et al. 2018) and (Merat, Louw et al. 2018) point out that a single conceptual paradigm is unlikely to suffice for programming autonomous systems. As a result, academics have developed solutions that include many philosophical notions. The organised standards for vehicle behaviour are influenced by deontology, a rule-based ethical framework, and consequentialism, a cost-based ethical framework. (Pakusch, Stevens et al. 2018) propose a three-tiered method for determining ethical decisions in autonomous vehicles. The first layer is a rational approach in which the vehicle respects deontological and consequentialist ethical standards. Artificial intelligence and a combined rational-artificial intelligence method are used in the second and third layers, respectively. In an optimum control problem, (Urooj, Feroz et al. 2018) offer the two ethical frameworks of deontology and consequentialism as parallels to constraints and cost, respectively. Many semi-autonomous and autonomous vehicles (Chen, Pei et al. 2019) are already designed using this sort of control formulation. As a result, the work described here employs several ethical frameworks, deontology, and consequentialism as a starting point and applies the principles to a controlled optimization problem.

This chapter has two objectives. To begin with, engineers will be introduced to ethical theories that mirror engineering paradigms to better understand how such frameworks may involve a specific ethical theory. The second objective is to apply these ethical standards to technical decisions that result in acceptable, justified autonomous vehicle behaviour. To reason about driving objectives as norms or costs as suitable, normative ethical theories, such as deontology and consequentialism, are employed. These objectives may be converted into constraints and cost functions that can be used in motion planning algorithms such as the MPC formulation. This allows for morally driven design decisions to be presented and evaluated in a basic traffic situation. Given the formulation of an optimization problem, selecting suitable weights for various purposes might be difficult. Therefore, an ethical theory known as virtue ethics in the form of role morality can help. Role morality and virtue ethics are founded on character alignment (Martinho, Herber et al. 2021). This paradigm, when applied to self-driven systems, directs algorithm development to accomplish desired behaviour for various vehicle kinds. To include ethical reasoning into the design of autonomous vehicle control, an MPC problem is applied to illustrate the influence of various traffic rule formulations. This highlights a broader issue of virtue ethics and the role of morality in self-driven cars, which is addressed in Section 3.7 further.

One of the most important ethical ideas is deontology. According to a set of norms, deontological ethics assesses the morality of one's acts. To be moral, one must, in essence, follow a set of principles that define the appropriate ethical behaviour, and these norms must be followed without exception. Deontological ethics is exemplified by Isaac Asimov's Three Laws of Robotics (Asimov, 1941), which state:

- ✓ A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- ✓ A robot must obey the orders given to it by human beings, except where such orders would conflict with the First Law.
- ✓ A robot must protect its own existence if such protection does not conflict with the First or Second Laws.

The Three Laws of Robotics provide a clear set of behavioural norms for the robots in Asimov's novels to obey, essentially acting as behavioural restrictions. A robot is free to operate as needed if it follows the three laws. Certainly, Asimov's robot novels frequently contain odd behaviour caused by opposing interpretations of these principles, highlighting the limitations of such an approach.

For the development of automated vehicle systems, deontology provides one form of motivating structure: rules that may be created and obeyed on the road. Conditionals and constraints, which are employed in decision-making and control algorithms to restrict and influence system behaviour, are comparable to these principles (for example, a conditional for actuation saturation or a constraint in an optimization problem). Constraints meant to prevent an autonomous vehicle from harming humans, inflicting property damage on itself or other things, or breaking traffic regulations are examples of such restrictions for an autonomous vehicle. The ability of rules to be hierarchical in a deontological framework is a significant characteristic since it establishes clear priorities. From a programming position, the ability to weave dependencies and hierarchies together gives a benefit of clarity in thinking for the algorithm's development. However, if an algorithm is designed strictly deontologically, it may create too restricted driving goals.

Consequentialism is a major normative ethical theory that examines the moral acceptability of actions exclusively because of their effects. To overcome the limits of deontology, consequentialism is sometimes presented as the opposite of deontological ethics. There are numerous varieties of consequentialism, but the focus here is on utilitarianism, which is a kind of consequentialism. Utilitarianism examines a scenario's expected utility and assesses the implications of actions depending on which provides the greatest outcomes (Enke 2020). Consider Santa Claus being hurt and needing to be rushed to the hospital since he gives so much good to the world via his numerous gifts. By using consequentialism, the ambulance driver may justify breaching traffic regulations and taking additional measures as needed to speed up Santa's recovery. Consequentialism has drawbacks as well, since defining what is "good" may be difficult.

Consequentialism, as a more precise form of utilitarianism, provides a foundation for thinking about ethical behaviour in an optimization problem. In control theory, optimum control employs a mathematical solution to identify the best control action. The viable option that minimises the cost function is the optimal control action (i.e., the morally correct decision) (i.e., the desired outcome toward which one strives). Minimizing damage to vehicle occupants might be one example of such a cost function for autonomous cars. In consequentialist form, the best option would be to manoeuvre the car to fulfil the aim of minimising injury to the occupants at all costs. This method has several drawbacks, such as the difficulty in creating or assessing the cost function (as with concepts like "damage" (Sadat, Casas et al. 2020) or making that cost function comprehensive (by, for instance, considering road users other than the occupants in this case).

2.3.1 Deontological Constraints

Cost functions examine the impact of various measures on several conflicting goals. To prioritise objectives, optimal controllers pay greater attention to higher cost or weight targets, placing the related costs considerably above those of the other objectives (Arkin 2016). This, however, only works if certain conditions are met. If some costs are orders of magnitude higher than others, the problem's mathematics may become unconditioned, resulting in rapid changes in inputs or extreme actions. These problems can be encountered in both mathematics and philosophy. Furthermore, hidden concessions in a cost function for some reasons may have an impact on the real meaning or priority of specific goals. While it may appear rational to punish both steering adjustments and pedestrian accidents, these goals are clearly prioritised. Rather than attempting to make an accident a thousand or a million times more expensive than a change in steering angle, it's better to define optimal behaviour in more absolute terms: the vehicle must maintain a strategic distance from hits, regardless of steering direction strength. As a result, the objective shifts from a consistent cost-cutting strategy to an ethical application of certain standards. These objectives can be defined numerically by imposing constraints on the optimization problems. Limitations may take on some forms that mirror the practises enforced by scientific rules or the framework's stated constraints. They might also indicate system limitations that should not be exceeded.

Restrictions on an optimal control problem can be used to model the control law by restricting vehicle traffic to paths that avoid pedestrians, cars, cyclists, and other obstacles, and can be used to capture ethical rules associated with an ethical manner, such as the goal of avoiding collisions with other road users. If there were a set of actions or control entries that could be performed to avoid a collision, the vehicle would never have one, and no other goals could alter or replace it. Some road codes may be coded simply as restrictions. As a result, the mobility of the chosen vehicle may be limited, either physically or ethically. In most cases, it is possible to drive comfortably while still adhering to all traffic regulations and keeping a safe distance from other road users. However, in certain circumstances, dealing with the restrictions put on the problem is beyond the range of possibility. These might be situations where death is inescapable from a moral standpoint. However, far more benign confrontations are both conceivable and prevalent.

These scenarios, in numerical terms, describe possibilities that are scientifically impossible to realise. As a result, there is no control input that can meet all the vehicle's movement restrictions. The more limitations are put on the vehicle's mobility, the more likely it is to

encounter a dilemma in which a constraint must be broken. Beyond the mere assertion that there is no ideal action, a vehicle must be selected to accomplish anything under these circumstances. When dealing with constraints in optimization problems, it's common to treat the constraint as a "soft constraint" or slack variable. The restrictions are typically valid, but if the issue becomes unsolvable, the solution will change it at an exorbitant cost. In this way, the framework may be certain that it will discover a solution to the problem and will make every effort to minimise constraint violations. A hierarchy of constraints would surely be enforced by giving the cost of certain constraint breaches a larger weight than the cost of others. When the vehicle reaches a dilemma scenario, it functions according to rules or ethical constraints until it confronts a dilemma, at which time the hierarchy or weight put on various constraints resolves the problem using a consequentialist method. In the presence of practicality, this becomes a dual system of ethics, in line with the philosophy offered by (Miller, Wolf et al. 2017) and (Urooj, Feroz et al. 2018), which addresses some of the difficulties with adopting a single ethical framework stated by (Arkin 2016).

The Three Laws of Robotics, proposed by science fiction author Isaac Asimov, are the most well-known hierarchical framework of ethical standards for self-driving cars. These guidelines do not provide a comprehensive ethical framework and would not be sufficient for moral behaviour in a self-driving car. When these principles were applied to real-life circumstances, Asimov's actions caused conflict. Nonetheless, beginning with the First Law, this fundamental framework is well adapted to addressing some of the moral issues that might arise. This law emphasises the value of human life and the responsibility of a self-driving car to protect it. The ability to reduce the number of accidents and fatalities is a key driver for the development and implementation of self-driving cars. As a result, it appears acceptable to position human life protection at the top of the hierarchy of rules for self-driving cars, which is essentially identical to the scenario in Asimov's laws.

Could it be enough for the car to just avoid a collision rather than aim to reduce human injury? The most common way for a human to be injured in a self-driving car is through direct contact during a collision. Limiting the responsibility to avoid collisions would mean that the vehicle would not need to be modified to sacrifice itself to save human life in an accident in which it was not involved. In theory, the moral responsibility is to avoid causing damage rather than to initiate a collision. Incidents with defenceless road users such as pedestrians or bicycles may be prioritised over collisions that just result in property damage.

In a strictly logical calculation, such an approach would not produce the optimal results. A small pedestrian injury may be less expensive than material damage. Collisions, in any event,

must be extremely uncommon. Due to restrictions imposed by scientific laws, careful design of the control system configuration would allow self-driving cars to avoid any collisions that are preventable. In exceptional instances where collisions are unavoidable, society can tolerate inferior results in self-driving vehicles that explicitly weigh human life in relation to other objectives to gain clarity and comfort. These are clear rules that can be actualized in a self-driving vehicle and organised by the correct decision about the violation of slack variables of the constraints. Prioritizing human life and the most vulnerable road users and expressing the resulting hierarchy in Asimov's laws, these are clear rules that can be actualized in a self-driving vehicle and organised by the correct decision about the violation of slack variables of the constraints. Such moral standards would just necessitate object categorization rather than attempts to improve harm estimates. This might be accomplished by utilising current recognition and perception systems, which do not always classify objects properly.

2.3.2 Consequential Cost

The basic strategy of ideal control - the selection of information sources that improve a cost function - is like the consequentialist rationality approach. Alternative utilitarianism aims to discover if the control inputs of this expanded model offer the ideal outcome in the moral sense when the moral implications of an action may be a cost function. The ideal controller follows the consequentialist approach to rationality since the vehicle may re-examine or act on its control data sources to reach the optimal result in a random scenario. We assume, as a theoretical model, that all elements of the ground may be weighted according to the degree of risk they pose to the vehicle. (Van den Hoven, Lokhorst et al. 2012) presented such a structure as a model of human driving that is dependent on Earth's values, and it has led to various ideas for the management of autonomous driving or driving assistance. These include (Lin 2016), the mechanical field approach potential (Wintersberger, Frison et al. 2017), the virtual guards of Donath and its partners (Binns 2018) and the work of self-driving vehicle control according to the risk possibilities by (Goodall 2014). The control calculation at that point begins with a critical direction for the engine, brakes, and controls that move the vehicle along that lane.
The construction of an acceptable cost function is the most important test of these approaches in terms of design and logic. The fundamental example offers a cost function for a single vehicle's danger. A more comprehensive societal stance would be reflected by a holistic technique. One possible solution is to calculate the cost of injury to various road users and consider it as a cost that may be reduced. Depending on the circumstances, the expenses may

include property damage or even death. Such consideration would need a large amount of measurement data on environmental obstacles as well as ways for assessing the possible effects of impact scenarios, potentially by overcoming the facts emerging from previous incidents. Aside from the applications that this consistent technique employs with the data, the behaviour of such cost functions as a result offers certain challenges. If it is anticipated that such an endeavour will be considerably described or addressed, the vehicle will attempt to minimise damage in the case of a worldwide problem and, as a result, decrease the societal effect of accidents. However, in such circumstances, the car may make a movement that further harms the occupant or the vehicle owner to prevent harm to others. Such benevolent tendencies may be good for business, but they are unlikely to be appreciated by the vehicle's owners or tenants. Take, for example, a vehicle that is primarily concerned with the residents' safety. With a few exceptions, the world perspective triumphed in the vehicle structure. Wax models, for example, and the resemblance to pedestrian accidents. Because a collision with a stroller would not threaten the car's rental of another vehicle, a vehicle that endangers the safety of its occupants cannot endangers the safety of pedestrians. Such cars are unlikely to result in a significant reduction in the frequency of traffic accidents during rush hour, and they are unlikely to improve social recognition.

With all these challenges in defining an acceptable cost function and getting the necessary data to accurately calculate the cost of actions, a straightforward consequentialist method that employs cost function capabilities to code the morale of robotic vehicles looks unfeasible. In any case, the primary notion of lowering the costs of punishing undesired actions or encouraging desired activities might be a beneficial and important aspect of the control calculation, both for physical observation, such as motion planning, and moral issues. The morality of friendliness described in this framework by (Krotov and Silva 2018) and (Pimentel and Bastiaan 2018) for computerised cars, for example, may be integrated since delays can be viewed as a cost function.

2.3.3 Ethics and Self-driven Vehicles

The iconic trolley dilemma may come to mind when the subject of autonomous cars and ethics comes up. The trolley issue depicts an unstable trolley that has broken brakes and has become free on a set of railway rails. Five people will be killed if the trolley continues its path. As a bystander, you have the option of intervening and diverting the trolley to another set of rails where an unsuspecting victim will undoubtedly die (Rehman and Dzionek-Kozłowska 2018).

If an autonomous vehicle takes the place of the trolley, an algorithm will have to decide whether to continue a collision path with five people or swerve and kill a random pedestrian. This would be a bad situation for an autonomous car (or anybody else) to find itself in on the road. As a result, there is a lot of research being done to try to figure out what the problem is and how to solve it.

Along the same lines as figuring out how to solve the trolley problem, some attention has been paid to the public's preferences for what an autonomous car should behave in a crash scenario when it must choose between striking one thing and striking another. In these preference surveys and trials, it is anticipated that an autonomous vehicle would be able to target a specific entity positively. Most of the participants wanted the vehicle to follow a utilitarian approach, sacrificing the few for the many (Rehman and Dzionek-Kozłowska 2018), (Faulhaber, Dittmer et al. 2019) and (Gupta, Vasardani et al. 2019). The societal conundrum emerges when considering whether an autonomous vehicle should be designed to follow utilitarian principles. The authors conclude that the participants do not want the autonomous car to be utilitarian or to follow their own preferences, but rather that manufacturers and legislators should decide how the car should operate in certain tragedy situations. The trolley dilemma was initially proposed by philosophers and ethics researchers to encourage engineers to consider the implications of their design decisions to avoid developing targeting algorithms unintentionally (Halalae and Miclosina 2019). As proposed by (Milfont, Davies et al. 2019), programming an autonomous vehicle to accept a user-defined priority or "command" list for collision situations has resulted in solutions (and, in essence, targeting algorithms). In specifically, (Schwarting, Pierson et al. 2019) used a priority list in conjunction with an object categorization algorithm to tell the autonomous car what action to take in a collision scenario, either on the occupants' personal ethics or a manufacturer-supplied default list. The crash scenario's outcome is predictable if the categorization is correct, according to the design. (Wagner, Borenstein et al. 2019) suggest the democratisation of preference learning, based on the findings of preference surveys that users are less likely to incorporate their ethics (i.e. utilitarian choices) into the vehicle. In the event of a trolley, if the vehicle fails and the law does not apply, the public's collective voices will determine who to target.

Some people utilise the concept of social welfare to spread damage more evenly rather than focusing remedies for instances. (Zhang, Shao et al. 2019) propose evaluating an objective function that multiplies the passenger and pedestrian utilities such that nonbinary steer angle solutions emerge in a trolley problem variant where the autonomous vehicle is ferrying a human occupant and it must suddenly swerve into a wall or kill a pedestrian. In other words,

an autonomous car with this objective function may try to strike both the wall and the pedestrian, causing little injury to all parties involved. This shifts the focus away from survey data and toward game theory techniques like those developed by (Martinho, Herber et al. 2021), who formulates social justice as a maximum algorithm. Rather of presuming a certain aim, social justice approaches maintain impartiality when it comes to the agents.

The recommended trolley scenarios solutions address a problem that is not the same as a motion planning issue. The autonomous vehicle continually analyses how to drive laterally and longitudinally to follow a particular reference trajectory and prevent accidents during motion planning. In crash situations, the trolley scenario solutions concentrate on making last-minute targeting judgments. Zawieska shows that there is a significant turning point in terms of scenario risk far before an accident (Zawieska 2020), implying that an autonomous vehicle equipped with the right motion planner might completely avoid trolley situations. This thesis takes a step back from trolley issues to address philosophical frameworks and human values in a broader sense to assist engineers in designing socially acceptable and reasonable motion planning algorithms.

2.3.4 Ethical Issues

The term "autonomy" originates from philosophy and refers to the limit of human power to legislate, formulate, deliberate, and choose to obey norms, rules, and laws from a moral standpoint (Kitchener 2016). This involves the freedom to choose one's own models, as well as the ability to define one's own goals and purposes in life (Lin 2016). The cognitive mechanisms that sustain and enable human dignity and action per excellence are inextricably linked. They usually contain qualities like self-awareness, self-consciousness, and self-development as grounds for reasons and values (Shaoshan, Li et al. 2017). Humans are the only creatures that have autonomy in the moral sense of the word. Even when dealing with highly advanced adaptive systems, using the word "autonomy" to describe basic items is incorrect (Miller, Wolf et al. 2017). Though, in scientific literature and public discussion, the word "autonomous" systems have been extensively used to highlight "the greatest level of automation and the maximum degree of human independence in terms of operational and decision-making autonomy" (Guanetti, Kim et al. 2018). However, autonomy in the novel sense is an essential part of human dignity that cannot be relativized.

In actual sense no intelligent system, no matter how sophisticated, can be described as "autonomous" (Miller, Wolf et al. 2017). In the original ethical sense, it cannot be recognised

as a moral person and inherit human poise (Pakusch, Stevens et al. 2018). Human poise as the basis of human rights implies that important human intervention and interest must be conceivable in areas of interest to humans and their environment. Unlike the automation of production, monitoring and deciding for humans in our own way is inappropriate, even if technically feasible (Urooj, Feroz et al. 2018). They should almost certainly figure out which goals are served by innovation, what is ethically relevant, and which actual objectives and conceptions are morally qualified for the search. This can't be left to robots, regardless of how amazing they are.

The ability and willingness to assume and assign moral obligations is a fundamental component of human origin that underpins all our ethical, social, and legal organizations. Moral obligation is interpreted here in a broad sense in which it might denote a certain part of human behaviour, such as responsibility, risk, causality, liability, receptive dispositions, and moral obligations related to social norms. Moral obligations, in whatever sense, cannot be assigned or transferred to "autonomous" innovation.

2.3.5 Ethics of Crashing

During and before accidents, human drivers are prone to making poor judgments. They must overcome tight time restrictions, a lack of manoeuvring expertise with their vehicles, and restricted sight. Self-driving cars, on the other hand, have far fewer resources for recognition and processing capacity. This method has hampered research on car moral studies by preventing the examination of future sensors and algorithms from eliminating all accidents (Operto 2011), (Muehlhauser and Helm 2013), and (Goodall 2014). Even if perfect vehicles should occasionally fail, an ethical decision-making system is still required (Malle, Scheutz et al. 2016)

These cutting-edge self-driving cars, which are outfitted with cutting-edge software and sensors, can make pre-crash judgments that properly detect adjacent vehicle trajectories and avoid high-speed manoeuvres. They will most likely transcend the constraints that humans face in this manner. When a collision is unavoidable, a computerised vehicle can choose the best course of action based on safety considerations and the likelihood of the outcome much faster and with more precision than a human driver (McBride and Hoffman 2016). On highways, it is typically more effective to brake and swerve during high-speed movements, thus the computerised system may conclude that braking alone is not optimal. The major flaw with self-

driving cars is that their judgments in the case of an accident are predetermined by a system programmer, rather than a human driver who can make decisions in real time.
The self-driving car can interpret and make decisions based on sensor data, but the selection is based on a logical sequence that was designed and coded months or years before. If a collision can be avoided, this procedure is simple: the car chooses the safest path and proceeds. If injuries cannot be avoided, the self-driving car must choose the most effective way to create an accident. This decision becomes a moral one (Redelmeier and Raza 2017), (Sarathy, Scheutz et al. 2017), and (Wintersberger, Frison et al. 2017).

2.4 Vehicle Model Design

Deontology and consequentialism's frameworks are the product of much logical investigation (Lin 2016) and (Kenwright 2018). Based on their original considerations, we will use these systems as tools to advance and clarify the selection of self-driving programming schemes. In this research work, a definition of the problem by the MPC is given since the explicit thinking on limits and costs in the MPC corresponds well to the standards of deontology and consequentialism. It will also show that the test of these two philosophical structures leads to an orderly treatment of the various questions. Requirements for the vehicle are set to maintain a strategic distance from accidents, follow dynamic conditions, and stay within the limits of its abilities. In terms of cost-effectiveness, the vehicle is geared towards ideal results by following a recommended path and providing incentives and value for human life. Interestingly, with these different goals, it is less certain that traffic rules are complicated or compelling, so exceptional representations are investigated.

2.4.1 Vehicle Models

The vehicle models used for the trajectory planning of self-driving vehicles are classified as point-mass vehicle models, kinematic vehicle models, and dynamic vehicle models.

2.4.1.1 Point Mass Model

Point mass models are straight models demonstrating the vehicle as a particle whose mass can move at both longitudinal and lateral speeds. The tyre model and vehicle geometry are not considered and can cause significant errors. In this manner, certain state constraints can be

added to enable the generated direction to be increasingly possible. Increased longitudinal and lateral speeds can be enforced by the acceleration associated with the most extreme tyre constraint limits (Gao, Lin et al. 2010), (Funke, Brown et al. 2015), (Jalalmaab, Fidan et al. 2015), and . In addition, the vehicle's sideslip edge can't be substantial for a vehicle in a non-drifting manoeuvre and can be forced, (Kong, Pfeiffer et al. 2015). Even with these limitations, a point mass model can't satisfactorily anticipate the vehicle's behaviour.

2.4.1.2 Kinematics Model*s*

Kinematics model are nonlinear models that show how a vehicle's geometry affects its performance. No tyre model is considered. However, to satisfy the passenger's comfort and avoid skidding, constraints may be imposed on the lateral acceleration to restrain it to ordinary driving values (Kong, Pfeiffer et al. 2015), (Sheth and Umbarkar 2015), and (Zhang, Sprinkle et al. 2015, Erlien, Fujita et al. 2016).

2.4.1.3 Vehicle Dynamics Models

Vehicle dynamics models consider tyre models in their models (Erlien, Fujita et al. 2016). (Houjie, Zhuping et al. 2016), (Nilsson, Brännström et al. 2016) and (Qian, Fortelle et al. 2016) compare the behaviour of a vehicle kinematics model and an open-loop vehicle dynamics model. The outcomes demonstrate that both models perform very similarly at low speeds in modelling vehicle behaviour. At higher speeds (more than $15m/s$), the dynamic model works much better if the manoeuvre includes steering angles of more than $1{:}5^0$ (Chae, Kang et al. 2017). Thus, when a self-driven vehicle is expected to perform high-speed manoeuvres with large lateral accelerations, a vehicle dynamics model is preferred over a vehicle kinematics model for use as an MPC model.

Vehicle dynamics models are derived based on Newton's second law, which states that wheel models are considered manoeuvring forces. The dynamic equation of the vehicle is non-linear, regardless of the tyre model, but the main source of non-linear behaviour of the vehicle is the wheel. The tyres have limited capacity and become saturated (Demirel, Ghadimi et al. 2017). They present a dynamic model of a four-wheeled vehicle with longitudinal, lateral, and yaw rate equations in the centre of gravity of the vehicle based on the four forces exerted on all four wheels. They used a Pacejka tyre model and studied the dynamics of the wheels in the model. They also consider load transfer resulting from longitudinal and lateral accelerations in the tyre

model to generate a more accurate vehicle model. Wheel dynamics increases the number of vehicle states by four but has a very small impact on vehicle model accuracy (Funke, Brown et al. 2017), and (Gallardo, Romeo et al. 2017) they used a four-wheel vehicle dynamics model without wheel dynamics, nor did they consider the effect of load transfer in the model.
A dynamic model of a four-wheeled vehicle without wheel dynamics can be simplified into a bicycle model. In a bicycle model, the tyres on each axle are modelled as rubber tires. A bicycle model is not linear (Siampis, Velenis et al. 2017). Look at the bicycle model with a Pacejka tyre model. They limit the lateral slip angles of the front and rear tires, since the large slip angles are not favourable. (Bayerlein, De Kerret et al. 2018) and (Berntorp, Hoang et al. 2019) consider a linear tyre model for the vehicle while the equations of motion of the vehicle are non-linear. They also force the tyre sideslip angles to keep the tyre in its linear force region and keep the vehicle model valid.

2.4.1.4 Linear Bicycle Model

The vehicle dynamics models are non-linear, and the MPC that uses them should be non-linear as well. However, when vehicle dynamics are considered, a linear bicycle model can be used in a quadratic MPC to deal with high-speed manoeuvres with significant lateral acceleration (Manenti 2011), (Iftekhar and Olfati-Saber 2012), and (Mladenovic and Abbas 2014) developed a nonlinear bicycle model with a Pacejka tyre model and longitudinal load transfer. They then linearize the model around the operating point. They also force the overall acceleration of the vehicle to stay in the friction circle. The circle is approximated by half-spaces so that the quadratic constraint is approximated by linear constraints to be utilised in a quadratic MPC.
Also, (Turri, Carvalho et al. 2013) developed a four-wheel vehicle model for vehicle lateral motion with a Pacejka tyre model where the longitudinal motion of the vehicle is known. They also consider load transfer in the model. At that point, they linearize the vehicle's model. They calculate the rear and front tyre models based on the total longitudinal force of the vehicle and linearize them. In this way, they consider the load transfer as well as the combined sliding effects. (Aripin, Md Sam et al. 2014) and (Gao, Gray et al. 2014) presented a nonlinear bicycle model with a Pacejka tyre model to model the lateral movement of the vehicle, then linearized the model. They utilise a linear tyre model for the rear tyre force and limit the slip angle of the tire. They estimate the cornering stiffness of the tyre and the most extreme sideslip angle of the tyre to obtain the best possible approximation of the tire's behaviour and to generate a lateral

force close to the maximum lateral force. For the front tire, they use the tyre force in the motion equations and derive the steering angle using the inverse Pacejka model. They also limit the slip angles to maintain the tyres in their linear force regions.

(Nilsson, Brännström et al. 2016) demonstrates a nonlinear bicycle model with a brush tyre model for the lateral motion of a race vehicle. Like (Gao, Gray et al. 2014), they utilise a rear tyre model for the front tire. They assume that the nominal curvature of the path is the curvature of the road. Thus, rather than utilising a linear tyre model for the rear tire, they linearize the brush model by the nominal sideslip angle corresponding to the nominal curvature of the path. This work is intended for racing vehicles that operate on high-speed bends that necessitate a large lateral angle to follow the course. For road vehicles, little tyre sideslip angles are required to follow the path, and the resulting tyre model would look like a linear tyre model. The paper also limits the front lateral force and applies a stability envelope to the rear tyre rather than restricting tyre sideslip angles. The envelope limits the yaw rate to its maximum steady state value corresponding to the road tire-road friction and the rear sideslip angles to the boundaries corresponding to the linear force region of the tire.

For example, (Qian, Fortelle et al. 2016, Funke, Brown et al. 2017) demonstrate a nonlinear bicycle model with a brush tyre model for the lateral motion of a racing vehicle. For the front and rear tires, they linearize the brush tyre model by the nominal sideslip angles that correspond to the nominal curvature of the path. They utilise a stability envelope like that of (Yi, Gottschling et al. 2016). To limit the lateral tyre force of the front tires, they consider the combined sliding action. They assume that the longitudinal force controlled by the driver remains constant and they limit the lateral force to the remaining tyre capacity. As referenced, for a road vehicle, the linearized tyre model in (Qian, Fortelle et al. 2016) and (Funke, Brown et al. 2017)is like a linear tyre model. In addition, the tyre sideslip angle constraints maintain the tyres in their range of linear forces to keep the linear tyre pattern in place. Several studies utilise a vehicle bicycle model with linear tyre models and limit lateral and rear side tilt angles (Rasekhipour, Khajepour et al. 2017, Brüdigam, Ahmic et al. 2018, Joa, Yi et al. 2019).

2.4.2 Ethical Vehicle Design

The legal and moral consequences of self-driving car judgments in the event of unavoidable crashes were barely mentioned. Most of the moral machine research is centred on military applications or general machine intelligence (Purves, Jenkins et al. 2015, Kowalczuk and Czubenko 2017, Urooj, Feroz et al. 2018). Machine ethics is a relatively new topic of study

that focuses on the development of autonomous robots that can demonstrate ethical behaviour in novel settings.

2.4.2.1 Rational Approach

Engineers must guide the self-driving system explicitly as to how it should react in certain situations. This rationalist approach is frequently expressed as deontology, in which the system is required to follow rules, or consequentialism, in which the system's aim is to maximise utility. Engineers like these logical techniques because computers can readily obey rules and optimise functions. Unfortunately, as stated in sections 2.3.1 and 2.3.2, this method has certain drawbacks.

2.4.2.2 The Artificial Intelligence Approach

For years, the automatic translation of languages has been based on rules developed by experts. The expectation was that the language could be defined by rules, with enough time to learn the rules and to write them. An alternative approach using algorithms to automatically learn a language without formal rules is much more successful than rule-based methods. These techniques are called artificial intelligence (Terziyan, Gryshko et al. 2018, Xu, Dherbomez et al. 2018). Linguistic translation provides an adequate analogy for ethical systems. In both areas, artificial intelligence methods are useful if the rules cannot be expressed.

Artificial intelligence methods can learn human ethics by observing human actions or rewarding their own moral behaviour (Rouse 2017, Soin and Chahande 2017, Taramov and Shilov 2017). A computer can determine the components of ethics without a person having to explain exactly why an act is or is not ethical (Purves, Jenkins et al. 2015). (Van den Hoven, Lokhorst et al. 2012) describe these techniques as "bottom-up" approaches, which may include techniques such as genetic algorithms and learning algorithms (Damm 2012). In a simple case, artificial neural networks, which use node layers in a connection-oriented computational approach to find complex relationships between inputs and outputs, were used in a simple case to classify hypothetical decisions as either moral or amoral. Hibbard suggested a similar methodology for formulating a consequentialist approach to machine ethics in which an independent artificial intelligence agent calculates the moral weights attributed to humans after questioning subjects in different hypothetical situations (Hibbard 2012). An automated vehicle project, an autonomous Carnegie Mellon land vehicle in a neural network, used a simple

network of artificial neurons trained to teach driving by monitoring a human driver for only a few minutes (Applin and Fischer 2015). A similar technique could be used with much more training data to understand how people should behave morally or otherwise in a complex driving situation when time matters. The neural network could be trained on a combination of simulations and recordings of crashes and near-misses, with human feedback on the ethical response. Artificial intelligence techniques have several disadvantages. If they are not carefully designed, they may mimic humans' behaviour rather than what they believe. Self-protection instincts that do not maximise overall safety can be realistic but not ethical. Ethics is about how humans should or want to behave, not how they currently behave, and artificial intelligence techniques should capture ideal behaviour.

Another disadvantage of some artificial intelligence approaches is traceability. Artificial intelligence can be complex, and artificial neural networks cannot explain how a decision was made based on the input data in a comprehensible manner. There is already anecdotal evidence of computers that have discovered relationships that researchers do not understand (Carvalho, Lefévre et al. 2015, Cunningham, Galceran et al. 2015, McBride and Hoffman 2016). They proposed the choice of decision trees to encourage transparency and a different type of ethics (Asimov's laws can be formulated easily as a decision flowchart). However, the risk of alteration is important for the automation of road vehicles. Ethics would probably require that all humans be equal. However, a vehicle manufacturer is encouraged to build vehicles that primarily protect their own occupants. A built-in self-protection component of self-driving vehicle ethics could be hidden in a complex neural network and could only be discovered by analysing long-term accident trends. Safety precautions must be taken to prevent this from happening.

Although artificial intelligence approaches allow computers to learn human ethics without the difficult task of formulating ethics as a code (Garip, Karayel et al. 2017, Sarathy, Scheutz et al. 2017, Wintersberger, Frison et al. 2017), they lead to actions that are probably not justified. When formed with limited information, an artificial intelligence can learn completely involuntary, unwanted, unexpected, and undesirable behaviours. Without further testing, artificial intelligence reasoning methodologies for self-driving vehicles cannot be recommended without artificial rules designed to increase transparency and prevent unethical behaviour.

Chapter 3: Mechanic Formulations Governing the Response of Self-driving Vehicles

3.1 Introduction

This section describes the techniques used for the mechanics formulations guiding the behaviour of self-driving vehicles in the research to develop a framework for optimising ethical decision-making using constraining and cost-optimizing techniques by assigning various potential functions to ethical problems corresponding to their moral values. A linear bicycle vehicle model will be used to model the behaviour of the vehicle with a model-based design technique. To actualize the relationship between ethics and technology in self-driving vehicles and to assess the performance of trajectory planning systems in terms of traffic safety and compliance, obstacle avoidance, and longitudinal and lateral manoeuvrability, a large model capable of simulating a realistic driving scenario that includes a variety of factors and with many cars dynamically entering and leaving the merging zone is created. As shown in figure 3.1 below, these driving scenarios provide several options for technical decisions such as car types (properties, dynamics), driving strategies (aggressive, defensive) & algorithms, intersection geometry (weather, time, road conditions), and events that cars react to (sudden brake, change in steering, sensor failure).

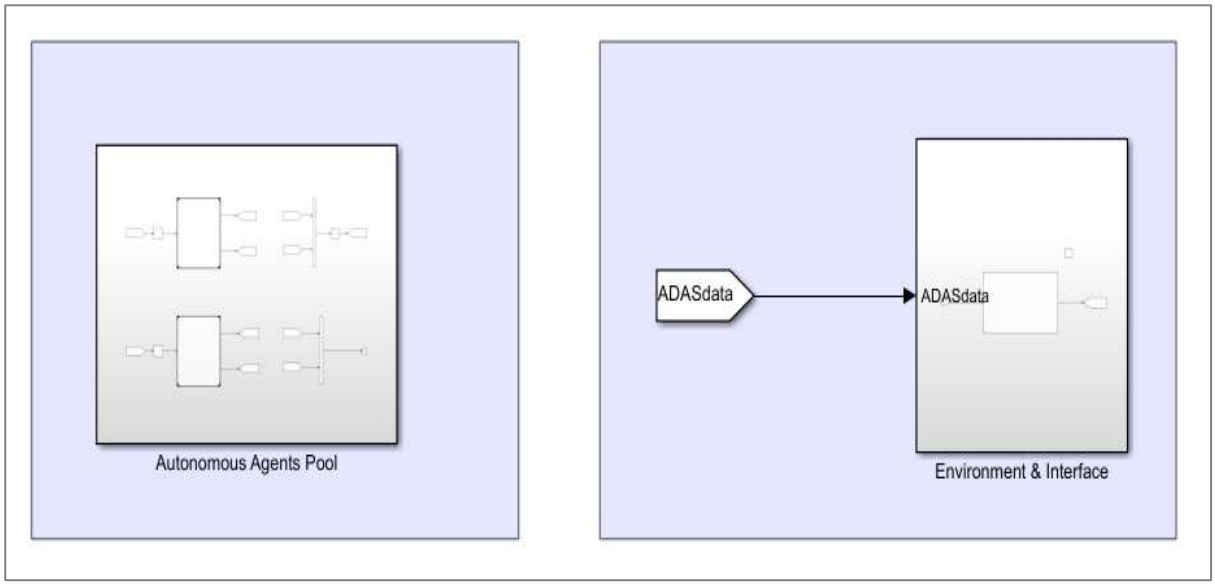

Figure 3.1: System Block Diagram

Autonomous agent pool

The agent pool has many vehicles of two different types, and each vehicle model has a driver strategy (planner) and vehicle dynamics (plant), which is a type of subsystem that repeats the execution of each element and concatenates the result. This makes it possible to simulate multiple vehicles of the same type with different behaviour at the same time. Inside the driver's behaviour, it checks whether it's safe to merge or switch and uses state-for-state parameters to calculate the longitudinal and lateral acceleration. These models of driver behaviour could easily be replaced with a different driver behaviour that has the correct output given the input.

Environment and Interface

The interface takes in either state of one of the outputs from the self-driven pool and fixes it into one of the chosen scenarios in the environment. These scenarios (single-lane curve, dual-lane merge, dual-lane switch, three-lane merge & switch, three-lane curve & switch, and four-lane merge & switch) are modelled in variance, which allows the system to make one block active at any given time during simulation. Each scenario has a sim-event road model subsystem where entities are generated, routed, and terminated and are connected to the driver strategy model of a self-driven agent pool through an interface that allows information flow between the subsystems. It's the glue that combines two domains together. For instance, a vehicle in any scenario is allowed to exit the system when they hit the crossing block, which shows that the position has hit a limit using information from the self-driven agent pool. In this case, the vehicle is removed from the simulation with the terminator block. Perception & Localization of each subsystem takes in either state from the model and maps it on to the road information. These subsystems must be updated whenever the model is updated with different map information or different road shapes. With this flexible framework, it's easier to model ethical issues and other types of agents and the environment in which they operate.

3.2 Code Generation

The codes were generated using MATLAB code-gen function from the toolbox for safety checks, lane merge and switch, lane curve and switch, and all the sensors for different driving scenarios. It was then integrated into the projects as a source code, for static and dynamic libraries as shown in Appendixes 1-17. The generated code is readable and portable with high efficiency and flexibility.

3.3 Model Predictive Control

Before the delays in the vehicle steering system are taken into account, it is compared to a model predictive control problem formulation without delay modelling, as shown by (Funke, Brown et al. 2015).

3.3.1 Formulation of the Problem

A four-state bicycle dynamic model with constant acceleration assumption is utilised in the baseline MPC formulation. Vehicle lateral velocity (U_y), yaw rate (r), heading deviation ($\Delta\psi$), and lateral deviation (e) are all included in the state vector (x):

$$x = [U_y \;\; r \;\; \Delta\psi \;\; e]^T \quad \text{..... Equation (3.1)}$$

The front steering angle (δ) computed from an affine vehicle model is the control input to the vehicle model (u):

$$x^{(k+1)} = A^{(k)}x^{(k)} + B^{(k)}u^{(k)} + c^{(k)} \quad k = 0, \ldots . n-1 \quad \text{..... Equation (3.2)}$$

for each time step in the prediction horizon (k) up to a finite number of time steps (n). The steering input in the original problem formulation by Funke et al. is front lateral tyre force, however in this study, steering angle (δ) is used, as in (Gray, Gao et al. 2013).
The relationship between front lateral tyre force (F_y) and steering angle (δ) is nonlinear. According to (Joa, Yi et al. 2019) depicts the relationship as follows:

$$F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3\mu F_z}|\tan\alpha|\tan\alpha \\ -\frac{C_\alpha^3}{27\ {}^2F_z^2}\tan^3\alpha \\ -\mu F_z \sin\alpha \end{cases} \quad |\alpha| < \tan^{-1}\left(\frac{3\mu F_z}{C_\alpha}\right) \quad \text{..... Equation (3.3)}$$

where slip angle (α) is the angle between the tyre heading and the tire's velocity vector, c_α is the tire's cornering stiffness, μ is the coefficient of friction, and F_z is the tire's normal load. Using tiny angles and the vehicle model in Appendix - 18, the front slip angle (α_f) for the front tyres may be stated as follows:

$$\alpha_f = tan^{-1}\left(\frac{U_y+ar}{U_x}\right) - \delta \approx \frac{U_y+ar}{U_x} - \delta \quad \ldots\ldots\ldots\ldots\ldots \text{Equation (3.4)}$$

Erlien, Funke, and Gerdes used an affine, time-varying model with consecutive linearization points to estimate the tyre curve for rear tyres to capture realistic behaviour while retaining the bicycle model's convexity. For the front tyres, a similar approach may be used:

$$F_{yf} = \frac{\delta F_{yf}}{\delta \alpha_f}\Big|_{\alpha_{f,0}} \left(\alpha_f - \alpha_{f,0}\right) + F_{yf}\left(\alpha_{f,0}\right) \ldots\ldots\ldots \text{Equation (3.5)}$$

This is essentially a Taylor expansion around an operational point ($\alpha_{f,0}$), with $\alpha_{f,0}$ determined from the preceding optimization's solution at each time step on the prediction horizon. The system is described by linear differential equations as in (3.4) and steer angle as the controller input.

A nonzero diagonal entry in the weighting matrix (Q) is associated with lateral deviation and heading deviation as these states are specified relative to a nominal or desired path. The following is the entire optimization problem:

$$\text{Minimize (u): } \sum_{k=0}^{n} v^{(k)T} R^{(k)} x^{(k)} + \sum_{k=1}^{n} x^{(k)T} Q^{(k)} x^{(k)} \quad \ldots\ldots\ldots\ldots \text{Equation (3.6a)}$$

$$\text{Subject to: } x^{(k+1)} = A^{(k)} x^{(k)} + B^{(k)} u^{(k)} + C^{(k)} \quad \ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.6b)}$$

$$\left|u^{(k)}\right| \leq u_{max}^{(k)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.6c)}$$

$$\left|v^{(k)}\right| \leq v_{max}^{(k)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.6d)}$$

where $v^{(k)} = u^{(k)} - u^{(k-1)}$ is the change in front steer angle, weighting matrix (R) penalizes changes in steer angle, and $u_{max}^{(k)}$ and $v_{max}^{(k)}$ are physical limits in the steering system.

The optimization problem (3.6) is a quadratic programme with a sparse structure that can be solved in real time using an efficient solver. CVXGEN, created by (Mattingley, Boyd et al. 2012) is utilised to solve for the input vector, $u = [u(0) \ldots \ldots u(n)]$ but only the first solution in the vector $(u(0))$ controls the steering system. The optimization task is executed at 100 cycles per second on a single core of a ruggedized computer with an i9 CPU. To see if delay

compensation was essential, the performance of problem formulation (3.6) was compared to a baseline to validate all the problem formulations in the chapter.

Despite the steering system's delay, the self-driving vehicle effectively navigates around the obstacle, as seen in figure 3.2 (top) plot. The command and measurement of the hand wheel angle while the self-driving system performs the manoeuvre are shown in the centre plot in figure 3.2 at time t = 1.63s, it also depicts an open-loop prediction horizon for the steering command (scaled by steering ratio). The best solution does not anticipate future steering behaviour well at the end of the prediction horizon now, suggesting a model mismatch. Along the prediction horizon, the root mean squared (RMS) error between the closed-loop command and the open-loop forecast is 16.13, indicating that the open-loop forecast accurately predicts future behaviour. The vehicle yaw rate has substantial oscillations as shown in figure 3.2 (bottom), indicating that the vehicle did not spin properly and had to correct its rotation during the manoeuvre. This happens because the delay prevents the proper amount of steer angle from being command when it's needed, necessitating more aggressive manoeuvring to avoid the disturbance occurring at a time step later than the predictive controller predicted. The yaw rate has an RMS error of 0.0638 rad/s.

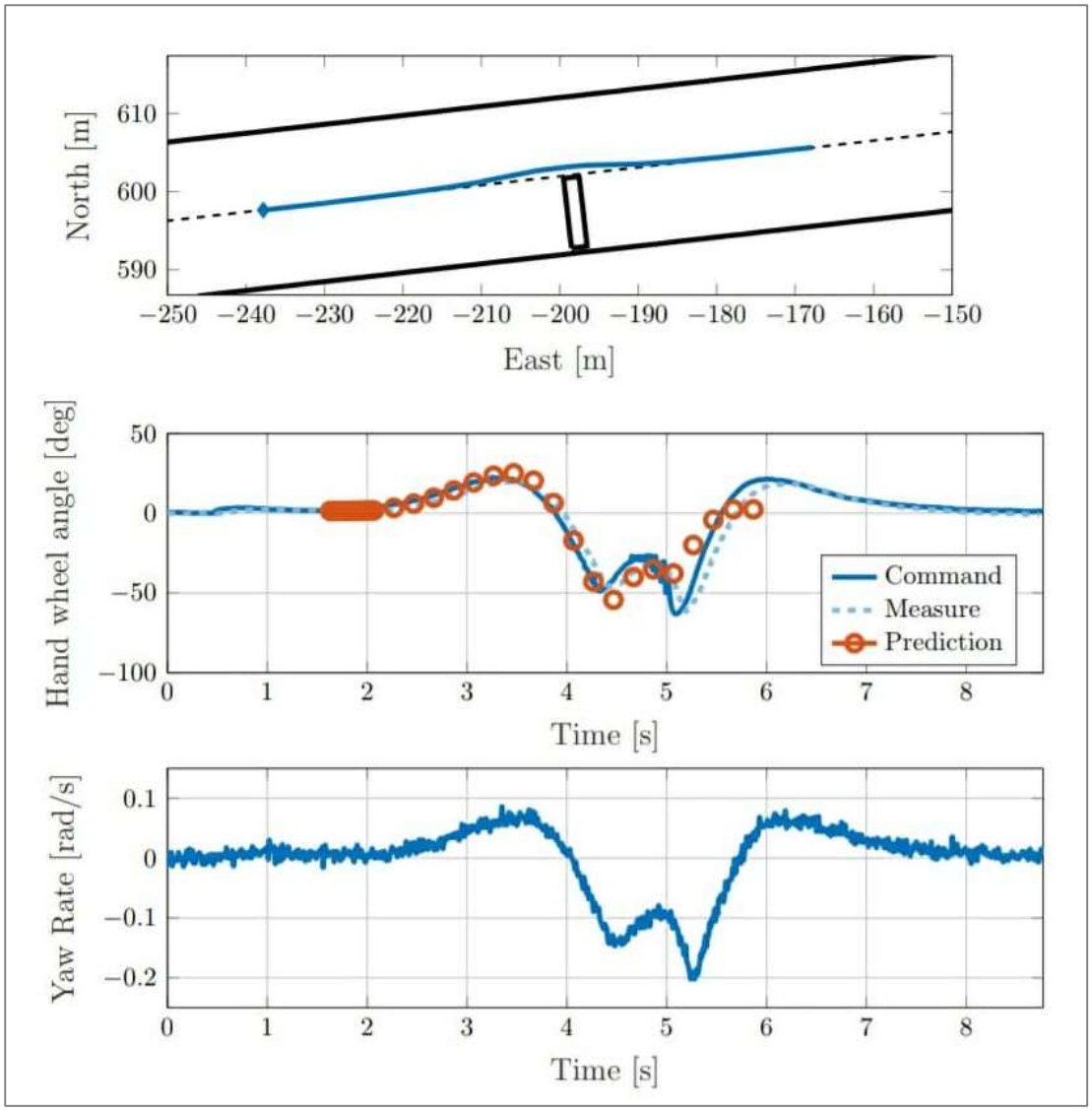

Figure 3.2: Baseline trajectory overhead, diamond indicates start of manoeuvre at the top, command and measuring hand wheel angle with open-loop prediction at one time step in the middle, and yaw rate at the bottom due to lateral perturbation from the nominal path.

3.4 Pure Delay Modelling

The steering system's delay should be accounted for, based on the outcomes of the preceding sections. The MPC problem formulation in this part simply includes the pure time delay.

3.4.1 Problem Formulation

The actuated steering command (u_{actual}) follows the commanded steering command (u) by a time delay as shown in equation (3.7) because of which the modelled state transition connection is formed as in equation (3.8).

$$u_{actual}(t) = u\left(t - T_{delay}\right) \quad \text{Equation (3.7)}$$

$$x\left(t + T_{pred}\right) = A(t)x(t) + B(t)u\left(t - T_{delay}\right) \quad \text{Equation (3.8)}$$

T_{pred} is the discretization time into the prediction horizon, where T_{delay} is the pure delay time and is known to be around $40ms$, thus the problem formulation (3.6) is changed to account for that. For the small-time steps of the prediction horizon, $T_{delay} = T_{Pred}$, such that equation (3.8) may be discretized as

$$x^{(k+1)} = A^{(k)}x^{(k)} + B^{(k)}u^{(k-1)} + C^{(k)}, \quad k = 0, \dots n-1 \quad \text{Equation (3.9)}$$

The optimal steering input calculated for T_{pred} seconds previously determines the transition from x (0) to x (1). When $T_{delay} = T_{pred}$ is taken into consideration, the optimization problem may be stated as in equation (3.10)

$$\text{Minimize (u):} \sum_{i=0}^{n} v^{(k)T} R^{(k)} x^{(k)} + \sum_{i=0}^{n} x^{(k)T} Q^{(k)} x^{(k)} \quad \text{Equation (3.10a)}$$

$$\text{Subject to: } x^{(1)} = A^{(0)}x^{(0)} + B^{(k)}z^{-d}u^{*(0)} + C^{(0)} \quad \text{Equation (3.10b)}$$

$$x^{(k+1)} = A^{(k)}x^{(k)} + B^{(k)}u^{(k-1)} + C^{(k)} \quad \text{Equation (3.10c)}$$

$$\left|u^{(k)}\right| \leq u_{max}^{(k)} \quad \text{Equation (3.10d)}$$

$$\left|v^{(k)}\right| \leq v_{max}^{(k)} \quad \text{Equation (3.10e)}$$

The zeroth item of the optimal solution vector determined d control iterations preceding system is $z^{-d}u^{*(0)}$. This solution should be identical to the T_{delay} seconds preceding solution. If the controller runs every $T_{control}$ seconds,

$$d = round\left(\frac{Tdelay}{Tcontrol}\right) \quad \text{Equation (3.11)}$$

Because of the delay formulation that leverages the prior control input in seeding equation (3.10b), the decision variable vector (u) has been decreased from size $(n+1)$ to size (n).

3.4.2 Simulation Results

The open-loop prediction improves by just considering the pure time delay in the steering system, as seen in the centre plot of figure 3.3. This method reduces the hand wheel angle RMS throughout the prediction horizon by approximately by a factor of two to 7.87 at time $t = 1.63s$. This manoeuvre's yaw rate has a decreased RMS error of $0.0574\ rad/s$, as seen in the bottom figure 3.3. The pure time delay problem formulation produces a trajectory that is comparable to the baseline trajectory.

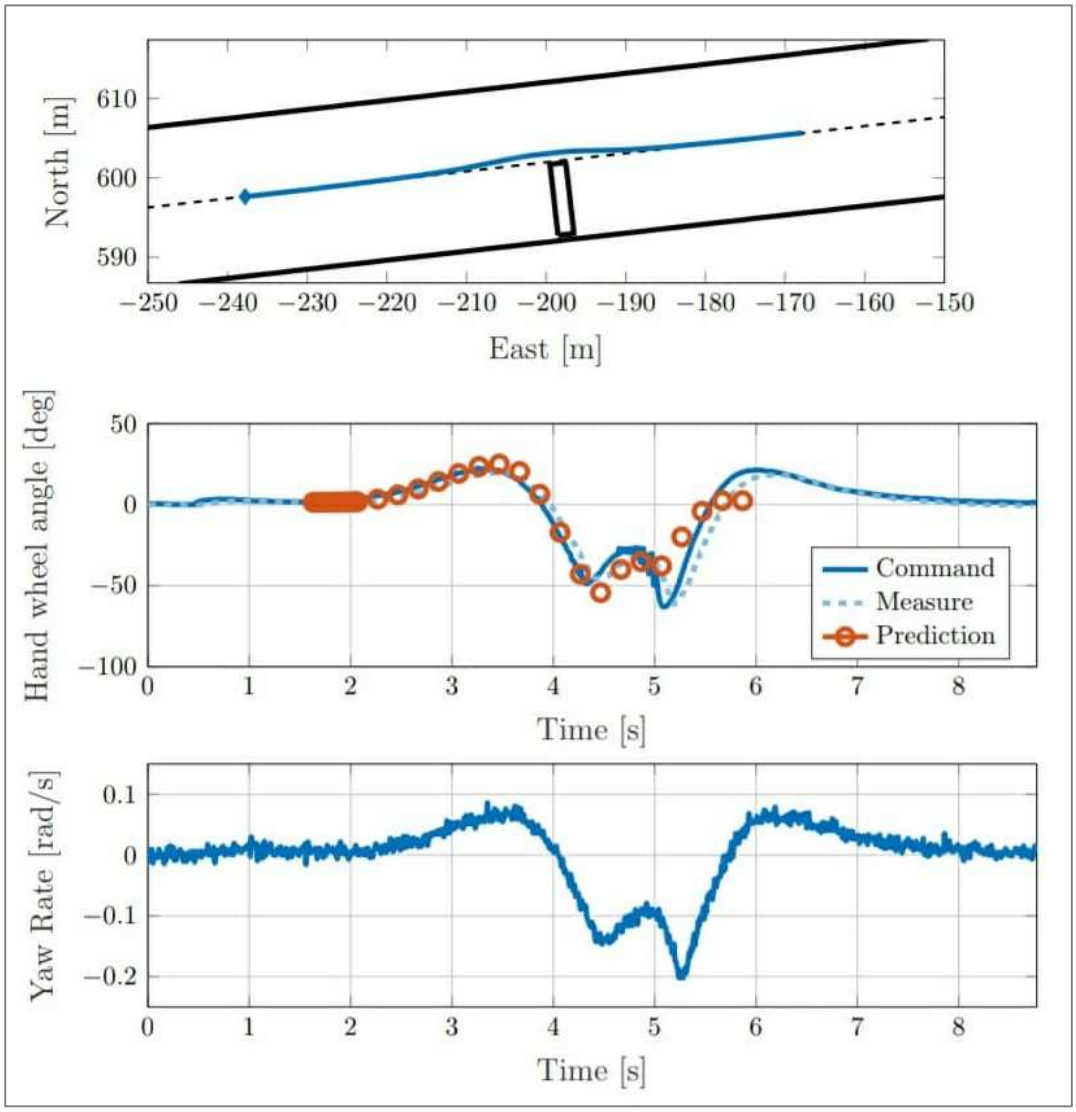

Figure 3.3: Pure time delay trajectory overhead, diamond indicates start of manoeuvre at the top, command and measuring hand wheel angle with open-loop prediction at one time step in the middle, and yaw rate at the bottom due to lateral perturbation from the nominal path.

3.5 Dynamic Lag Modelling

The dynamic lag in the steering system, is included into the MPC problem formulation to improve the open-loop prediction even more. A first-order lag model with the pure time delay, a first-order lag model tuned to lump in the pure time delay, and a second-order lag model adjusted to lump in the pure time delay are all compared in the next section.

3.5.1 First-Order Lag

The state vector is appended with a fifth state f_{ol} to represent the steering input to the affine vehicle model after a first-order delay to accommodate for first-order dynamics. The new state vector is transformed into

$$x = [U_y \quad r \quad \Delta\psi \quad e \quad \delta_{fol}]^T \quad \text{Equation (3.12)}$$

The affine vehicle model is also supplemented by δ_{fol}, which is written as

$$\dot{\delta_{fol}}(t) = -\frac{1}{\tau}\delta_{fol}(t) + \frac{1}{\tau}u(t) \quad \text{Equation (3.13)}$$

where (τ) is the time constant. Both problem formulations (3.6) and (3.10) employ the five-state vector, but with different values for the time constant. The step responses are as shown in figure 3.4

3.5.2 Second Order Lag

The steering delay is approximated as a second-order system, because there is a pure time delay with apparently first-order dynamics. In the same way as the first-order equation,

$$x = [U_y \quad r \quad \Delta\psi \quad e \quad \delta_{sol} \quad \dot{\delta_{sol}}]^T \quad \text{Equation (3.14)}$$

The new states are linked to the actual steering input in the same way as the previous states were.

$$\ddot{\delta_{sol}}(t) = -\omega_n^2\delta_{sol}(t) - 2\zeta\omega_n\dot{\delta_{sol}}(t) - \omega_n^2 u(t) \quad \text{Equation (3.15)}$$

This delay model is adjusted to have the step response shown in figure 3.4 since it is exclusively used for problem formulation (3.6).

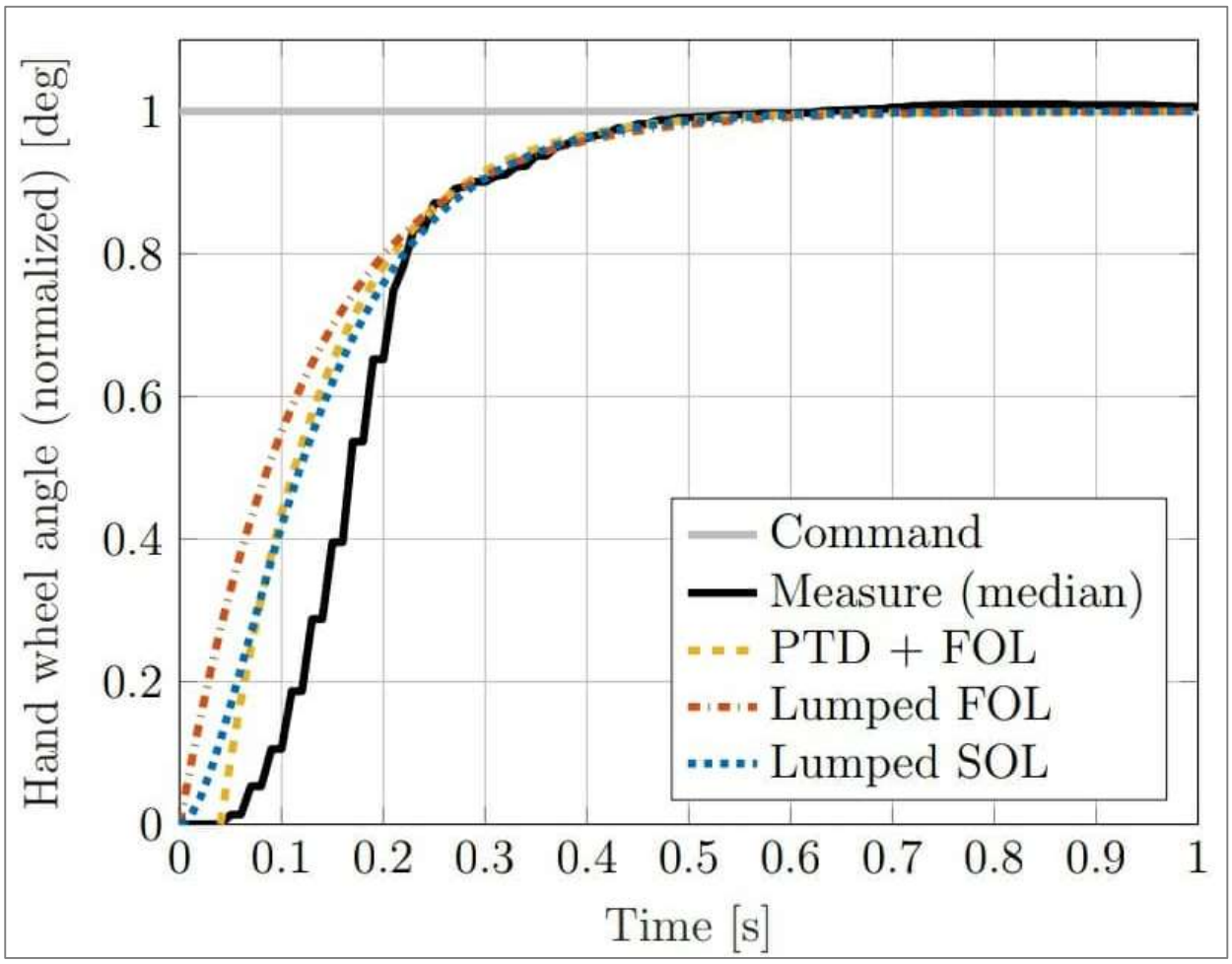

Figure 3.4: First- and Second-order models on median step response data.

3.5.3 Simulation Results

Because the steering system's real dynamics are unknown, several simulations were conducted to see which of the problem formulations allowing for dynamic lag worked as illustrated in figures 3.5, 3.6, and 3.7. of the RMS error for the hand wheel angle over the prediction horizon at $t = 1.63s$, as well as the RMS and maximum absolute yaw rate. Because of its connection to lateral acceleration in the inertial frame, the maximum absolute yaw rate is as shown in table 3.1 below

$$\alpha_y = \dot{U} + rU_x \quad \text{Equation (3.16)}$$

Table 3.1: RMS of hand wheel angle (HWA) along prediction horizon, RMS of yaw rate, and maximum absolute yaw rate

Simulation	Prediction horizon HWA RMS^{+} (^{0})	Yaw rate RMS (rad/s)	Max yaw rate* (rad/s)
Baseline	16.13	0.0638	0.2259
Pure time delay	7.87	0.0574	0.2024
PTD + FOL	7.76	0.0536	0.1895
Lumped + FOL	8.16	0.0555	0.1678
Lumped + SOL	8.38	0.0562	0.1665

$^{+}$@ t = 1.63s *absolute value

In comparison to the rU_x term, the vehicle frame's lateral acceleration (V_x) is modest, and longitudinal velocity is stable throughout all trials. As a result, the maximum absolute yaw rate indicates how much lateral acceleration car passengers are exposed to throughout each manoeuvre.

Because the corresponding models accurately capture the steering actuation, the open-loop prediction horizons in the central plots of figures (3.5), (3.6), and (3.7) appear qualitatively comparable. The measured response is smooth, the yaw rate response also reflects the overall smoothness of the manoeuvre. Compared to the baseline and pure time delay implementations, the RMS error and maximum of the yaw rate continue to improve quantitatively.

A thorough investigation of the figures in table 3.1 reveals that there is a trade-off to be made when choosing the “best” delay modelling problem formulation to employ. The aggregated first order and second-order RMS errors, as well as the maximum absolute yaw rate, have minimal difference. The lumped second-order method offers a somewhat smoother yaw rate response at the cost of a wider state space in terms of computation. If the pure time delay is to be clearly simulated, extra information in-vehicle is required, such as the number of control time steps required to delay the first input to the optimization problem. Because of the comparable speed, smaller state space, and easier implementation, I decided to go with the lumped first-order method.

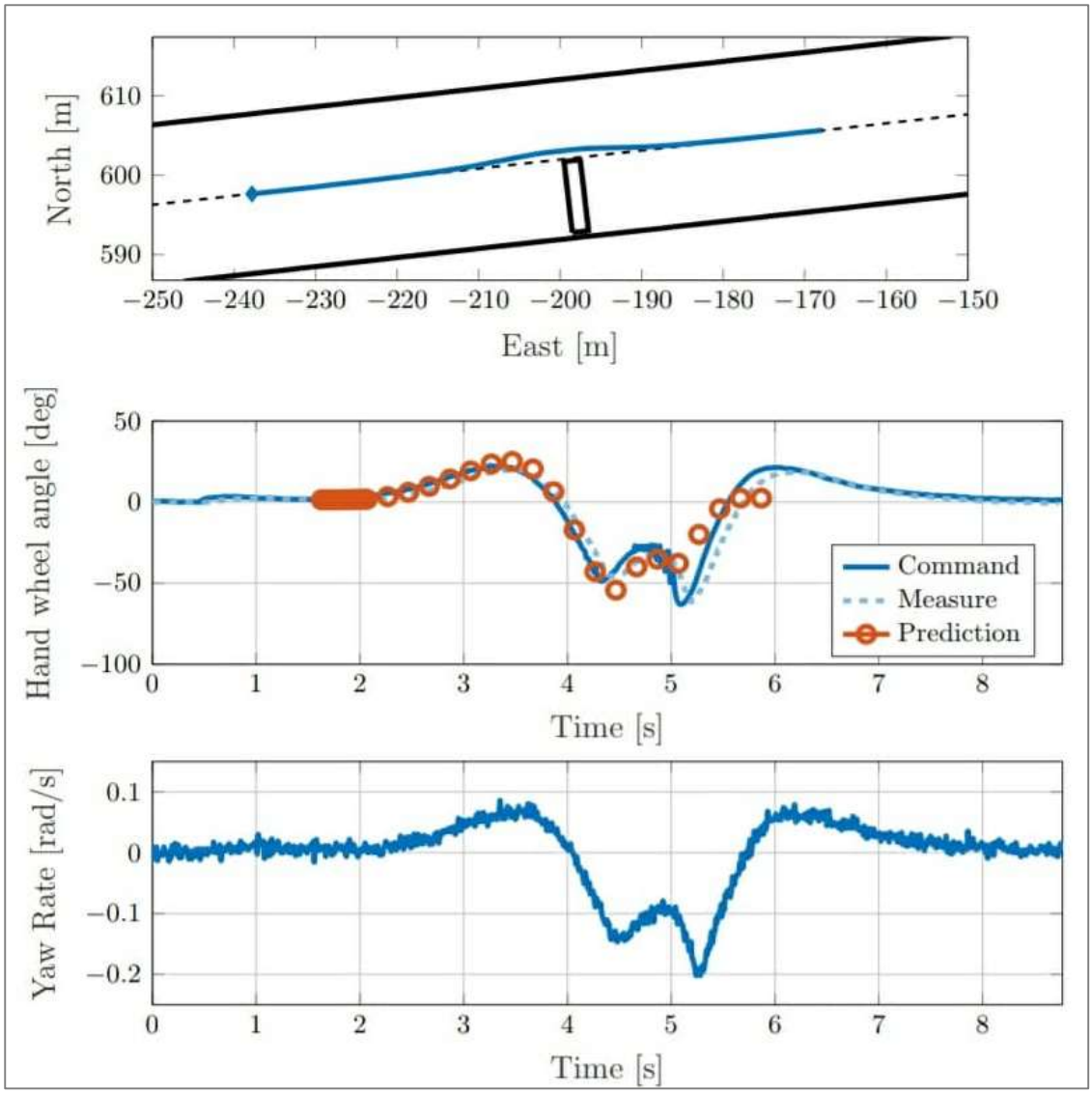

Figure 3.5: Pure time delay with first-order dynamics trajectory overhead, diamond indicates start of manoeuvre at the top, command and measuring hand wheel angle with open-loop prediction at one time step in the middle, and yaw rate at the bottom due to lateral perturbation from the nominal path.

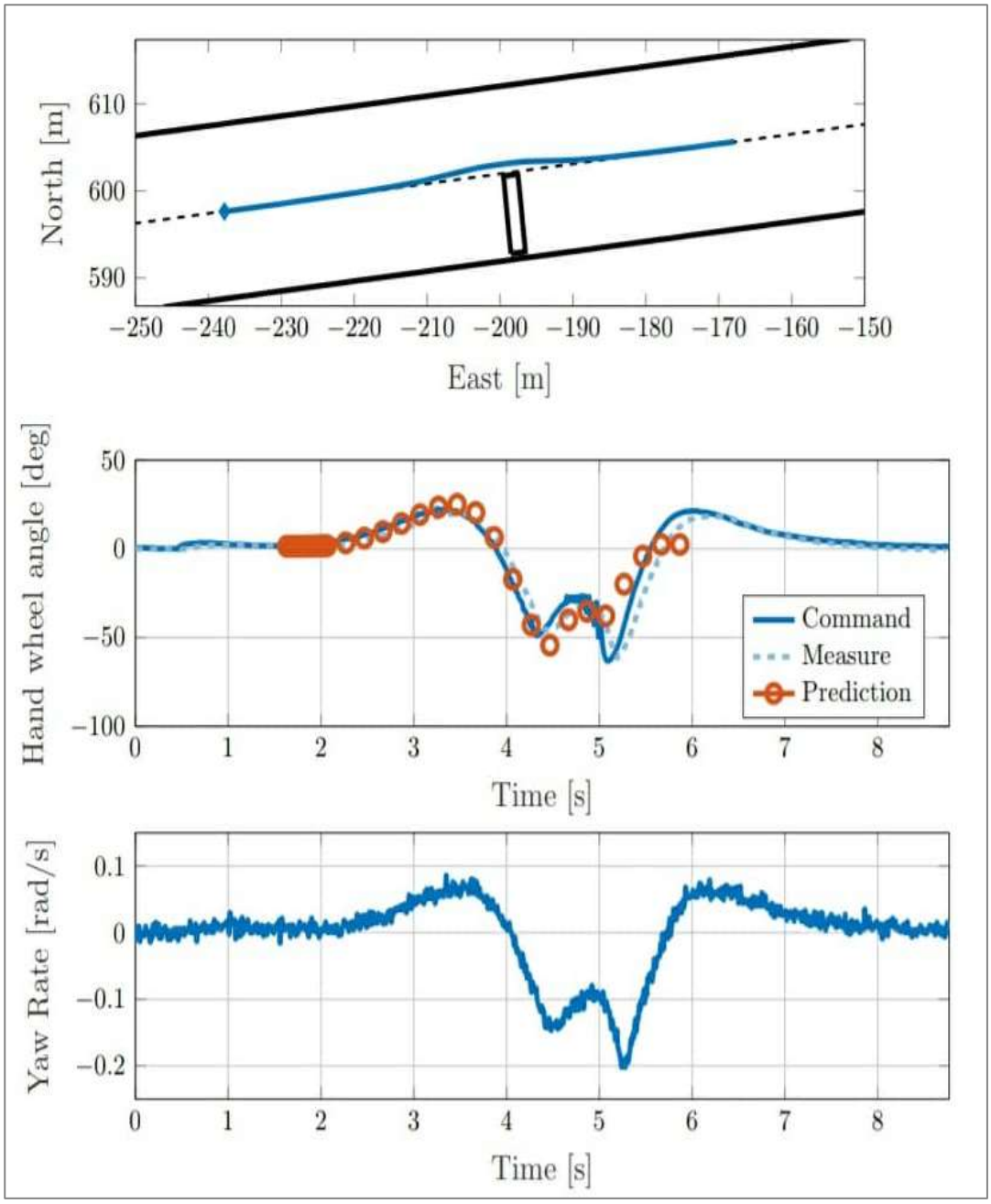

Figure 3.6: Lumped first-order lag trajectory overhead, diamond indicates start of manoeuvre at the top, command and measuring hand wheel angle with open-loop prediction at one time step in the middle, and yaw rate at the bottom due to lateral perturbation from the nominal path.

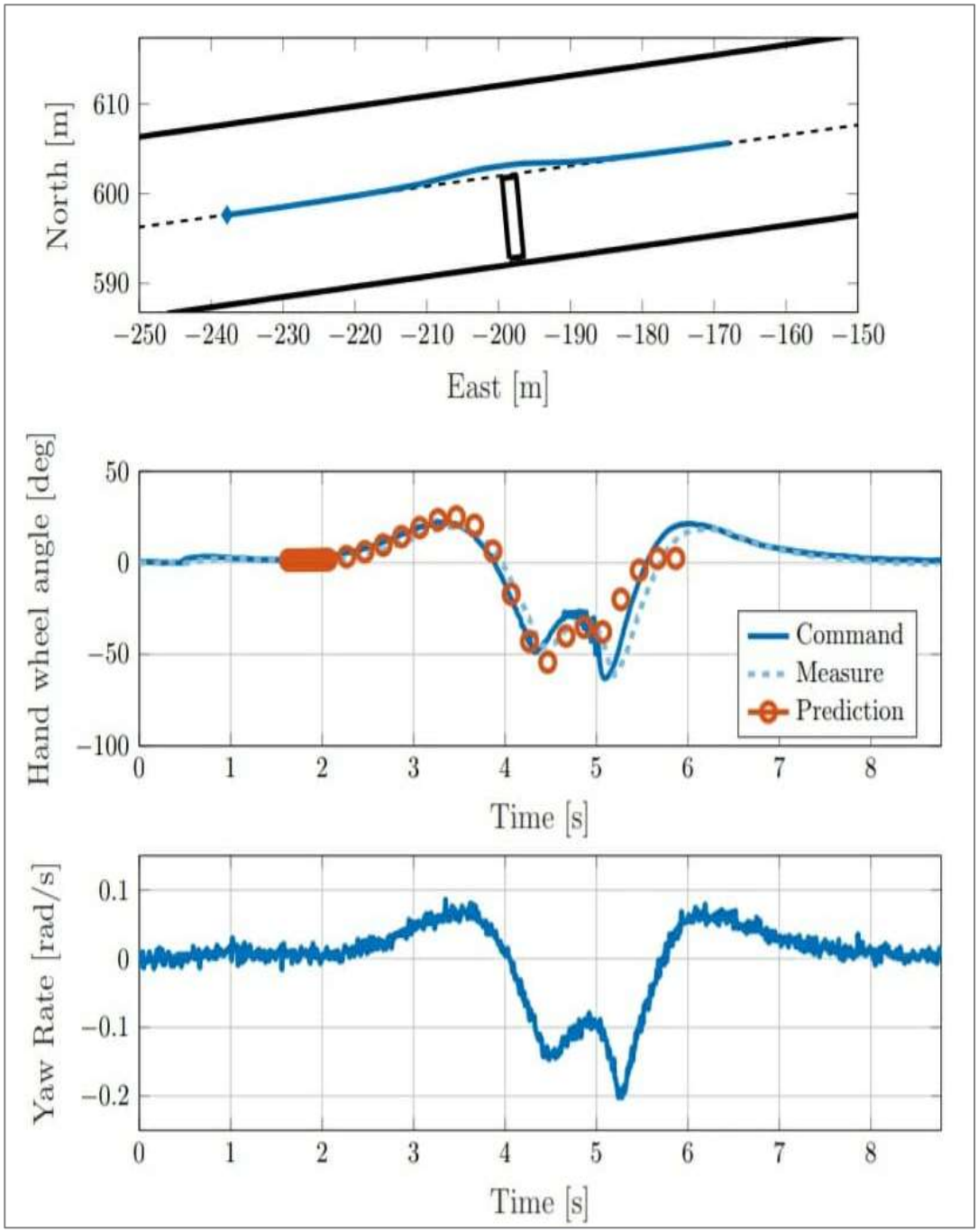

Figure 3.7:Lumped Second-order lag trajectory overhead, diamond indicates start of manoeuvre at the top, command and measuring hand wheel angle with open-loop prediction at one time step in the middle, and yaw rate at the bottom due to lateral perturbation from the nominal path.

3.6 New Optimization Framework Formulation

The given vehicle dynamics model and restrictions are used to build an optimization of ethical decision-making control in this part. The controller's goal function now includes the possible scope, road rules, obstacle avoidance, traffic regulation compliance, ethical consideration, ethical limitations, and related expenses. Based on the projected values, the new framework would be able to forecast the vehicle's reaction to a certain horizon and optimise vehicle dynamics, command tracking, obstacle avoidance, road laws, and ethical reasoning up to that horizon. The ethical restrictions were implemented as soft constraints that may be readily broken but are sanctioned when they are. To allow for certain violations, a slack variable was introduced to the constraint equation, as well as a penalty term in the objective function to punish the violation. The tyre force limits are supple. This is since the restrictions are simulations of the actual tyre limitations, which may differ from the actual limitations. Furthermore, the predicted state inaccuracies may result in constraint violations. To prevent infeasibility owing to constraint violation, the tyre restrictions are treated as soft constraints.

3.6.1 Scenario Creation

A basic, realistic driving scenario involving several elements, including accident avoidance, mobility concerns, and traffic rules is built to contextualise the link between ethics and engineering in self-driving vehicles. As in the case of a self-driven car moving at a steady speed down a two-lane highway as shown in figure 3.8 below. An obstruction in front of the car is blocking the present lane of the ego vehicle. This straightforward concept raises a slew of engineering considerations. Section 3.6.3 explains how the different elements from this scenario (collision avoidance, mobility, traffic regulations, and speed) are included into an MPC model.

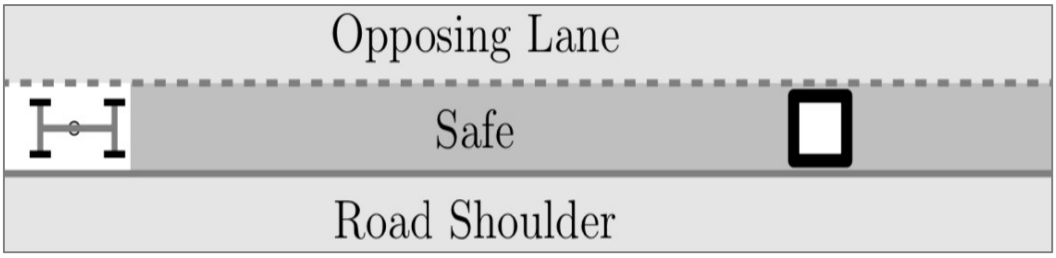

Figure 3.8: The shaded areas denote driving zones. The most secure sector is the vehicle's present lane, free of obstructions. As the car leaves the lane, the safety of the driving zone decreases.

Programming the vehicle's ability to continue going is one engineering design possibility. This would include moving into the opposite lane or onto the road shoulder to avoid the obstruction and continue their journey. If the car decides to join the opposite lane, the vehicle may be in violation of a traffic rule for a limited period. A double yellow line, for example, might be used as a lane divider. If the vehicle moves to the road shoulder to avoid the obstacle, it complies with the double yellow line traffic regulation and continues to travel; nevertheless, the road shoulder must be available and safe, and it is not intended for normal driving. When weighing these choices, it's clear that the contending demands of mobility, safety, and legality must be balanced.

Another option, if mobility is a concern, is to set the self-driven car to closely adhere to traffic regulations. The fundamental principle is that traffic laws are strictly. However, if the vehicle is caught between the twin aims of avoiding the obstruction and following the double yellow line, it may stop and remain stopped indefinitely. This action may have an adverse effect on the mobility and safety of nearby cars.

The possibility of engineering choices must be established not only in terms of the type of action to be taken, but also in terms of the degree of action to be taken. The amount of space between the vehicle and the obstacle, for example, is a design issue while crossing the double yellow line to manoeuvre around the obstruction and facilitate smooth traffic movement. If incoming traffic emerges, a small distance between the vehicle and the obstacle allows the vehicle to keep closer to its original allocated lane, but it increases the danger of brushing against the obstruction's side. A broader berth guarantees that the vehicle passes without colliding with the barrier, but it also positions the car further into the opposite lane, making it take longer to return to its original allocated lane. Another technical issue that demands significant ethical thought is the degree to which a vehicle is tuned to break a traffic rule. A layer of engineering design considerations, in addition to the kind and degree of action performed, incorporates the fact that various types of vehicles may be allocated varying traffic law permits depending on their projected function in society. This is shown subsequently with the example of an ambulance and a taxi, both of which transport passengers yet have significantly distinct road behaviour due to their respective missions.

Deconstructing this basic and frequent driving situation demonstrates the wide range of vehicle behaviours that may be induced by various engineering design decisions, as well as the need of making those design decisions in a logical, rational, and unassailable manner. Not only engineers, but also other road users who will use the roads with autonomous cars, as well as regulators who oversee traffic safety, should be able to understand the judgments. Engineers

can use a reasoning tool to evaluate the ethical consequences of their engineering design decisions if they comprehend philosophical frameworks in the context of engineering.

3.6.2 Design Alternatives

Deontology and consequentialism are philosophical systems that have been the subject of significant investigation. These frameworks are utilised as reasoning tools to rationalise and explain design decisions in the programming of an autonomous vehicle because of their formative presence in philosophy. As previously discussed, there are several methods for programming an autonomous vehicle. Because the explicit consideration of constraints and costs in MPC translates well to the ideas of deontology and consequentialism, an MPC version of the problem is used in this chapter. The vehicle's actions are constrained by deontological constraints, and consequentialism is accomplished via the cost function, according to this philosophical argument. As a result, philosophical frameworks are applied to the design of this restricted optimization problem.

The sections that follow show how using these two philosophical frameworks to address the problem leads to a logical investigation of the problem's many aims. Constraints are placed on the vehicle to ensure that it avoids collisions, follows dynamical equations, and steer within its capabilities. The cost function's goal is to get the vehicle to follow a predetermined course while maintaining adequate occupant comfort. In contrast to these other goals, it is less apparent if traffic regulations constitute a cost or a constraint, thus other representations are being investigated.

3.6.2.1 Path Tracking

Following a predetermined path is a fundamental goal of an autonomous vehicle. This goal implies that the reference path is provided by a higher-level planner and is not guaranteed to be free of obstacles. Because following a path is a physical condition based on a measure of position differences, path tracking might be ensured using a constraint derived from deontological reasoning. This might be in the form of a constraint that the vehicle's location on the path must be equal to the intended position. Following the path is not a rigorous necessity for preserving safety, according to additional investigation; if an impediment occurs on the path, the vehicle should have the choice to divert. Considering this logic, a cost function may be used to achieve the aim of path tracking through optimization, as shown in figure 3.9. As a

result of using a consequentialist framework for path tracking, the vehicle is given the flexibility to deviate; if path tracking were indicated as a rule in a deontological framework, rule conflict and problem feasibility would present a safety concern. For the specific purpose of path tracking, the more flexible principles of consequentialism are used instead of the core notion of deontology, which is that rules must be followed without exception.

Path tracking's goal is converted into a mathematical framework from a consequentialist perspective. The vehicle must minimise lateral deviation from the path (e) and heading error ($\Delta\psi$) to follow the path using a cost function as in equation (3.17)

$$J_x = \sum_{i=0}^{n} x^{(k)T} Q^{(k)} x^{(k)} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{ Equation (3.17)}$$

Where x is the vehicle state vector encompassing e and $\Delta\psi$, k is the discrete time step in the prediction horizon, and the weight matrix (Q) only contains diagonal, non-zero entries corresponding to e and $\Delta\psi$ (explained in more detail in section 3.6.3.1 and Appendix -19).

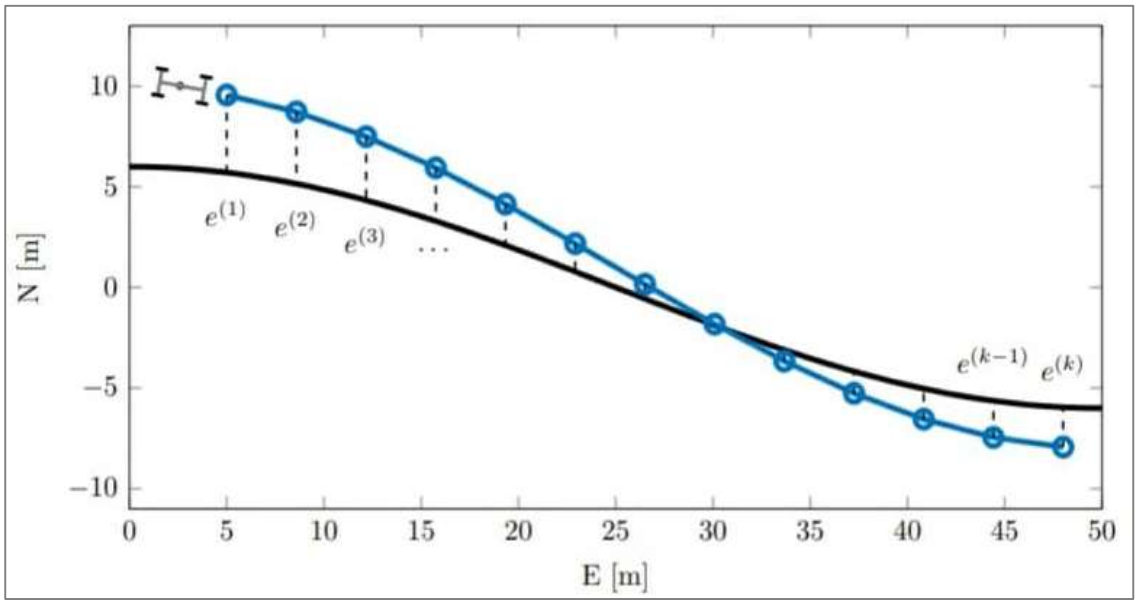

Figure 3.9: Calculating the cost based on the difference between the desired path (black) and the vehicle's actual path (blue with dots).

3.6.2.2 Steering Control

There are several designs aims for vehicle steering. As part of mobility, the steering must function within the actuator's limitations, contribute to path tracking and obstacle avoidance, and be smooth. The first of these objectives, keeping the actuator within its limitations, may be expressed as a maximum slew rate constraint. Because this cap represents a physical restriction on an actuator, it was chosen for this design. Physical limitations are imposed as constraints in the deontological sense of rigorous rule compliance because they must be highly prioritised in the control system. It is simply not possible to implement a solution that necessitates control inputs that exceed physical constraints. As a result, the most acceptable classification for the slew rate limit is deontological. Furthermore, accepting the vehicle's physical boundaries is comparable to acknowledging natural laws, which can serve as guiding principles. The following is a mathematical representation of this limit:

$$\left|F_{yf}^{(k)} - F_{yf}^{(k-1)}\right| \leq F_{yf}\, max\ slew \quad \ldots\ldots\ldots\ldots\ldots \text{Equation (3.18)}$$

$F_{yf,maximum\ slew}$ is the maximum slew rate of the steering system, and F_{yf}, is the lateral front tyre force.

Additional design goals for steering emerge when the limitation of maximum slew rate is followed. The steering smoothness, which is impacted by the change in input from time step to time step, is one of the most important goals. Because most riders anticipate a level of comfort when riding in a vehicle, steering smoothness is a utilitarian criterion to incorporate in the control algorithm. Occupant comfort is a desired feature, but like path tracking, might result in safety trade-offs if encoded as a hard and fast rule of matching a certain rate via an equality condition or remaining under a rate via an inequality constraint. If the vehicle must swerve rapidly to avoid an obstruction, it will be restricted by the need to maintain smooth steering and may not be able to steer and manoeuvre quickly enough to prevent a collision. A cost related with steering smoothness that the algorithm will decrease can, however, be included in the cost function when evaluated from a consequentialist perspective. If smoothness is subordinate to more highly valued criteria related to safety, I prefer to account for it in the cost function. As a result of the cost-benefit analysis, steering smoothness for occupant comfort is viewed as a cost. By linking a cost with the change in steering, or more accurately, the lateral front tyre force, the occupant comfort level is included into the objective function as in eq. (3.19)

$$J_{fyf} = R\sum_k \left\| F_{yf}^{(k)} - F_{yf}^{(k-1)} \right\|_2^2 \quad \text{Equation (3.19)}$$

where R stands for the associated cost. The differential in front steering angle is reduced by limiting the lateral front tyre force difference over the prediction horizon. As a result of this term in the cost function, smooth steering implementation is achieved, which might affect occupant comfort.

3.6.2.3 Obstacle Avoidance

When it comes to driving highways, avoiding obstacles is a top consideration. As discussed in the previous sections, the potential of collisions and the need to preserve the capacity to avoid them is the reason for choosing consequentialist costs over deontological principles for path tracking and steering smoothness. Because collision avoidance is perhaps the most important feature of a self-driven car, therefore I decided to look at it through the lens of deontology and use it as a constraint.

The deontological principles that control obstacle avoidance are derived from separating the environment into tubes through which the vehicle may safely pass and defining the envelope in which each tube lies. The self-driving vehicle can select from one of three options in the previous scenario illustrated in figure 3.8: pass the vehicle by joining the lane on the left, pass the vehicle by going onto the right shoulder, or stay in the lane. The limits of each tube in the environment may be built using a nominal path, which is essentially the centreline of the lane in which the vehicle travels in the situation described here. These envelopes are defined by a series of time-varying constraints on the maximum and lowest lateral offset from the nominal path (e) required to stay in the tube. To guarantee the trajectory is collision-free, the vehicle's trajectory across the prediction horizon is limited to stay within this envelope. Because the nominal path does not have to be obstacle-free, the vehicle's environmental envelope may force it to deviate from it.

Based on the constant longitudinal vehicle speed (U_x) and the assumption that the distance along the path is exclusively a function of U_x, the environment is sampled at discrete positions along the nominal path. The vehicle's projected position as it approaches the prediction horizon is shown in figure 3.10b. The objects are expanded to fit the sample, as illustrated in figure 3.10c, to correspond with discrete sampling. This expansion identifies possible gaps (defined as lengths bigger than the width of a car) between items. A graph search method creates tubes like the ones illustrated in figure 3.10d by connecting neighbouring viable gaps. For the vehicle

to avoid accidents, one tube must include the whole prediction horizon. This tube approach resembles LaValle's vertical cell decomposition (LaValle 2006), Suh and Bishop's tube feasibility for robotic arm motion planning (Suh and Bishop 1988), and Ziegler, Bender, Dang, and Stiller's driving corridors (Ziegler, Bender et al. 2014).

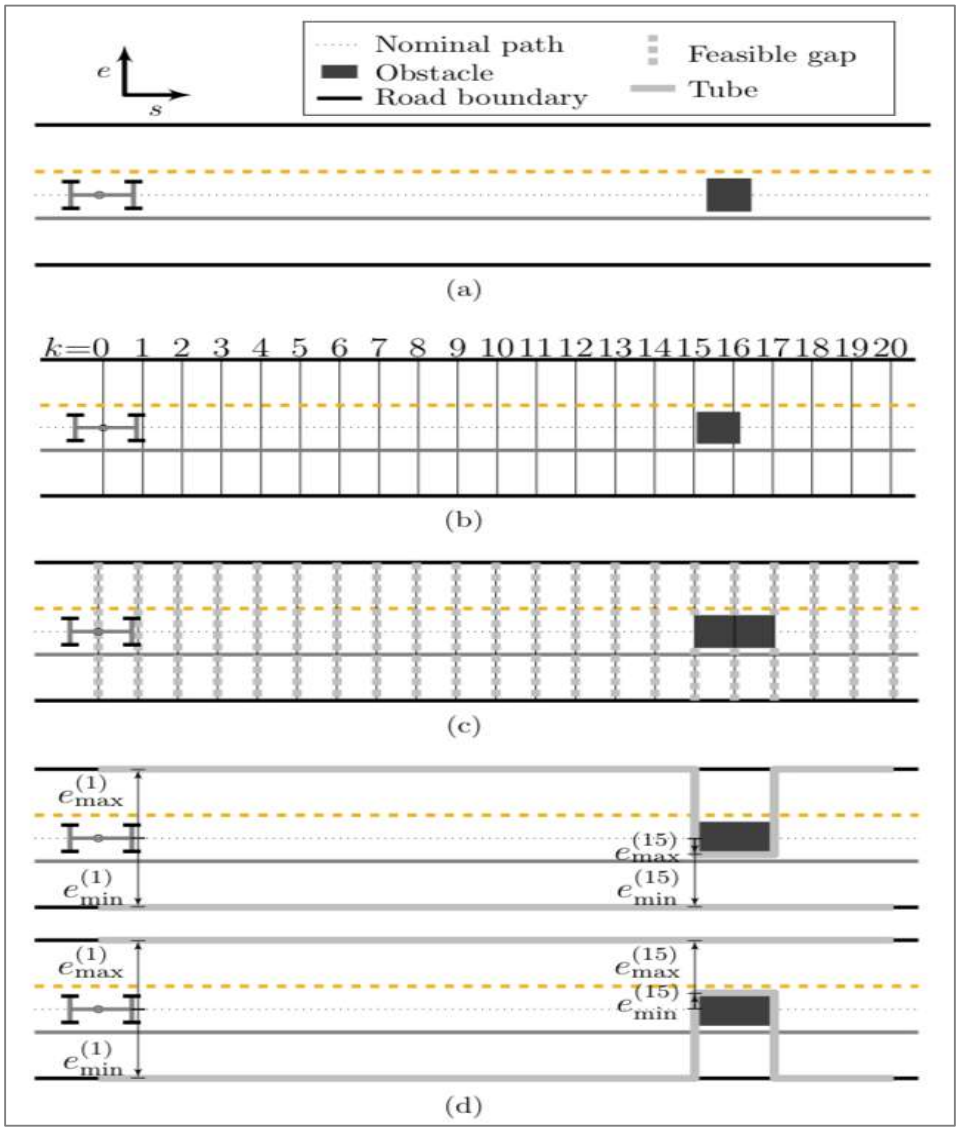

Figure 3.10: Environmental envelope generation using two tubes examples. (a) starting with a set of obstacles along the nominal path, (b) discretization along the s direction, (c) extension of objects along that same s direction, which creates alignment with the discretization and from which feasible gaps between objects are identified, and (d) connecting adjacent gaps into tubes which define maximum (e (k) max) and minimum (e (k) min) lateral deviation from the nominal path at each time step (k).

Due to the characteristic that every linear combination of produced trajectories included within a tube will also be contained within that tube, the set of collision-free trajectories corresponding to a single tube is a convex set. This characteristic allows for the rapid identification of optimum trajectories with fast optimization techniques. The lateral deviation (e) bound for each time step (k) represented by each tube is given by the following linear inequality:

$$H_{env}x^{(k)} = G_{en}^{(k)} \quad \text{Equation (3.20)}$$

and,

$$H_{env} = \begin{bmatrix} H_{env,left} \\ H_{env,right} \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & -1 \end{bmatrix} \quad \text{Equation (3.21)}$$

$$G_{env} = \begin{bmatrix} G_{env,left} \\ G_{env,right} \end{bmatrix} = \begin{bmatrix} e_{max}^{(k)} & -\frac{1}{2}d - d_{buffer} \\ -e_{min}^{(k)} & -\frac{1}{2}d - d_{buffer} \end{bmatrix} \quad \text{Equation (3.22)}$$

The lateral deviation boundaries for time step k are provided as $e_{max}^{(k)}$ and $e_{min}^{(k)}$, the vehicle width is d, and the environmental envelope is represented by the subscript env. A buffer, which specifies a recommended minimum distance between barriers and the vehicle and may account for vehicle orientation changes in establishing minimum gaps between obstacles, can be used to further increase occupant comfort.

3.6.2.4 Traffic Regulations

The distinction between rule-based and cost-based design is most unclear in traffic legislation. This contradiction is exemplified by the case presented in section 3.6.1. Traffic regulations are deontological in nature since they impose structures and norms. Humans, on the other hand, are not necessarily deontological in their approach to traffic regulations. In the real world, drivers in section 3.6.1 scenario make decisions based on criteria including clearance from the obstruction, traffic in the opposing lane, and overtaking speed. After then, the motorist must decide whether to cross the double yellow line. Human compliance with traffic regulations appears to be less deontological and more like a consequentialist balancing of safety, mobility, and legality, given that people frequently choose to violate the line, notably when passing a cyclist. As a result, while transitioning from human driver activities to programming a self-driven car, the option of whether to consider traffic regulations as deontological or consequentialist is critical.

As discussed in Section 3.6.1, if traffic regulations are established as a rule, they can easily result in traffic gridlock. When laws are defined as a cost, it implies that they are designed to

be violated from the start. Given the problem, a "soft" constraint is used to encode traffic regulations in the MPC formulation by including a slack variable. The cost of a constraint violation is scaled using the slack variable. Because the slack variable augments the constraint to make it less stringent, the constraint is considered consequentialist when the cost is comparable to other objectives. Due to the substantial cost associated with making the slack variable value non-zero, a very high weight on the slack variable leads the constraint to dominate all other objectives in a deontological way. To translate the morally motivated design decisions into the algorithm, a cost for the slack variable is incorporated, which corresponds to either crossing the road divider (S_{letf}) or joining the road shoulder (S_{right}). As shown in section 3.7, treating traffic regulation compliance as deontological or consequentialist leads in considerably different vehicle behaviour. These many driving results highlight the necessity of aligning programming decisions with social expectations, and these philosophical frameworks aid in thinking and justifying the driving behaviour to use.

3.6.3 Model Predictive Control Formulation

The vehicle's optimal path is determined by the control algorithm, which considers the costs and limitations imposed on its motion. The vehicle divides the globe into many viable tubes, each representing a convex optimization problem. The vehicle then determines the most cost-effective path through each tube, as shown in figure 3.11. The mathematics underlying determining the best path in each tube and its connected cost is described in the following sections.

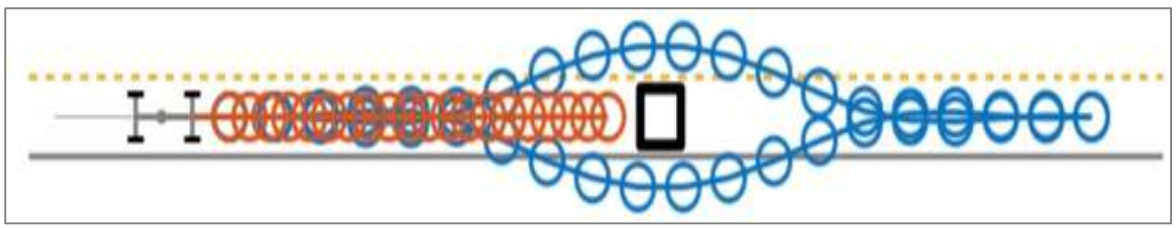

Figure 3.11: The three tubes define the generic manoeuvre options to avoid an obstacle. The left and right tubes are depicted in blue while stopping is depicted in red

3.6.3.1 Vehicle Model

The MPC controller uses a four-state bicycle model as its vehicle model. Vehicle sideslip (β) and yaw rate (r) are two velocity states, while heading deviation ($\Delta\psi$) and lateral deviation (e) are two position states, all of which are detailed in Appendix -18. As a result, the vehicle state vector is given in equation (3.23)

$$x = [\beta \quad r \quad \Delta\psi \quad e]^{\tau} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3. 23)}$$

The actuator is believed to be front steering in this chapter, and the vehicle is equipped with steer-by-wire technology. This allows the computer programme to control the lateral front tyre force that is wanted (F_{yf}). The findings are based on a constant longitudinal speed maintained by a PD cruise controller unless the car must stop, which isn't essential.

3.6.3.2 Optimization Formulation

The optimal path and control inputs for each tube in the environment are the result to the following optimization problem.

$$\text{Minimize: } \sum_k x^{(k)T} Q^{(k)} x^{(k)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24a)}$$

$$+ R \sum_k \left\| F_{yf,opt}^{(k)} - F_{yf,opt}^{(k-1)} \right\|_2^2 \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24b)}$$

$$+ \sum_l [\sigma_{env} \quad \sigma_{env}] S_{env,opt}^{(l)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24 c)}$$

$$+ \sum_l [\sigma_{tra} \quad \sigma_{tra}] S_{tra,opt}^{(l)} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24d)}$$

$$\text{subject to: } x^{(k+1)} = A_d^{(k)} x^{(k)} + B_d^{(k)} F_{yf,opt}^{(k)} + d_d^{(k)} \ldots\ldots\ldots \text{Equation (3.24e)}$$

$$\left| F_{yf,opt}^{(k)} \right| \leq F_{yf,} max\ slew \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24f)}$$

$$k = 0 \ldots\ldots (T-1)$$

$$H_{env} x^{(l)} \leq G_{env}^{(l)} + S_{env,opt}^{(l)} + S_{tra,opt}^{(l)} \ldots\ldots\ldots\ldots \text{Equation (3.24g)}$$

$$l = (T_{split} + 1) \ldots\ldots T$$

$$\left| F_{yf,opt}^{(i)} \quad F_{yf,opt}^{(i-1)} \right| \leq F_{yf,max\ slew} \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (3.24h)}$$

$$i = 0 \ldots\ldots T$$

Where $T_{split} + 1$ is defined as the case when the time steps for the environmental envelope are longer. The lateral front tyre forces ($F_{yf,opt}$), vehicle state (x), slack variable on environmental constraint ($S_{env,opt}$), and slack variable on traffic regulation, the variables to be optimised ($S_{tra,opt}$). The costs of vehicle states (Q), the cost of input change (R), and the costs of slack variables (*env* & *tra*) are the adjustable factors in this optimization problem.

The slack variables are used to create a hierarchy of deontological constraints and ensure that the problem always provides a viable solution. As a result, the unbounded slack variables exist. In a deontological paradigm, a slack variable with greater weights has a higher priority. The slack variables are utilised to make the constraint version of traffic regulations weighted lower than the obstacle avoidance version. They might also be used to include vehicle stability constraints like those in (Beal and Gerdes 2012) and (Bobier and Gerdes 2013) in a more complete form of a vehicle control system. In a deontological sense, as established by (Funke, Brown et al. 2017), the slack variable weights for such constraints should be put below those for collision avoidance. Other cost terms, such as those proposed by (Wei, Dolan et al. 2010), might be included into the cost function.

The Optimization problem (3.6) is a quadratic programme with a very sparse structure that can be solved in real time using an efficient solution. In this work, CVXGEN, created by (Mattingley and Boyd 2012), is utilised to find the best lateral front tyre forces ($F_{yf,opt}$). The initial solution ($F(0)_{yf,opt}$), which is then translated to the required steering angle as stated in Appendix -18, provides the control input to the self-driven car for the next time step. Appendix-19 shows a different problem formulation in which the road lane dividers and shoulders are added as additional constraints with a slack variable.

3.7 Simulation Results

The scenario from section 3.6.1 prepares the groundwork for showing an ethically driven automotive design in real-world simulations. The vehicle selects a different tube or a different trajectory inside that tube reliant on the driving condition and the engineering design decisions made in the algorithms. The weights are varied in these simulations to represent various interpretations of the traffic regulations. As a result, the philosophical argument of various goals may be transformed into the actual motion of the vehicle using mathematics. The weights' actual numerical values aren't important; rather, the relative values have an impact on the

optimization problem's solution. They can be determined by perceptual algorithms or developed as a function of geographic indications in practise.

3.7.1 Driving Scenarios

This section covered the simulation of dynamic complex systems and observing the emerging behaviour based on model-based driving scenarios where the entities of vehicles are separated and independent from the environment or scenario in which they operate. The content of the model will be dynamically changing, and model logic will determine when entities enter and leave the simulation. It involves many entities of different fidelity and properties. The simulation of the interaction of the environment and entities which are autonomous, heterogenous, parallel and with different life spans. The modelling approach is sectioned into three segments, namely, environment (simulation scenario where the agent resides, i.e., road, pedestrian etc), agent (heterogeneous agent with dynamics), and interface (data exchange among agents and the environment).

The interface takes in either state of one of the outputs from the self-driven pool and fixes it into one of the chosen scenarios in the environment. These scenarios (single-lane curve, dual-lane merge, dual-lane switch, three-lane merge & switch, three-lane curve & switch, and four-lane merge & switch) are modelled in variance, which allows the system to make one block active at any given time during simulation. Each scenario has a sim-event road model subsystem where entities are generated, routed, terminated, and are connected to the driver strategy model of a self-driven agent pool through an interface that allows information flow between the subsystems. It's a glue that combines two domains together.

For instance, vehicles in any scenario can exit the system when they hit the crossing block, which shows that the position has hit a limit using information from the self-driven agent pool. In this case, the vehicle is removed from the simulation with the terminator block. The various simulation outputs are as shown in figures 3.12 – 3.17 below. With this flexible framework, it is easier to model ethical issues and other types of agents and the environment in which they operate.

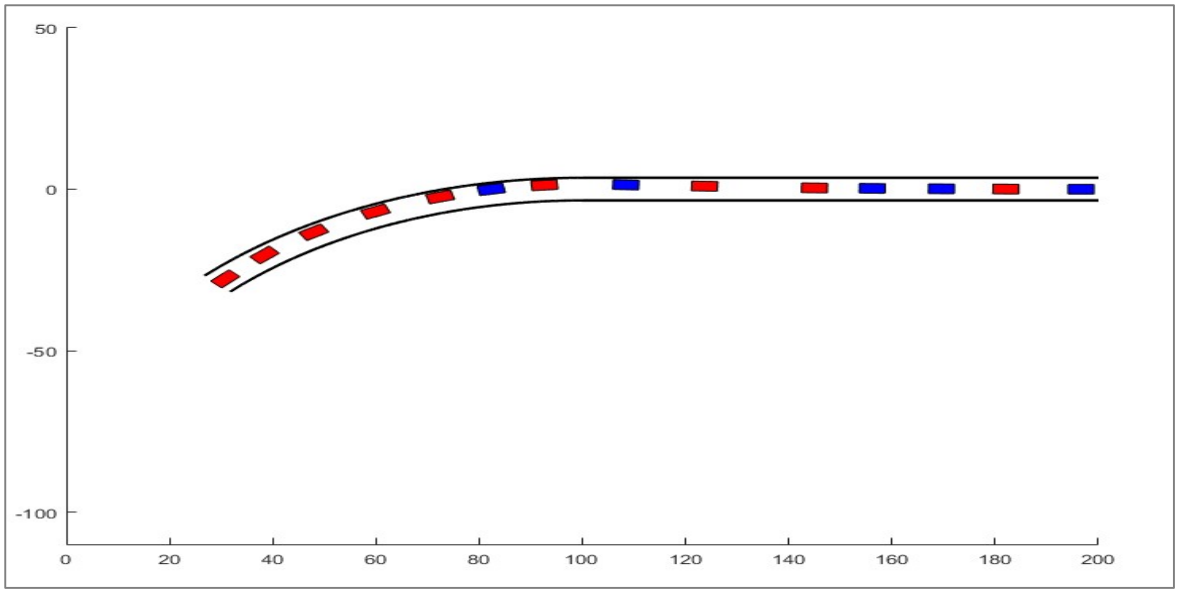

Figure 3.12: Single-lane Curve

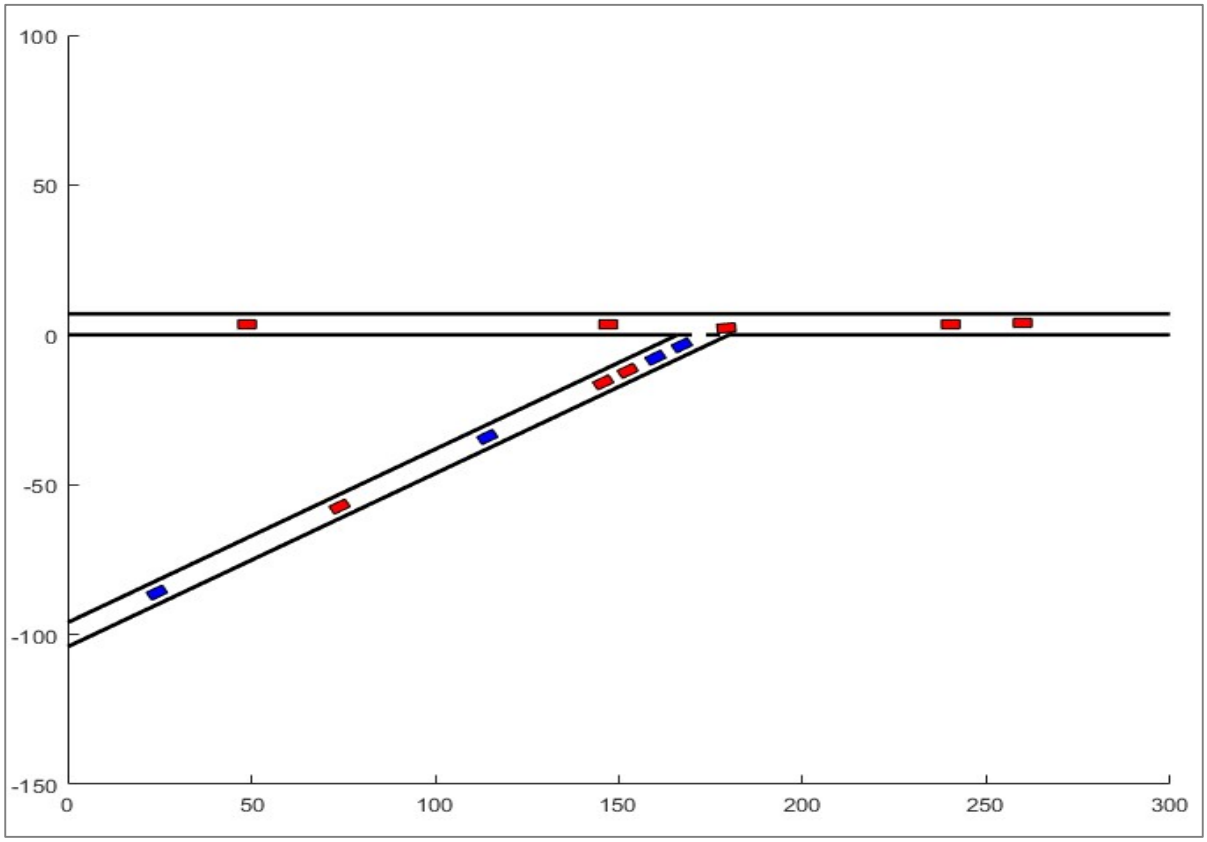

Figure 3.13: Dual-lane Merge

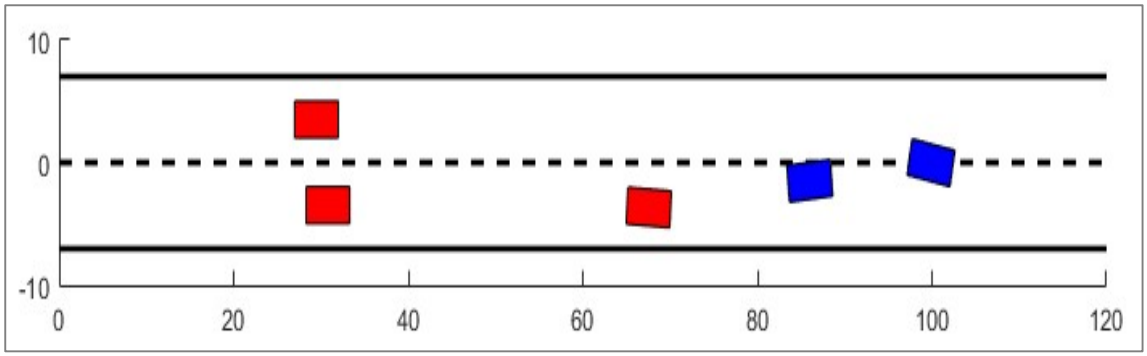

Figure 3.14: Dual-lane Switch

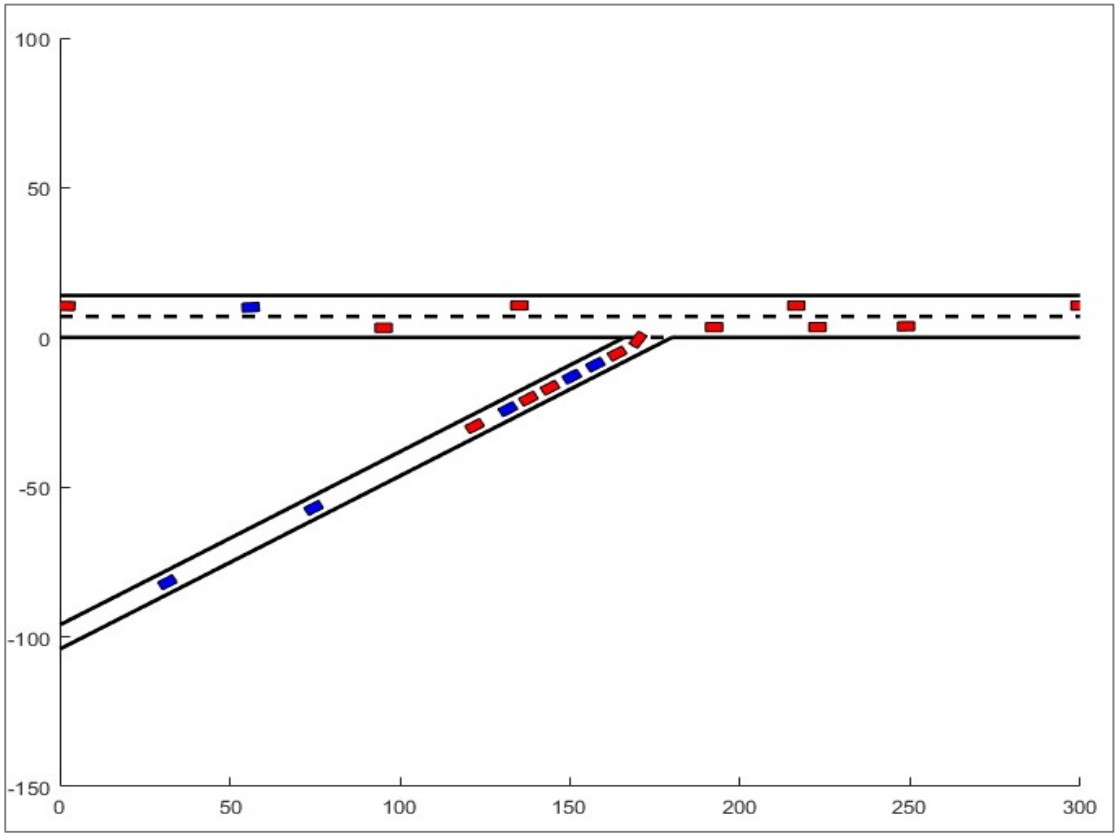

Figure 3.15: Three-lane Merge & Switch

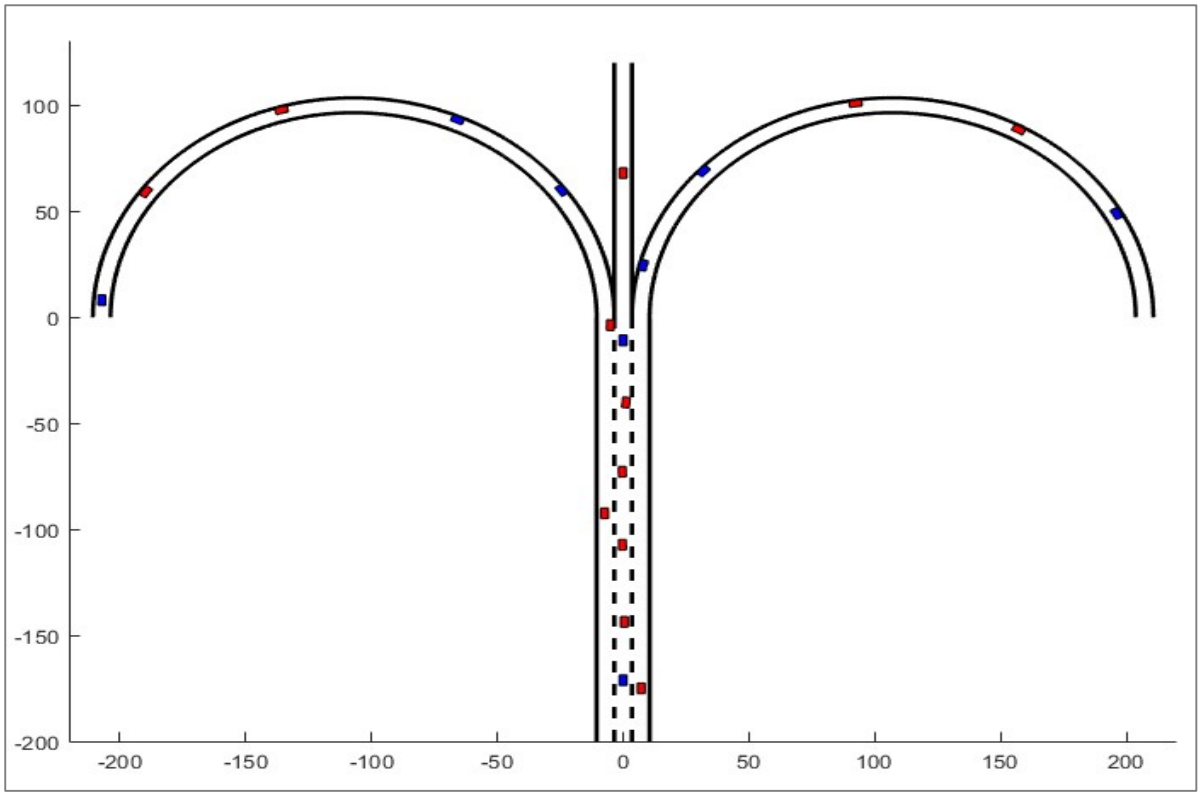

Figure 3.16: Three-lane Curve & Switch

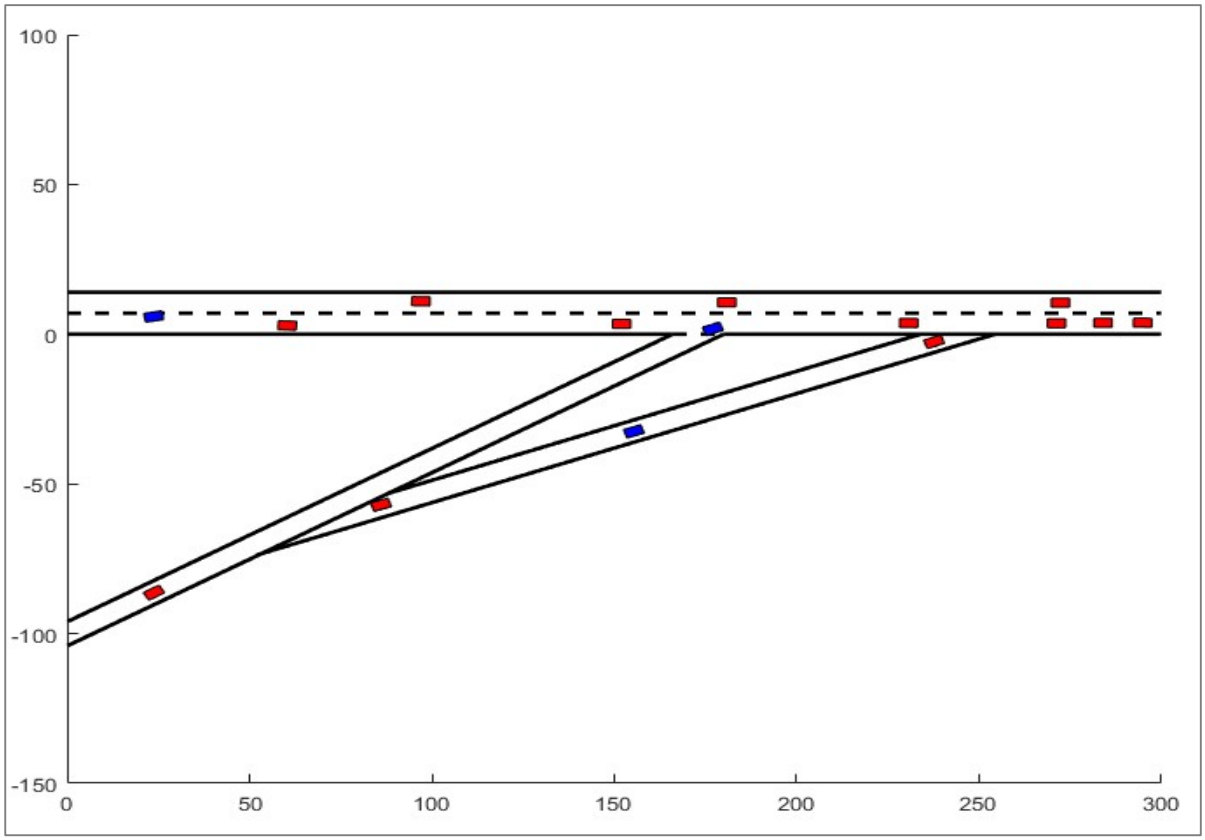

Figure 3.17: Three-lane Curve & Switch

3.7.2 Consequentialist Costs of Traffic Regulations

The decision to include traffic regulations as slack variables in the MPC formulation is explained in section 3.6.2. can have an impact on how the vehicle behaves when navigating the situation described in Section 3.6.1. A consequentialist method provides for greater flexibility in weighing road limits to reflect the strictness of the consequences of crossing it. Table 3.2 depicts a set of weights in which the shoulder is effectively treated as a hard constraint by assigning a high cost to the slack variable (representing, for example, a curb on the side of the road), whereas the double yellow lane divider is treated more as a cost by assigning a lower weight to it. The car manoeuvres to the left of the obstruction and passes the divider, as shown in figure 3.18. The car does not cross far into the opposite lane, instead staying near to the obstruction, because to the comparatively high weight placed on the divider in comparison to the path tracking weight.

After trading costs on the road shoulder and the road divider, the car travels to the right of the obstruction in figure 3.19. (representing stricter adherence to the regulation). The weights used for this case are shown in table 3.3. The trajectory and resultant steering angle are both mirrors of the initial instance, as the weights define basically a mirror copy of the prior simulations.

Table 3.2: weights resulting in a pass on the left

Parameter	Symbol	Value	Unit
Lateral error	Q_e	0.7	m^{-1}
Heading error	Q	0.5	rad^{-1}
Smoothness	R	0.1	kN^{-1}
Environmental slack	σ_{env}	500	m^{-1}
Road divider slack	σ_{left}	10	m^{-1}
Road shoulder slack	σ_{right}	150	m^{-1}

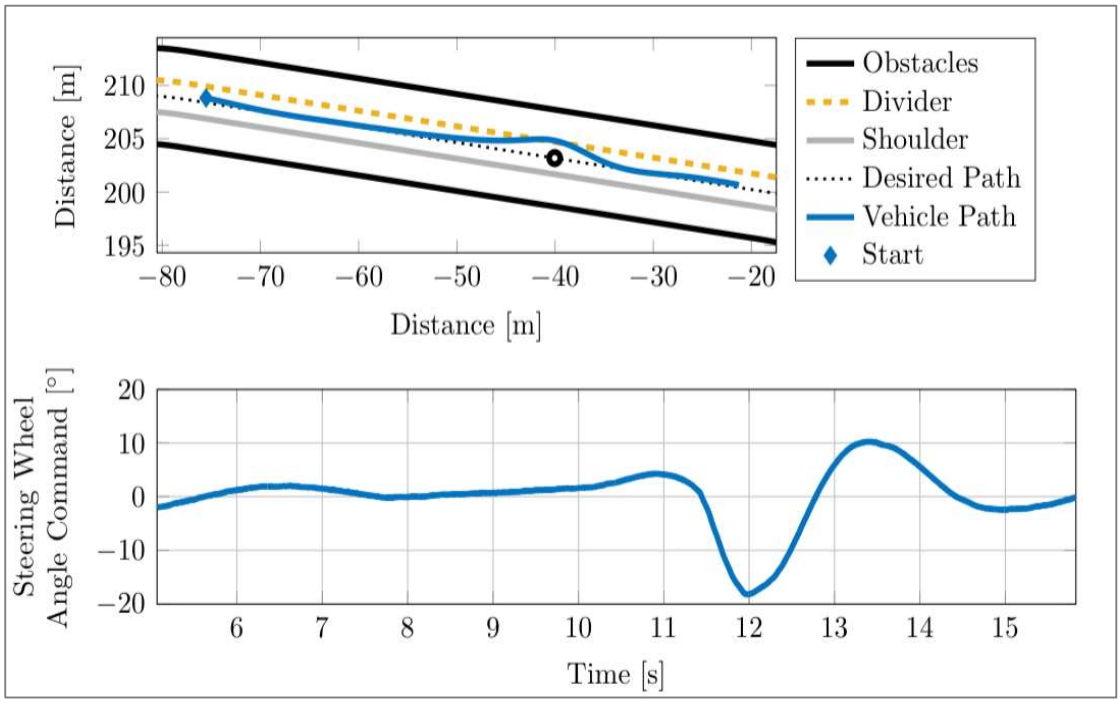

Figure 3.18: Vehicle manoeuvres to the left of the obstacle and crosses the divider (The left tube is chosen because the traffic lane divider is considered safe to cross).

Table 3.3: weights resulting in a pass on the right

Parameter	Symbol	Value	Unit
Lateral error	Q_e	0.7	m^{-1}
Heading error	Q	0.5	rad^{-1}
Smoothness	R	0.1	kN^{-1}
Environmental slack	σ_{env}	500	m^{-1}
Road divider slack	σ_{left}	150	m^{-1}
Road shoulder slack	σ_{right}	10	m^{-1}

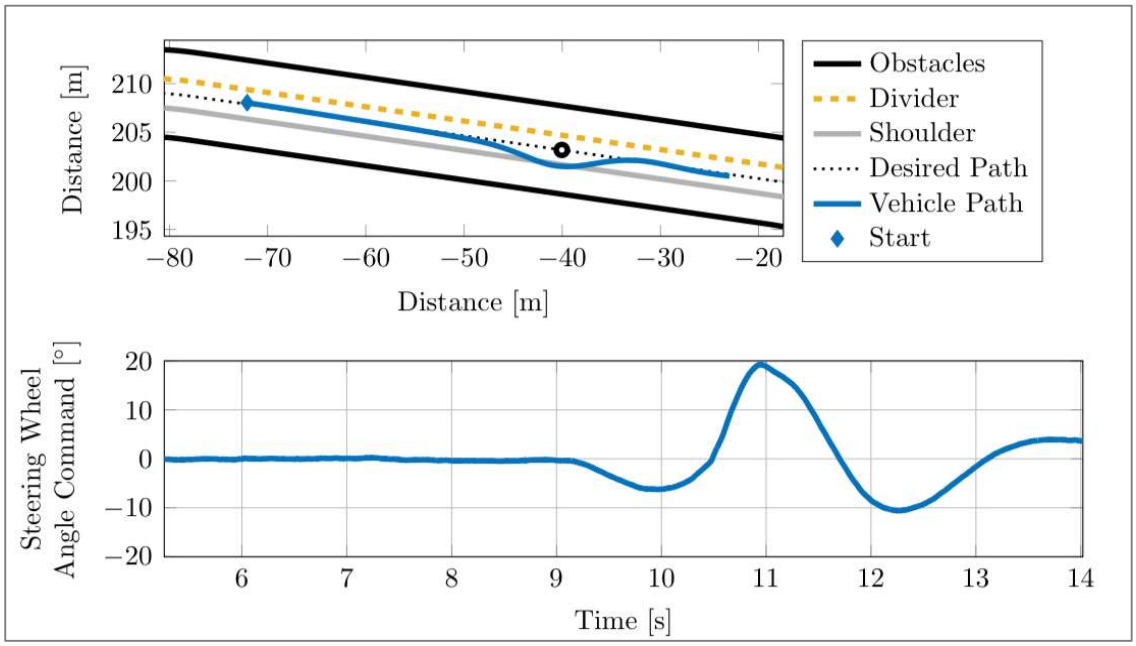

Figure 3.19: Vehicle manoeuvres to the right of the obstacle (The right tube is chosen because evaluation of the scenario determined it is safer to pass around the obstacle via the road shoulder).

3.7.3 Deontological Constraints in Traffic Regulations

Deontologically describing traffic regulations as hard norms is another philosophical technique to accounting for traffic laws. The road boundary constraints begin to be like hard or deontological restrictions when the slack variable weights grow compared to the other weights. With this weight selection, the tubes to the left and right of the vehicle have unacceptably high costs, as indicated in table 3.4, thus the vehicle must remain in the tube that corresponds to the lane. The car must come to a complete pause since the tube is obstructed by the barrier. Figure 3.20 depicts the course of the vehicle while a separate longitudinal PD controller directs a braking force to bring it to a stop before the tube's end.

Different treatments of environmental limits and accompanying traffic rules were integrated into the MPC formulation. Treating the double yellow lane line as a rigid, deontological restriction, on the other hand, removes a lot of freedom from the vehicle path and fails to reflect how humans drive. This implies that traffic regulations may need to be changed to provide programmers the same degree of freedom as human drivers. Without that choice, programmers must use a consequentialist method to decide how much to weight the traffic regulations. The relative values of the weights used for the traffic slack variables dictate how the car manoeuvres around obstacles in this MPC formulation. When a lane divider is a single, dashed yellow line or a double yellow line, these values can be affected by geographic indications. The weights can also be computed by a perception algorithm that uses data from lidars and cameras to assess the ground's safety in each tube. The MPC formulation's versatility in accounting for responsible decision-making is demonstrated by these findings.

Table 3.4: weights resulting in a pass on the left

Parameter	Symbol	Value	Unit
Lateral error	Q_e	10	m^{-1}
Heading error	Q	1	rad^{-1}
Smoothness	R	0.1	kN^{-1}
Environmental slack	σ_{env}	500	m^{-1}
Road divider slack	σ_{left}	150	m^{-1}
Road shoulder slack	σ_{right}	150	m^{-1}

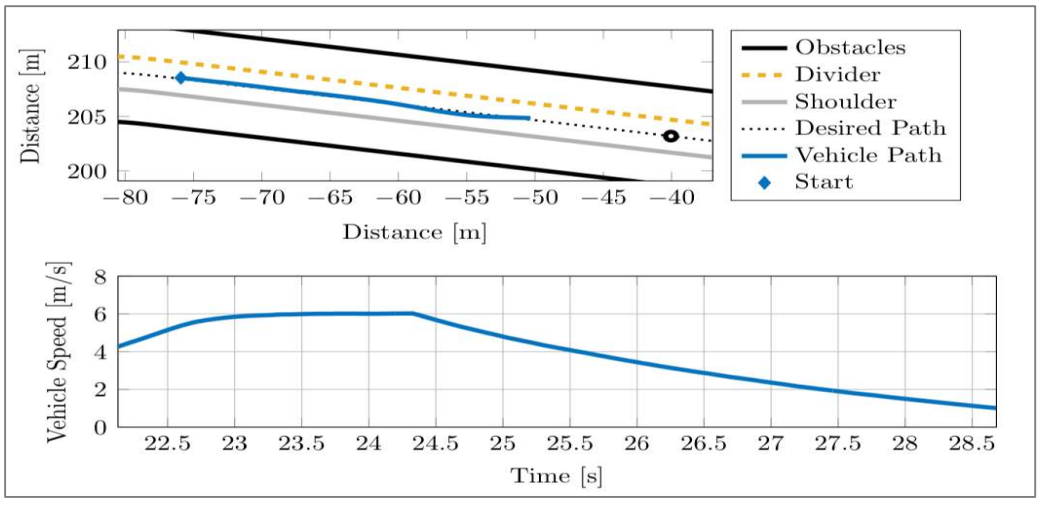

Figure 3.20: Vehicle remain in the tube corresponding to the lane (Since left and right path options are weighted equivalently, the tube is blocked and the vehicle brakes to a complete stop).

3.7.4 Costs, Constraints, and Weights

The preceding illustrations show that depending on whether traffic regulations are interpreted deontologically or consequentially, self-driven cars might act quite differently. There are two advantages to mapping the costs and constraints of deontology and consequentialism. As shown in this chapter, it may be used to explain a design objective as a cost or constraint by utilising deontological or consequential reasoning. Also, if an engineer develops a cost or constraint-based method, this mapping might reveal what sort of behaviour the implementation might entail.

To further grasp the consequences of this mapping, moral psychologist Greene proposes that deontological rationalisation occurs in sensitive circumstances, whereas consequential reasoning occurs in less severe situations (Greene and Sinnott-Armstrong 2008). This explain why, until the severity of the situation worsens, sticking to the double yellow line is best portrayed as a cost (i.e. oncoming traffic or erratic behaviour of the obstruction). Treating some traffic regulations as rules may be annoying, though not inherently restricting, when the circumstance is low risk.

The choice of weights in the optimization problem eventually affected the vehicle's compliance with traffic rules, in addition to the costs and limitations (left and right). The weights used in the optimization problem eventually affected the vehicle's compliance with traffic regulations, in addition to the costs and constraints chosen (δ_{left} and δ_{right}). Path deviation (Q_e and $Q_{\Delta\psi}$), obstacle avoidance (δ_{env}), and even occupant comfort (R) is all affected by the other weights in the cost function. Because some of these weights are just in the cost function (and therefore not a slack variable), they can only have a substantial impact on the vehicle's behaviour. Nevertheless, the concepts of deontology and consequentialism do not give insight into the selection of weights that may impact the vehicle's behaviour in other ways.

3.8 Vehicle Behaviour

Until now, the MPC formulation has been influenced by traffic circumstances for a single vehicle. The findings reveal the concepts that fortify consequentialism and deontology as ethical theories, as well as how the theories may be applied to different types of vehicle behaviour. These theories, on the other hand, do not clearly guide the selection of relative numerical weights, which has a significant influence on design objectives for self-driven cars beyond safety. As a result, this part is inspired by the introduction of virtue ethics, a third

normative ethical theory. As stressed by deontology and consequentialism, virtue ethics focuses ethical behaviour on character rather than right acts or consequences. A choice is ethical if it follows the inclination of a moral being, according to the virtue ethics paradigm. To put it another way, moral individuals act virtuously if they always do the right thing in the right moment, according to their character (Nay and Zagal 2017).

The concept of an agent's character naturally leads to a more particular concept termed role morality, which will be used in this study. Role morality is the concept that behaviour that is acceptable within the framework of a certain professional role and circumstance may not be acceptable outside of that context (Evans 2017). In disciplines such as law and medicine, role morality is used to explain behaviour that would be considered unethical if it occurred outside of a professional setting. Samson, the Parisian executioner, is an extreme case, as (Applbaum 2000) points out. A less severe example is a doctor providing medicine to someone who isn't his or her recognised patient. While it is permissible to write prescriptions within the professional boundaries of a doctor-patient relationship, it is not permitted outside of this position. These acceptable roles and codes of behaviour are founded on public expectations of the service given by experts in that specific area; thus, role morality is drawn from a collective decision on what is best for society (Nay and Zagal 2017), rather than from any individual in charge of establishing the rules.

The kind of duty or character that various vehicles should have been thus a significant problem in the creation of self-driven. The function of a vehicle has an impact on how strictly it must adhere to traffic regulations. Figures 3.18 to 3.20 demonstrate how a self-driven car can decide whether to break the traffic regulation of respecting a double yellow line boundary for safety reasons. The degree of obedience or violation necessitates the use of a guiding principle, which might be influenced by role morality. Vehicles serve several functions in society. It is acceptable for an ambulance to run a red light when transporting a passenger to the hospital who is in a life-threatening condition: the function of an ambulance in society is to take patients to the hospital as fast as possible to save lives. A cab transporting a harried customer, on the other hand, may not speed through a red light to save time since its societal function does not justify it. While deontology and consequentialism allow vehicle aims to be justified as constraints or costs, morality can assist in determining the strength of the applied rules and costs for various vehicles. The background for why an ambulance might be programmed to contemplate breaching traffic regulations more freely than a taxi, for example, is set by role morality.

The nature of a vehicle that can acceptably breach rules, such as an ambulance, can be represented using the MPC formulation provided in section 3.6.2 by changing the weights for various purposes. The weight of tracking errors can be decreased when trying to replicate the desirable behaviour of an emergency response vehicle, for example. Because lateral and heading errors have smaller costs, the vehicle has more flexibility to depart from the path. figures 3.21 and 3.22 illustrate the simulation findings using the weights from tables 3.5 and 3.6, respectively. The technique outlined in section 3.6.3 was used to calculate these relative weights. The emergency vehicle begins executing the manoeuvre sooner, as seen in figures 3.21 and 3.22. This is due to the lower relative costs of lateral and heading error in terms of smoothness, allowing for higher path deviation. The smoothness of the manoeuvre might be beneficial to an injured passenger in this situation.

While a vehicle may be programmed to weight traffic regulations and manoeuvre objectives based on its social function, having an engineering system that does not always follow the rule is a worrying prospect. This raises another moral question: should the engineers who build such a system supervise determining the rules' weights? Is adapting the regulation to the programming reality of self-driven systems a better option? While translating philosophical frameworks into technical terms does not provide a straightforward answer, it can assist in raising the relevant issues that must be answered to deploy self-driven cars.

Table 3.5: weights resulting in a pass on the left

Parameter	Symbol	Value	Unit
Lateral error	Q_e	0.3	m^{-1}
Heading error	Q	0.25	rad^{-1}
Smoothness	R	0.1	kN^{-1}
Environmental slack	σ_{env}	500	m^{-1}
Road divider slack	σ_{left}	5	m^{-1}
Road shoulder slack	σ_{right}	200	m^{-1}

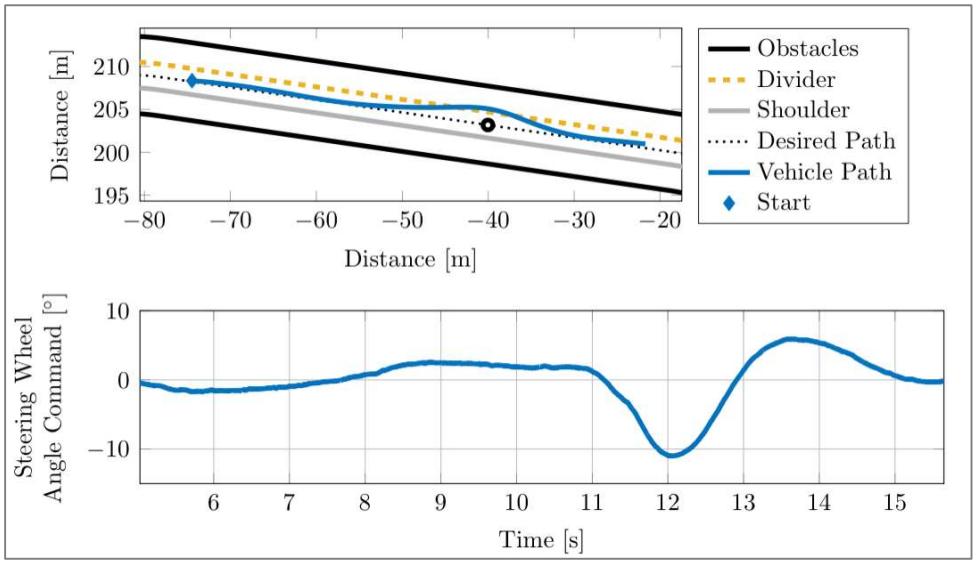

Figure 3.21: The relatively lower costs on lateral and heading error allow the vehicle more freedom to deviate from the path (Reduction of the cost on path following allows the vehicle behaviour to model emergency response vehicle character).

Table 3.6: weights resulting in a pass on the left

Parameter	Symbol	Value	Unit
Lateral error	Q_e	0.3	m^{-1}
Heading error	Q	0.25	rad^{-1}
Smoothness	R	0.1	kN^{-1}
Environmental slack	σ_{env}	500	m^{-1}
Road divider slack	σ_{left}	200	m^{-1}
Road shoulder slack	σ_{right}	5	m^{-1}

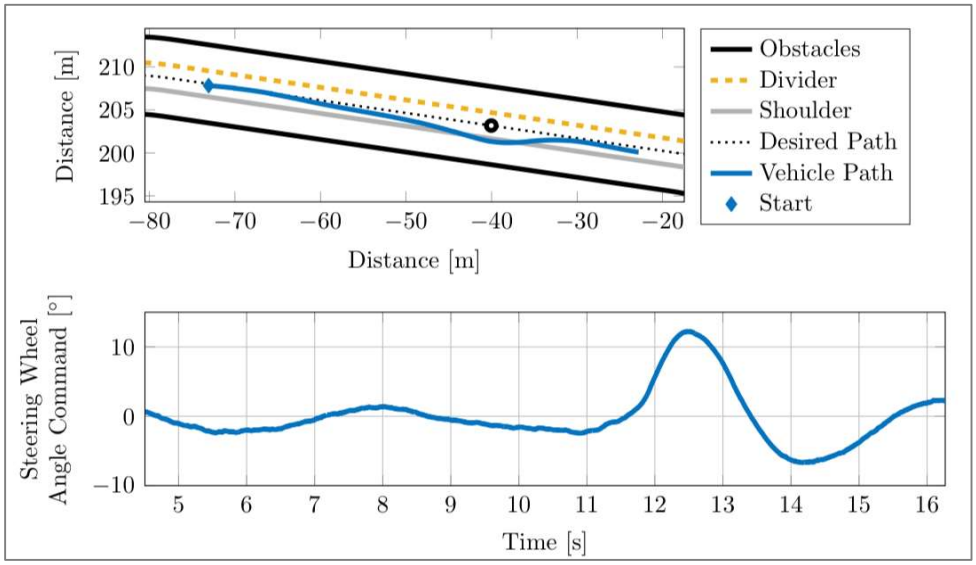

Figure 3.22: The relatively lower costs on lateral and heading error allow the vehicle more freedom to deviate from the path (Reduction of the cost on path following allows the vehicle behaviour to model emergency response vehicle character).

3.9 Conclusion

Deontology and consequentialism are normative ethical theories that help engineers comprehend the consequences of various design decisions while programming autonomous cars. Constraints and cost functions are examples of rule- and cost-based engineering approaches. By making these linkages, engineers working at the most critical levels of programming self-driven cars will be able to link their design decisions to wider social acceptability concerns. Engineers can utilise the mapping of philosophical principles to mathematical frameworks as a technique for thinking and reasoning about autonomous vehicle motion planning systems. Through an MPC formulation, this chapter looked at how to prioritise objectives like path tracking, vehicle occupant comfort, and traffic regulations in the cost function while constraining obstacles like vehicle slew rate restrictions. Different weighting systems within the control formulation, depending on the vehicle type and purpose, are based on the idea of role morality. Additional vehicle objectives may be required in more complicated circumstances. However, the basic obstacle avoidance manoeuvre described earlier in this chapter highlights some of the difficulties in combining legal compliance with the desire to coexist peacefully with human drivers in traffic. To reap the benefits of self-driven cars, legal and ethical issues must be better integrated into the control code. A formalisation of human values might aid society in better comprehending and trusting a self-driven car technology.

Chapter 4: Human-centred Decision Making

4.1 Introduction

Recognizing the similarities between normative philosophical and mathematical frameworks aids in the formulation and discussion of specific algorithms for self-driving vehicle decision making. The parallelism also aids in comprehending the consequences of different objective trade-offs on human worth. However, how can engineers know which values to include in the algorithms? Mobility, safety, and legality aren't the only human values. The formal, iterative process of value-sensitive design (VSD) is used in this chapter to develop ethical considerations that link values to technical specifications.

Many stakeholders, such as pedestrians, bikers, and car occupants, use the roads, and they all have values that shape their expectations. Public perceptions of autonomous vehicle driving behaviour are expected to be shaped by similar human values. Some appropriate criteria to examine include mobility, safety, and legality (Point, by City et al. , Mehrara Molan and Ksaibati 2021). The difficulty for self-driving vehicle designers is to relate these human values to engineering standards. Integrating stakeholders and their values into the design process of algorithms for self-driving vehicle decision-making algorithms is one method to overcome this problem.

Many design methods consider human values and requirements, as well as the demands of diverse stakeholders. Human-centred design (HCD) is a well-known design method (Maguire 2001, Giacomin 2014). HCD techniques entail engaging with a group of stakeholders, most of whom are direct users of the technology, for the designers to receive feedback on how to enhance the design. Current HCD techniques, according to (Miller and Cushman 2018), lack an ethical perspective in terms of justifying designs or recognising their possible ethical consequences. Life-based design (LBD) (Leikas, Sigfrids et al. 2020) is another design method that incorporates an ethics investigation. LBD investigates the demands of stakeholders via their quality of life, taking a comprehensive and holistic approach to the design challenge. It begins with determining human needs in the design activity, then users and technology requirements, and finally whether the human quality of life improves because of the created technology. An ethical evaluation is clearly included in the investigation of the improvement in quality of life. LBD is justified by the fact that the technology must increase the users' quality of life. Both HCD and LBD are iterative design methods that aim to enhance technology design for their respective consumers. HCD is concerned with usability values, whereas LBD is

concerned with quality-of-life values. Numerous additional values must be examined when many stakeholders are affected by a proposed technology, especially when these values may clash.

Another human design method that incorporates ethical concerns is value-sensitive design (VSD) (Umbrello and De Bellis 2018), which does so early in the design cycle by openly exploring values typically prioritising those with ethical significance (Borning and Muller 2012) throughout the whole design process. VSD is a three-part technique that iterates through conceptual, technical, and empirical studies to solve any general design problem. Human values are linked to the devolved technology at every level of the design process. VSD is particularly useful in a design challenge if there are value conflicts concerning ethical problems according to (Friedman, Kahn et al. 2013) and (Manders-Huits 2011) show how identifying indirect stakeholders (a component of the conception process) revealed privacy issues for pedestrians in the design of an office space with a virtual window overlooking a public plaza. (Van Wynsberghe and Robbins 2014) utilise VSD to create a list of requirements that will guide future security system designs in implanted medical devices. Also, because of VSD's universality, it may be tweaked to fit specific design needs. (Van Wynsberghe 2013) adds the moral framework of care ethics to VSD to ensure that health care robots represent stakeholder values.

Engineers consider some human values while developing algorithms for self-driving vehicle motion planning. As Paden, and Cap's assessment framework of a self-driving vehicle approaching an unsignalized pedestrian crossing (Paden, Čáp et al. 2016) shows, many algorithm designs prioritise safety and efficiency. In the development of the reward function of a partially observable Markov decision process (POMDP) for speed control in pedestrian settings (Bai, Cai et al. 2015) and (Bouton, Nakhaei et al. 2018) similarly focus on safety and efficiency. As did (Schratter, Bouton et al. 2019) in the development of a reward function for entering occluded junctions in a speed control POMDP. Engineers seek to relate human values to engineered technology, as seen by these instances. The assessment frameworks and motion planning rules' emphasis on safety and efficiency shows the difficulties of designing for two opposing criteria. (Pouya and Madni 2020) also include occupant comfort in the reward function and recommend that traffic restrictions might be added in future POMDP design iterations. Their conversation reveals a willingness to consider the many human values in question while designing a motion planning policy. To account for various values, a technique that can assist in selecting which values to include would be beneficial, because humans' value more than simply safety and efficiency. Having a list of recognised values may also help you

figure out where there are conflicts between different stakeholders and morals. Value conflicts can be identified early in the design process, allowing engineers to create technology that expressly resolves them rather of relying on patchwork methods to manage value tensions after a system has been deployed.

It is claimed that VSD can aid in the development of motion planning algorithms for self-driving vehicles by filling in the gaps in the design process. VSD is used to validate the relationship between human values and engineering specifications by defining a more comprehensive list of human values at play in the design challenge and resolving value conflicts through design justification. The design task of a speed controller for the scenario of a pedestrian crosswalk is demonstrated in this chapter using a modified application of VSD for a self-driving vehicle decision making. The speed restriction is the only constraint on the path's pace. The self-driving vehicle will most likely need to slow down to properly negotiate the circumstance. Acceleration command from POMDP policies created with VSD govern the speed. The VSD speed controllers are first and second rounds of the design process, not final products. VSD's iterative approach aids in documenting how values are combined into the speed control design, as well as how tensions between values are addressed.

4.2 Value-Sensitive Design

VSD approach includes three stages: conceptual, technical, and empirical (Friedman, Kahn et al. 2013, Evans 2017) identifying the values included by the designed technology is a part of the approach at the conceptual phase. The direct and indirect stakeholders of the technology are also determined during the conception phase. Some technology implementations are more adapted to maintain some values than others, according to VSD. The technology being created is developed using the technical solutions that are most in accordance with the defined values (from the conceptual phase). Finally, the empirical phase allows for quantitative and qualitative assessments of the generated design, such as data analysis or findings from user research. During this time, you can check to see if the designed technology matches the concept. The designer iterates through the various phases of design development until all three are in sync. As they create new technology, engineers iterate implicitly through the conceptual, technical, and empirical phases. VSD is a tool that aids in the formalisation of the engineering process by recognising and monitoring values inherent in technology across iterations.

4.3 First Iteration

Because of the variety of scenarios that a self-driving vehicle may experience on the road, designing a decision-making algorithm for it is an extensive design challenge. The list of stakeholders and values may be unsustainable to design for as a first iteration, given the potential for such a broad influence. This chapter will focus on a specific scenario as a case study to limit the stakeholder and value consideration area and simplify the design effort. Figure 4.1 illustrates a two-lane roadway with a single, dashed yellow line. The route also has a well designated pedestrian crossing. A big, illegally parked vehicle sits just in front of the crossing. The crossing is partially obstructed by the blocking van from the perspective of the autonomous car approaching the crosswalk. The steering controller from Chapter 3 will continue to guide the car in a lateral direction around the van while avoiding obstacles. The goal of the design task is to create a speed control algorithm that allows the self-driving vehicles to safely drive the scenario along the provided path.

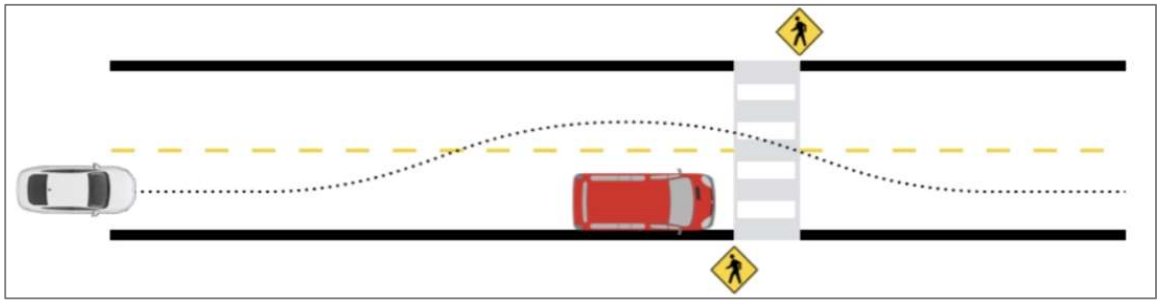

Figure 4.1: Scenario for an occluded pedestrian crosswalk

4.3.1 Conceptualization

The direct and indirect stakeholders engaged in the scenario and design challenge, as well as the human values, are identified to begin the VSD process. Engineers and programmers are forced to think more thoroughly about the repercussions and who is affected by the created technology when they identify both direct and indirect stakeholders. The self-driving vehicle, its occupants, any pedestrians who may cross the street, and the authority of traffic regulations are all direct stakeholders in this scenario. Because the autonomous car is expected to be able to pursue an obstacle-free path around the occlusion, the obstructing vehicle parked on the road is an indirect stakeholder. The focus of this initial iteration is on these stakeholders, although there are many others, including bicyclist and spectators.

VSD and the engineering process rely heavily on determining the human values at play in the scenario and design task. The human values of mobility, safety, and legality are all weighed in traffic scenarios. More values at risk can be discovered by examining the stakeholders. Human values to consider in this conception come from (Talhelm, Haidt et al. 2015, McNamara, Willard et al. 2019), rather than interacting with actual stakeholders. According to Haidt, human beings are born with a set of values (or moral foundations) such as care and respect for others, fairness and reciprocity, respect for authority, and individual autonomy, whereas Choi et al. believe that trust and transparency are critical for self-driving vehicle adoption. Because the way these moral values are articulated might lead to various technological solutions, more detailed definitions are offered by considering the stakeholders to explicitly describe what each value means in this scenario:

- ✓ *Individual autonomy:* Individual autonomy of the vehicle's occupants recognises the desire to go quickly from one location to another with minimal obstruction.
- ✓ *Respect for authority:* The autonomous vehicle's interaction with traffic regulations is based on respect for authority.
- ✓ *Care and respect for others:* The desire to avoid harming other individuals demonstrates care and respect for others.
- ✓ *Trust and transparency:* When a pedestrian think that an approaching vehicle will surrender to his or her right-of-way when crossing in a crosswalk, *trust* develops. While *transparency* occurs when the self-driving vehicle's activities help to build such confidence.
- ✓ *Fairness and reciprocity:* Fairness and reciprocity affect both vehicle occupants and pedestrian stakeholders in the sense that the self-driving vehicle should not conduct biased or discriminating behaviours based on data about the stakeholders. All personnel engaged with the autonomous vehicle should be treated equally.

Because these human values must be included in the decision-making algorithm, I link them all to an engineering specification for usage in the technical implementation phase as shown in Table 4.1.

Table 4.1: Human values mapping to engineering specifications for the first VSD iteration.

Human value	Engineering specification	Representation
Safety		
Legality		v_t
Care and respect for others	Safety and Legality	d_t
Respect for authority		c_t
Fairness and reciprocity		
Mobility	Efficiency	v_t
Individual autonomy		
Trust	Smoothness	a_t
transparency		Δt

4.3.1.1 Legality and Safety

The connection between legality and safety when crossing an obstructed pedestrian crosswalk is complicated. Pedestrians in crosswalks have the right-of-way, according to the (Navet and Simonot-Lion 2017). When approaching a crossing with the possibility of a pedestrian present, a car should slow down and be ready to stop, as required by Vehicle Code §21950:

- ✓ Except as otherwise provided in this chapter, a driver of a vehicle must give the right-of-way to a pedestrian crossing the street in a designated or unmarked crosswalk at an intersection.
- ✓ A pedestrian's responsibility to use reasonable care for his or her safety is not relieved by this section. A pedestrian may not abruptly abandon a curb or other safe location and walk or run into the path of a vehicle that is so near that it poses an imminent threat. While in a designated or unmarked crosswalk, no pedestrian may unduly halt or delay traffic.
- ✓ When approaching a pedestrian in a designated or unmarked crosswalk, the driver must take all reasonable precautions and lower the vehicle's speed or take any other measure required to ensure the pedestrian's safety.
- ✓ (d) Subdivision (b) does not relieve a driver of a vehicle of the responsibility to exercise appropriate care for the safety of any pedestrian crossing a street in a designated or unmarked crosswalk.

Following the law and driving safely are tightly linked, as the vehicle code implies. For this iteration, I assume that legality and safety are the same engineering criteria for this scenario: if the self-driving car follows the law, it'll also perform safe actions. Vehicle speed (v_t), distance to crosswalk (d_t), and whether a pedestrian is crossing the roadway are the essential pieces of information required for safe and lawful decision-making (c_t). Again, this ties up with the ethical principles of compassion and regard for others, respect for authority, and fairness and reciprocity.

4.3.1.2 Mobility and Efficiency

The human value of mobility is reflected in the time efficiency measure. The speed of the vehicle (v_t) for a particular path has a direct relationship with time efficiency. The moral value of individual autonomy is central to this goal.

4.3.1.3 The quality of smoothness

Smooth driving improves passenger comfort and fosters stakeholder confidence and transparency. Smoothness may be measured in longitudinal control by measuring the change in vehicle speed, which is the same as knowing the acceleration command (a_t) and the time change (t).

4.3.2 Technical Implementation of Technology

To handle the pedestrian occlusion problem, many decision-making techniques may be adapted. VSD contends that the choice of technology or algorithm implicates ethics, rather than just picking an approach at random. The longitudinal motion is selected to be controlled by a stochastic optimization problem, whereas the lateral motion is controlled by the formulation from Chapter 3. A stochastic optimization problem can balance the specified values while accounting for predicted uncertainty in the driving situation. The problem can also be expressed as an open-loop or closed-loop planning dilemma. Closed-loop planning considers future state information because it divides the planning problem into smaller sub-problems (i.e. dynamic programming), but open-loop planning such as stochastic MPC does not since it involves creating a static sequence of actions. (Gray, Gao et al. 2013). A closed-loop planning technique is used, with the problem represented as a partially observable Markov decision process

(POMDP) (Sunberg and Kochenderfer 2018), to get an offline policy to examine and validate before placing on a self-driving vehicle. Every design decision made throughout the development of the POMDP is linked to values from the conceptualization phase to rationalise the engineering and openly document their embedding of these values.

4.3.2.1 Markov Decision Process with Partially Observation

An agent in a POMDP makes decisions depending on its previous observations. O_1, ……. O_t. The history is summarised in a belief state b, which is a distribution across the states, to decrease the amount of data kept. The best policy is represented by a series of alpha vectors that translate the belief state into a control input or action. The state vector captures the information needed to address each value in the goal function, given the values of safety and legality, efficiency and mobility, and smoothness.

$$x = [v_t \quad d_t \quad c_t]^T \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.1)}$$

and the control input,

$$u_t = a_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.2)}$$

where v_t denotes vehicle speed, d_t denotes vehicle distance from crosswalk, c_t denotes pedestrian detection, and a_t denotes longitudinal acceleration Appendix -20 has an alternate formulation that expresses the activities as desired speed. Because the roadway's peak speed is assumed to be *10m/s*, the vehicle speed is constrained by the speed restriction to meet the safety and legality objectives. The pedestrian detection is a Boolean value since the pedestrian is either crossing or not, and the detection does not rely on additional information about the pedestrian that may be discriminatory to preserve the principles of fairness and reciprocity. The control input, or action, is set to a maximum of $3m/s^2$ to give pleasant acceleration and deceleration values, furthering the goal of smoothness for occupant comfort.

The distance to the crosswalk and vehicle speed is calculated using a point mass model of the vehicle for the dynamics (or state transitions). Over time, the detection of a pedestrian crossing retains some ambiguity. When a pedestrian is spotted, there is a 90% chance that the pedestrian will be detected again at the following time step. This probability was chosen to reflect the high possibility that the pedestrian will stay in the crosswalk while he or she crosses the roadway, while also admitting that the pedestrian will not stay in the crosswalk indefinitely. When the

pedestrian is not identified, there is a 50% probability that he or she will continue to be undetected, capturing the occlusion's ambiguity. This probability was chosen to represent the chance of a pedestrian appearing. In the first iteration, the pedestrian state transition probabilities are set haphazardly to show the process. The state transition probabilities might be derived using event-based statistics or another model in practise. The control loop assumes that the distance to the crossing and vehicle speed are both ideal. However, there is observation uncertainty for pedestrian crossings, with a false positive rate of 5% for detecting and a false positive rate of 5% for not detecting the pedestrian, capturing sensor noise uncertainty. These false positive rates were selected arbitrarily low, although they would be caused by the perception system's capacity to identify pedestrians in practise.

The aim is for the self-driving vehicle to seamlessly pass across the crosswalk in a safe and efficient manner while complying to all applicable traffic regulations. For each state and action, the reward function determines the stage cost $g(x_t, u_t)$, which ties the conception values to the technological implementation. The reward for a state action pair is calculated by summing the state and action's stage costs (4.4), (4.5), and (4.6).

The stage cost for legality and safety is partly generated from physical characteristics, resulting in a computed reward that is a function of the amount of deceleration required to stop the vehicle at a particular condition. The constant deceleration needed to come to a complete stop given the distance to the crossing and vehicle speed is calculated using the constant acceleration point mass equation.

$$a_t = - v_t^2 / 2d_t \quad \text{Equation (4.3)}$$

As a result, the following stage cost for safety and legality is calculated:

$$g_{safe}(x_t, u_t) = - (\zeta \frac{v_t^2}{d_{t+\varepsilon}} + \eta 1(d_t = 0)) 1(c_t) \quad \text{Equation (4.4)}$$

where $\varepsilon > 0$ is a denominator buffer to soften the constraint, $\zeta > 0$ is a weight on the penalty incurred by driving quickly as the vehicle approaches the crosswalk, $\eta > 0$ is a terminal penalty independent of velocity to encourage the vehicle to stop when the pedestrian is crossing, and $1(\cdot)$ is a function that evaluates to 1 if the Boolean logic is true and 0 if it is false.

The stage cost for efficiency and mobility is given by:

$$g_{efficient}(x_t, u_t) = - \lambda v_t 1(\neg c_t) \quad \text{Equation (4.5)}$$

When the pedestrian is not crossing, a reward weight of $\lambda > 0$ is used to promote a higher speed. The goal of achieving smoothness for occupant comfort is accomplished by imposing a penalty stage cost on the change in velocity:

$$g_{smooth}(x_t, u_t) = -\xi (v_t - v_{t+1})^2 = -\xi (a_t \Delta t)^2 \quad \text{Equation (4.6)}$$

where ξ is the penalty weight for significant variations in velocity, and the cost of this stage is solely determined by the current input and the time step.

The QMDP technique is utilised to estimate an optimal solution to solve the POMDP (Sunberg and Kochenderfer 2018). Although QMDP implies that the state will be completely observable at the next time step, it is well suited to this problem since the activities are not information gathering, meaning that they do not directly lower the scenario's uncertainty. Another reason to utilise QMDP is that it is an offline solution, which means that the policy may be examined ahead of time before being deployed on a vehicle in the following step (empirical analysis). The state and action spaces are discretized in this technique to solve the POMDP, but the state transitions are kept continuous using multilinear grid interpolations (Davies 1996). The vehicle speed is increased in $0.5 m/s$ increments, the vehicle distance to the crosswalk is increased by 1m, and accelerations are measured at $0.1 m/s^2$ intervals. The sizes of the state and action spaces of the POMDP were kept modest by using this discretization: 2,563 total states (including the terminal state) and 61 potential actions. Every design option in the POMDP is linked back to a value from the conceptualization phase as a method to record and justify the engineering of this technology throughout the technical execution. The next part demonstrates how well the conception is realised by this implementation.

4.3.3 Empirical Research

The qualitative and quantitative assessments are part of the third step of the VSD approach. To understand how the VSD process affects the design of a speed control algorithm, a deterministic proportional speed control is used as a baseline. The baseline and POMDP policies are compared in a qualitative debate.

4.3.3.1 Baseline

The baseline is a deterministic proportional speed control. Once a pedestrian has been spotted, a constant deceleration is directed depending on the current vehicle velocity and distance to the crosswalk, as shown in eq. (4.7a), which also assumes a constant acceleration point mass model. If no pedestrians are spotted, the vehicle continues proportional cruise control with gain k_p and known desired velocity v_{des}, as shown in eq (4.7b). These reasoning is as shown below:

If c_t

$$a_t = - {v_t^2}/{2d_t} \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.7a)}$$

else

$$a_t = k_p(v_{des} - v_t) \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.7b)}$$

Due to the small number of design choices explored in this baseline implementation, the baseline is purposefully basic to enable for analysis of design features.

4.3.3.2 Comparison of Policies:

Figure 4.2 depicts the baseline controller (4.7), which is a closed-loop strategy that maps every condition to an action. The vehicle speed is represented on the horizontal axis, while the distance to the crosswalk is represented on the vertical axis. The activity is indicated by the colours. Pedestrian detection cuts the policy amount in half. While a pedestrian is identified, the baseline policy appears to be safe, as shown by the vehicle coasting (zero acceleration) when it is further away from the crossing and increasing brake orders as it draws closer. However, while a pedestrian is crossing, this aspect of the regulation is ineffective. When the pedestrian is not recognised, on the other hand, it is efficient but not safe. Because of the uncertainty of a pedestrian crossing, this dichotomous behaviour shows the necessity to anticipate moving from one set of logic to the other. For the situation, the basic policy does not explicitly resolve the value conflict between safety and efficiency.

The POMDP's closed-loop policy, which was created during the technical implementation phase, is represented as a set of alpha vectors, each of which corresponds to a different action. Figure 4.3 depicts the associated action based on the optimal anticipated utility for each state. The policy recommends a compromise between efficiency further away from the crosswalk and safety as the vehicle approaches while the pedestrian is crossing. While the pedestrian is

not crossing, there is a substantial gain in terms of safety while retaining some efficiency. The policy similarly suggests that measures will be carried out smoothly across the state area.

Table 4.2 summarises the weights used in the reward function for this strategy. These weights were selected during simulation to create great combination of positive accelerations away from the crossing, negative accelerations near to it, and were fine-tuned to resolve the value tension between the pedestrian and the vehicle, as shown in the following section. With these weights, the term for continuous deceleration in g_{safe} yields a penalty of -2.5 when $v_t = 10m/s$, $d_t = 0m$, and c_t, whereas $g_{efficient}$ is 2.5 when $\neg c_t$. The extra penalty η in g_{safe} implies that in this first implementation, safety and legality are emphasised. The buffer's numerical value was set such that the numerator does not evaluate to zero and the size of the constant deceleration term is limited. Finally, ξ is used to generate smooth acceleration commands.

The objective of this initial iteration is to figure out how to deal with value conflicts in the implementation. and for safety (ζ) and legality (η), efficiency (λ), and smoothness (ξ) are the special weights that are closely linked to human values. If the general design appears to be good, exact benefits can be fine-tuned using a Pareto, or multi-objective, optimization over these weights to better decide the value trade-off to make. This is shown in section 4.5. Though, in this situation, further weight adjustment is postponed until additional study indicates that the design is suitable.

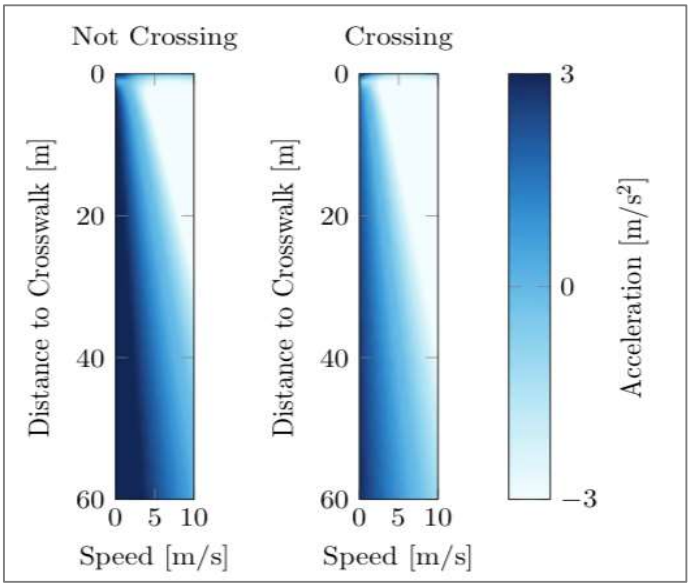

Figure 4.2: Baseline closed-loop policy mapping each state to an action.

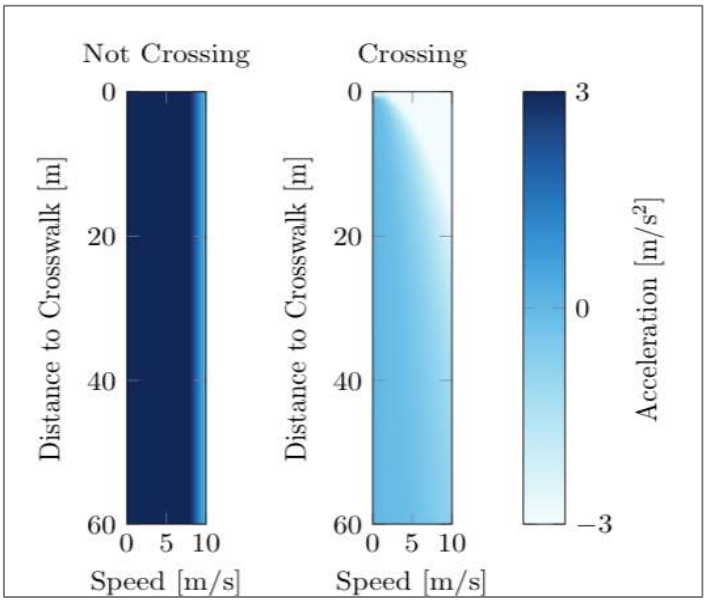

Figure 4.3: Closed-loop policy depicting optimal action at that state assuming perfect state information.

Table 4.2: Weights of the reward function

Variable	Weight	Unit
Safety and legality (ζ)	0.2	s^2/m
Safety and legality (η)	0.2	–
Buffer (ϵ)	8	m
Efficiency (λ)	0.25	s^2/m
Smoothness (ξ)	1	s^2/m

4.3.3.3 Simulation Results

To demonstrate how effectively the VSD speed controller realises human values. Using a deterministic model predictive steering control, the vehicle is tasked with following an obstacle-free course around the occluding vehicle by simulation as described in Chapter 3. Observations of vehicle speed, vehicle distance to crosswalk, and detection of pedestrians are utilised to update the belief with a Bayesian filter for the POMDP policy execution. The approximate best course of action is then taken as in eq. 4.8

$$argmax \propto_a^T b \quad \text{Equation (4.8)}$$

where $\propto_a$ is an alpha vector for each action a and b is a vector representing the belief state. The POMDPs.jl package is used for both the policy solver and policy execution.

On a two-lane highway, there is an obstructed pedestrian crossing in the simulation scenario. The car begins at the road's beginning, where it is halted. The person appears in the crosswalk from behind the occluding car as the vehicle approaches the crosswalk. The control algorithms have no idea when the pedestrian is going to emerge. The overhead driven trajectory, acceleration instructions, and speed profile for the baseline and POMDP policies are depicted in figures 4.4 and 4.5, respectively. The circles in figures 4.4 and 4.5 show when the intensity filter identified the pedestrian. The initial time step after detection in both techniques mandates a significant slowdown. The baseline control is unable to lawfully yield to the pedestrian since it is travelling at full speed when the pedestrian emerges. The POMDP strategy, on the other hand, causes the vehicle to descend significantly sooner and achieve a lower maximum speed. As a result, the autonomous agent can successfully halt the unexpected pedestrian.

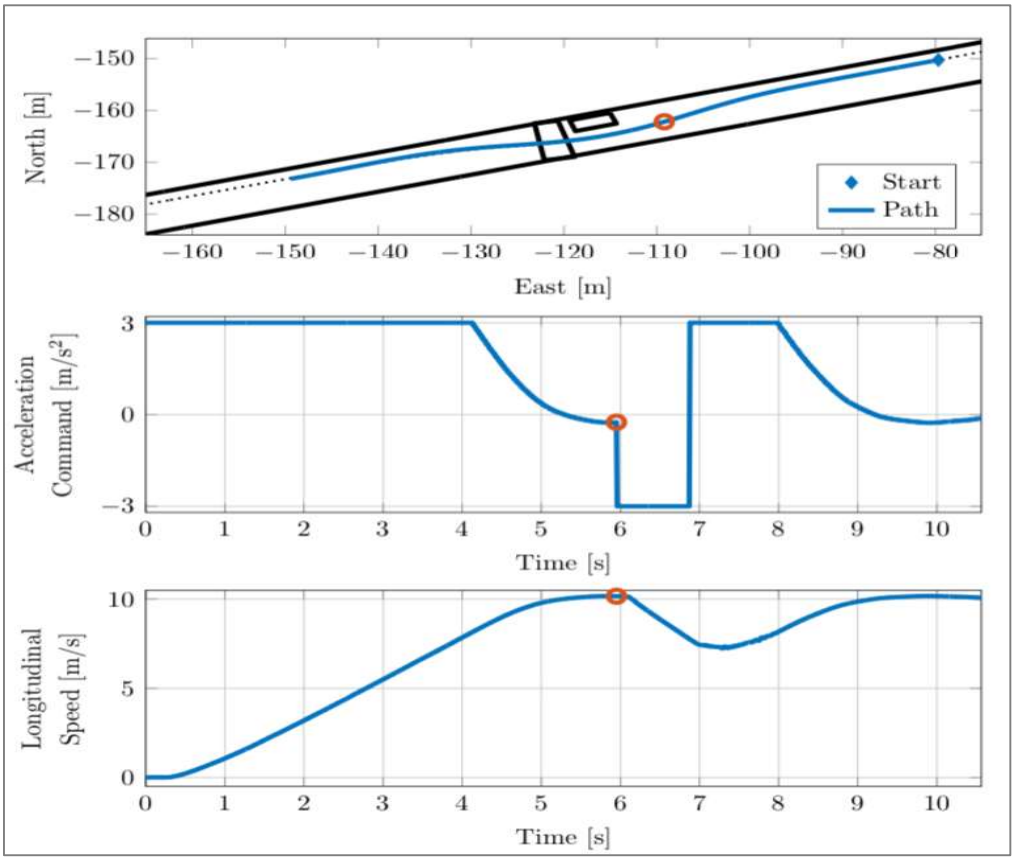

Figure 4.4: Baseline trajectory overhead, acceleration command, and speed profile using deterministic speed control (circle indicates when the pedestrian was detected). The vehicle decelerates upon detection of the pedestrian but does not yield.

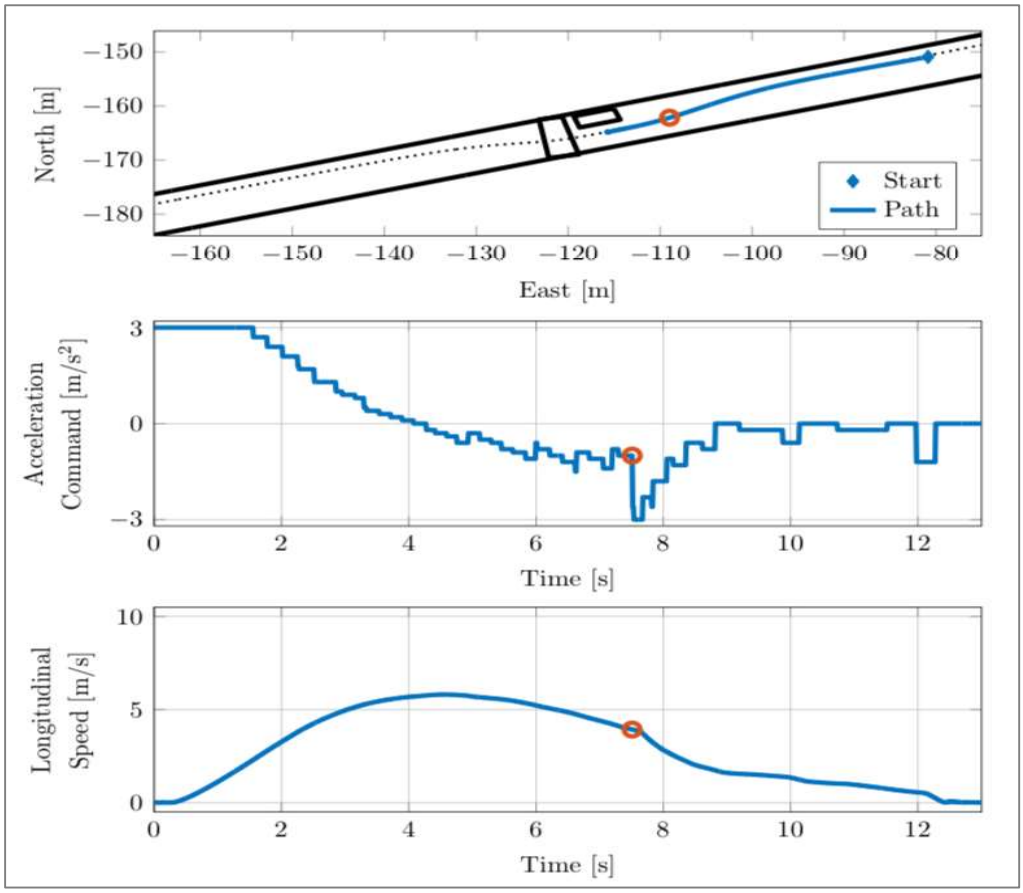

Figure 4.5: POMDP trajectory overhead, acceleration command, and speed profile using belief about pedestrian detection (circle indicates when the pedestrian was detected).

4.3.4 Observation

The challenge of creating an algorithm in the face of conflicting values is seen in this initial iteration of the speed control concept. There are several aspects of the implementation that should be highlighted, as well as other areas that might be improved.

4.3.4.1 Successful Outcomes

- ✓ The self-driving vehicle was able to yield to the pedestrian after accounting for the pedestrian's uncertainty. Because the POMDP predicted future state information, the car approaching the crossing at an "acceptable pace" had a significant impact.
- ✓ The only information on the pedestrian that was used was whether he or she had been discovered. Fairness and reciprocity were mostly preserved, but they should be made more apparent.

- ✓ The tension between safety, legality, and efficiency may be handled with the right weights.
- ✓ The choice to represent the problem as a POMDP and solve for an offline policy aided in the investigation and balancing of some of the design task's value tensions.

4.3.4.2 Improvements to be made

- ✓ Remove the braking authority restriction, which leads to a priority of occupant comfort over safety.
- ✓ Although the POMDP formulation was created with passenger communication in mind, it only optimised for velocity smoothness and ignored the shock that car occupants perceive because of irregular acceleration orders. Smoothness appears to be accomplished while just considering closed-loop rules, however it may not have been effectively accounted for with this first iteration.
- ✓ This situation does not apply to non-obstructed crosswalks.
- ✓ The value tension relies heavily on pedestrian modelling, thus additional attention is required there

The next version will look at ways to keep these great qualities while also resolving some of the implementation's drawbacks.

4.4 Second Iteration

The value-sensitive design iterative approach may be used to re-evaluate the design work as well as identify ways to enhance the technical implementation. The scenario is changed in the second iteration to emphasise pedestrian behaviour unpredictability. The design job can explore how pedestrian intent influences self-driving vehicle behaviour and vice versa by removing the occluding vehicle from the equation. The presence of an occluding car obfuscates the pedestrian-vehicle interaction. As a result, the occlusion is eliminated, and a pedestrian is positioned on the roadside, as illustrated in figure 4.6. The designer should develop an improved familiarity of the pedestrian-vehicle interaction as the iterations proceed. The occlusion might then be re-introduced into the design task or utilised as a test case during the analysis phase.

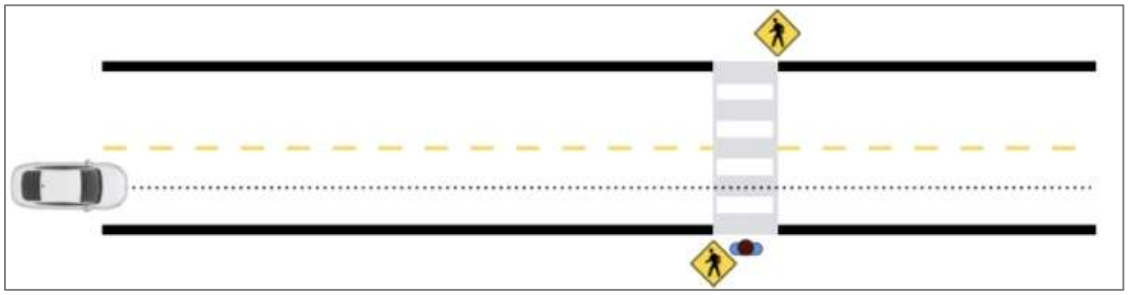

Figure 4.6:The scenario of the pedestrian crosswalk with occlusion is eliminated.

4.4.1 Conceptualization

Various stakeholders are still involved in the design process, which touches on a wide range of human values. The autonomous vehicle's occupants, pedestrians who may cross the street, and the authority of traffic regulations are now the direct stakeholders. The values in question in the scenario are mobility, safety, legality, care and respect for others, fairness and reciprocity, respect for authority, trust and transparency, and individual liberty, even with the occluding vehicle removed. The values are defined in the same way as in section 4.3.1, but the way they are translated into technical specifications will be improved.

Only human values connected to a technical objective were expressly evaluated in the previous phase. This iteration clarifies how each defined value should be represented in the technology. The Fairness and reciprocity are not easily translated into technical objective. Instead, it becomes a higher-level design constraint that restricts the information to be non-discriminatory, such as no age or gender data.

The remaining values are handled by connecting them to technical specifications that may be used to capture them as shown in table 4.3 below.

4.4.1.1 Respect for Authority and Legality

Safety and legality are not precisely the same criteria, according to the California Vehicle Code. To be safe, the vehicle code merely requires drivers to use "due care," which is not the same as being safe. The vehicle code also requires that the vehicle speed be reduced and that any required steps be taken to ensure pedestrian safety. Vehicle speed (v_t), vehicle distance to crossing (d_t), and pedestrian conduct are the most important pieces of information for making legal decisions. The self-driving vehicle must know if the pedestrian is transitioning from the

sidewalk to the crosswalk to protect the pedestrian. The non-discriminatory pedestrian position (c_t) and pedestrian posture (p_t) are believed to reflect pedestrian behaviour.

4.4.1.2 Care, Safety, and Respect for others

A stricter interpretation of the vehicle code reflects the importance of safety. The goal of safety is to minimise danger and injury. The same data as for legality is required to obtain this value: vehicle speed (v_t), vehicle distance to crosswalk (d_t), pedestrian position (c_t), and pedestrian posture (p_t).

4.4.1.3 Efficiency and mobility

The value of mobility still captures the measure of time efficiency, which is exactly proportional to the vehicle's speed (v_t) on a straight road.

4.4.1.4 Smoothness and Mobility

Smooth driving is another component of mobility that impacts occupant comfort and fosters trust and sincerity among stakeholders. The value of mobility is meant to be increased in this iteration by employing both the previous acceleration command (a_{t-1}) and the present acceleration instruction (a_t) for smooth actions adjustments.

4.4.2 Technical Implementation of Technology

An additional iteration gives you the option of using a different approach or algorithm that is more in line with the stated values. The POMDP is maintained in this iteration because it appears to provide possible resolution, and it helped to expose value conflicts in the prior iteration. The best policy for controlling the longitudinal acceleration of the vehicle based on the belief of a pedestrian crossing is computed using dynamic programming once more. Equation 4.9 below captures the information needed to address their respective values in the objective function given the technical specification of legality, safety, and mobility.

Table 4.3: A summary of human values mapped to engineering specifications for the second VSD iteration.

Human value	Engineering specification	Representation
Fairness reciprocity	Do not use discriminatory information	
Legality Respect for authority	Legality	v_t d_t c_t p_t
Safety Care and respect for authority	Safety	v_t d_t c_t p_t
Mobility Individual autonomy Trust transparency	Mobility	v_t a_{t-1} a_t

$$x = [v_t \quad d_t \quad c_t \; p_t \; a_{t-1}]^T \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.9)}$$

and the control input,

$$u_t = a_t \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (4.10)}$$

Where v_t denotes vehicle speed, d_t denotes vehicle distance from crosswalk, c_t denotes pedestrian location, p_t denotes pedestrian posture, and a_{t-1} and a_t denote prior and present longitudinal acceleration, respectively. Because the roadway's peak speed is 10m/s, the vehicle's top speed is constrained by the speed restriction to meet both legality and safety requirements. The pedestrian is either in a crosswalk or on the sidewalk, and his or her posture is either halted, inattentive, or in motion while making eye contact with the car. The pedestrian states do not rely on other potentially discriminating information about pedestrians to maintain the ideals of fairness and reciprocity. The control input was formerly limited to 3m/s^2 to give pleasant acceleration levels; however, this hampered the vehicle's capacity to be safe. Allowing deceleration up to 10m/s^2, the control algorithm allows the vehicle to utilise its maximum braking capability.

To compute the distance to the crossing and vehicle speed, the dynamics (or state transitions) still rely on a point mass model of the vehicle. To further examine the value tensions for the

design task, a new model for pedestrians is built (table 4.4). The chance of a pedestrian moving from the sidewalk to a crosswalk is determined by their posture. When a pedestrian is preoccupied, the chance is 50%, and when the pedestrian is moving, the likelihood is 86.7% (Schroeder, Rouphail et al. 2014) and the statistics on yield and non-yield occurrences for an aggressive pedestrian at site B are used to compute the likelihood of 86.7%. The likelihood of transitioning is a function of the vehicle's distance from the crossing while the pedestrian is stopped while making eye contact with it.

$$p_r(c_t|\neg c_t;\ p_t = STOPPED) = (p_{xing}/d_{max})\ d_t, \quad \text{Equation (4.11)}$$

where d_{max} is the highest distance the vehicle is away from the crossing, and p_{xing} is the probability of 52.34% (Schroeder, Rouphail et al. 2014) and for a pedestrian waiting on the near side at site B, the data on yield and non-yield occurrences yielded a chance of 52.35%. The pedestrian is expected to remain in the crosswalk for the next time step once within the crosswalk. For the sake of simplicity, the control loop assumes complete knowledge on the vehicle's distance from the crossing, vehicle speed, and pedestrian posture. The pedestrian position, however, there is observation uncertainty for the pedestrian position, which captures sensor uncertainty with a false positive rate of 5%. These false positive rates were selected arbitrarily low but would derive from the perception system's capacity to recognise pedestrians in practice.

The aim remains for the self-driving vehicle to travel across the crosswalk in a seamless, safe, and efficient manner while complying to all applicable traffic regulations. For each state and action, the reward function defines the stage cost g (x_t, u_t), which links the conception values to the technological implementation once more. The reward for a state action pair is calculated by adding the state and action's stage costs (4.12), (4.13), and (4.16).

Table 4.4: The pedestrian transition model for the second VSD iteration

Pedestrian posture	Transition probability $p_r(c_t/\neg c_t)$
Distracted	0.5
Stopped	0.523^{+} (d_t/d_{max})
Moving	0.867^{+}

$^{+}$*computed using yield event statistics (Schroeder et al. 2011)*

The following is the stage cost for legality, which is derived from the constant acceleration point mass formulae related to the constant deceleration required to come to a complete stop given the distance to the crosswalk and vehicle speed:

$$g_{legality}(x_t, u_t) = -\zeta \frac{v_t^2}{d_t + \varepsilon} 1(c_t) \quad \text{Equation (4.12)}$$

where $\zeta > 0$ is a weight on the penalty suffered by driving rapidly as the vehicle approaches the crossing, and $\varepsilon > 0$ is a buffer in the denominator to ease the restriction.
the stage cost for safety is given by.

$$g_{safety}(x_t, u_t) = -\eta 1(c_t \wedge d_t < 0) \quad \text{Equation (4.13)}$$

where $\eta > 0$ is a terminal penalty independent of the velocity to boost the vehicle to come to a complete stop while a pedestrian is crossing.
The stage cost for mobility is divided into two components:

$$g_{efficient}(x_t, u_t) = \lambda v_t 1(\neg c_t) \quad \text{Equation (4.14)}$$

and

$$g_{smooth}(x_t, u_t) = -\xi (a_{t-1} - a_t)^2 \quad \text{Equation (4.15)}$$

As a result, the total cost of mobility at each stage is:

$$g_{mobility}(x_t,u_t) = g_{efficient}(x_t,u_t) + g_{smo} \quad (x_t,u_t) = \lambda v_t 1(\neg c_t) - \xi (a_{t-1} - a_t)^2 \quad \text{Equation (4.16)}$$

where $\lambda > 0$ is a reward weight to boost faster speed while the pedestrian is not crossing, and $\xi > 0$ is a penalty for excessive acceleration changes.

The QMDP technique is utilised once more to estimate an optimum solution for the POMDP. Vehicle speed is increased by $0.5m/s$, vehicle distance to crosswalk is increased by 1m, and accelerations are measured at $0.5m/s^2$ intervals in this iteration. The size of the state and action spaces of the POMDP was kept minimal by using these discretisation's: 142,884 total states (including terminal states) and 27 potential actions.

4.4.3 Empirical Research

The simulation results are the subject of the empirical analysis in the second iteration. The state space of this POMDP, unlike the baseline, cannot be completely described in three dimensions,

therefore a policy comparison is not included in this study. Because the POMDP policies are based on the prior acceleration instruction.

4.4.3.1 Simulation results

In this simulation, pedestrian detection is used to identify whether the pedestrian is in the crosswalk influence area (the sidewalk) or in the crosswalk, using the specification of a static polygon for the form of the road. The self-driving car must follow a straight line down the road, using the same deterministic predictive steering control paradigm as Brown et al. in Chapter 3. A pedestrian crossing on a two-lane highway is used in the simulation scenario. When a pedestrian reaches the crosswalk influence area, the vehicle is travelling at a high speed, and the policy comes into play. The pedestrian may or may not shift into the crosswalk as the car approaches the intersection. The pedestrian's transition is unknown to the control algorithms. For comparison with the new POMDP policies, the baseline from the first iteration is employed once more. Because it will not surrender to the pedestrian until he or she has reached the crosswalk, it is termed an aggressive baseline. A cautious baseline, in which the vehicle begins to yield to the pedestrian once he or she reaches the crossing influence area, is also explored as an option. Except for the crosswalk influence area, which decides when to transition from cruise control to brakes, the rules are identical. They do not, however, take into consideration the position of the pedestrian. For the aggressive and cautious baselines, figures 4.7 and 4.8 show the overhead driven trajectory, acceleration instructions, and speed profile, respectively. The circles represent the times when the computer vision system recognised the pedestrian in the crosswalk. The person never entered the crosswalk since the aggressive baseline has no circle. Because the pedestrian did not enter the crosswalk, the car continued to travel at the speed limit, never yielding to him. When the vehicle was $12.99m$ away from the crossing, the perception system identified the pedestrian in the crosswalk influence region, and the vehicle yielded to the pedestrian effectively.

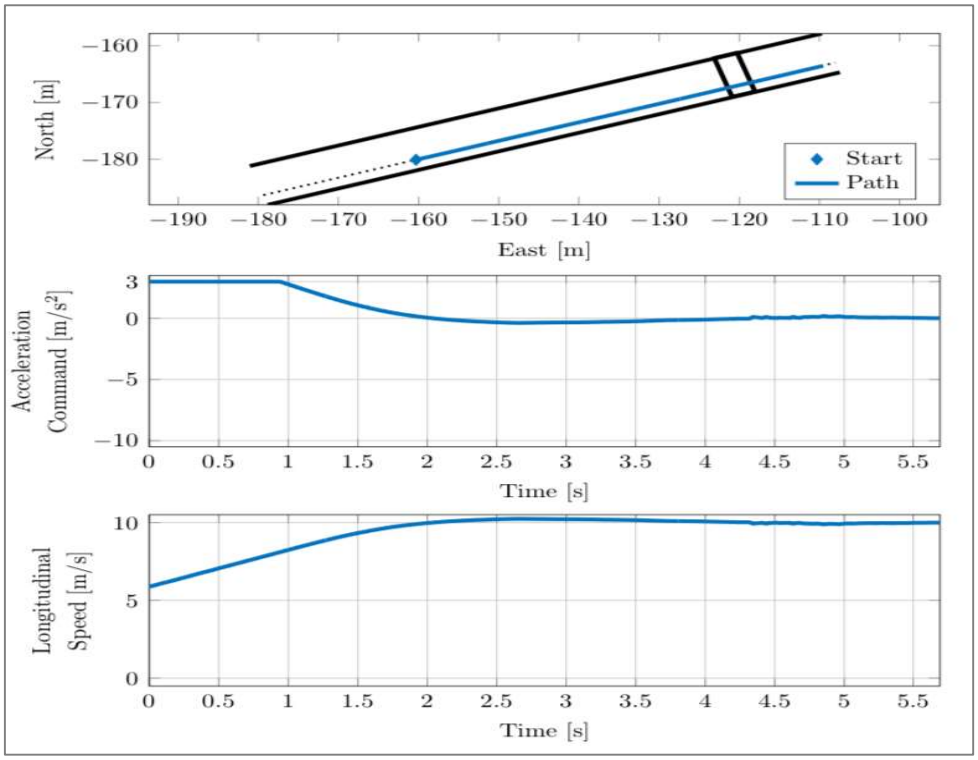

Figure 4.7: Deterministic speed control is used to provide an aggressive baseline trajectory overhead, acceleration command, and speed profile. The pedestrian does not enter the crosswalk, thus there is no red circle.

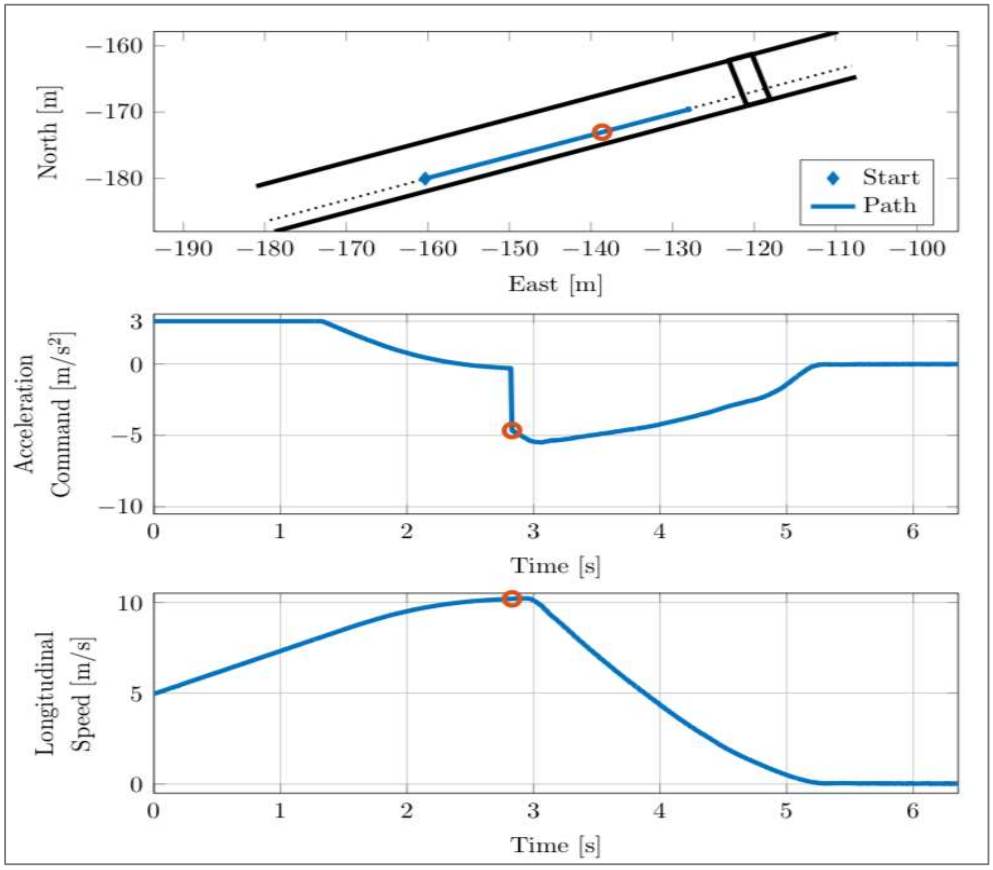

Figure 4.8: Overhead trajectory with a conservative baseline, acceleration instruction, and speed profile with deterministic speed control (circle shows when the pedestrian was spotted).

To update the belief using a Bayesian filter for the POMDP policy execution, an observation of the vehicle speed, vehicle distance to the crosswalk, pedestrian posture, and pedestrian position is utilised, much as the previous iteration. The overhead driven trajectory, acceleration instructions, and speed profile for POMDP policies are depicted in figures 4.9 to 4.12. The circles represent the times when the computer vision system recognised the pedestrian in the crosswalk.

The pedestrian postures are independent of each other in this second POMDP implementation. As a result, the reward function for each position employs a distinct set of weights (table 4.5). This makes sense since each pedestrian stance is a discrete scenario that necessitates a different vehicle reaction. The buffer's numerical value is still arbitrary; it's set such that the numerator doesn't evaluate to zero and the constant deceleration term's magnitude is kept to a minimum. The remaining weights can be fine-tuned with more analysis and Pareto optimization (see section 4.5), but they're picked first to test if this design is suitable.

Table 4.5: The reward function's weights in relation to pedestrian posture (p_t)

Variable	Distracted weight (p_t)	Walking weight (p_t)	Stopped weight (p_t)	Unit
Legality (ζ)	0.01	0	0.01	s^2/m
Buffer	8	8	8	m
Safety (η)	0.5	0.5	0.5	-
Mobility (λ)	0.05	0.01	0.03	s/m
Mobility (ξ)	0.003	0.01	0.003	s^2/m

At the extreme states and actions where $v_t = 10m/s$, $d_t = 0m$, and $a_{t-1} - a_t = 13\,m/s^2$, the weights are set so that safety, efficiency, and smoothness are prioritised to identical normalised values: $\eta_n = 0.5$, $\lambda_n = 0.5$, and $\xi_n = 0.507$. When $v_t = 10m/s$ and $d_t = 0m$, the legality term is normalised to $\zeta_n = 0.125$, implying a lesser priority. Figure 4.9 shows that when a pedestrian approaches the crosswalk influence region, the policy applies modest negative accelerations to the vehicle, slowing it to about $1.5m/s$ and allowing it to coast until the pedestrian enters the crosswalk. The vehicle comes to a complete stop after the pedestrian has entered the crosswalk. The parameters for efficiency and smoothness raise to normalised values of $\lambda_n = 1$ and $\xi_n = 1.69$ when the pedestrian is walking as shown in figure 4.10. With more efficiency, the vehicle

travels along the road at a quicker pace, necessitating more smoothness to smoothly slow the vehicle from the faster speed if a pedestrian reaches the crosswalk. Because the pedestrian transitions to the crosswalk with a high likelihood, the pedestrian has a high conviction of 0.36 that he or she is crossing. This indicates that the impact of the safety and legality parameters has a significant impact on self-driving vehicle behaviour long before the pedestrian enters the crosswalk physically. To decrease the huge influence of the safety and legality terms, one of them (legality in this case) is set to zero to enable the car to approach the crossing. The car coasts again with these weights until it detects the pedestrian. The car comes to a complete stop as it approaches the crossing.

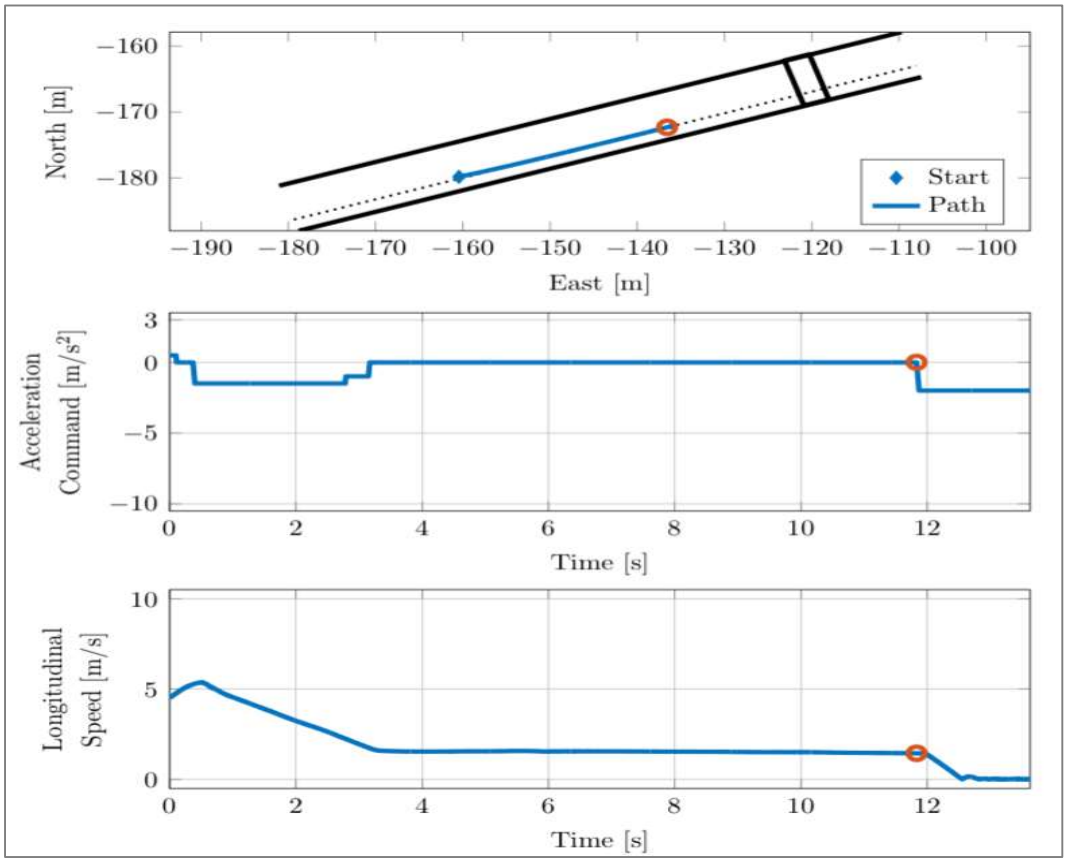

Figure 4.9: Overhead POMDP trajectory of a distracted pedestrian, acceleration command, and speed profile based on pedestrian crossing belief (circle indicates when the pedestrian was detected).

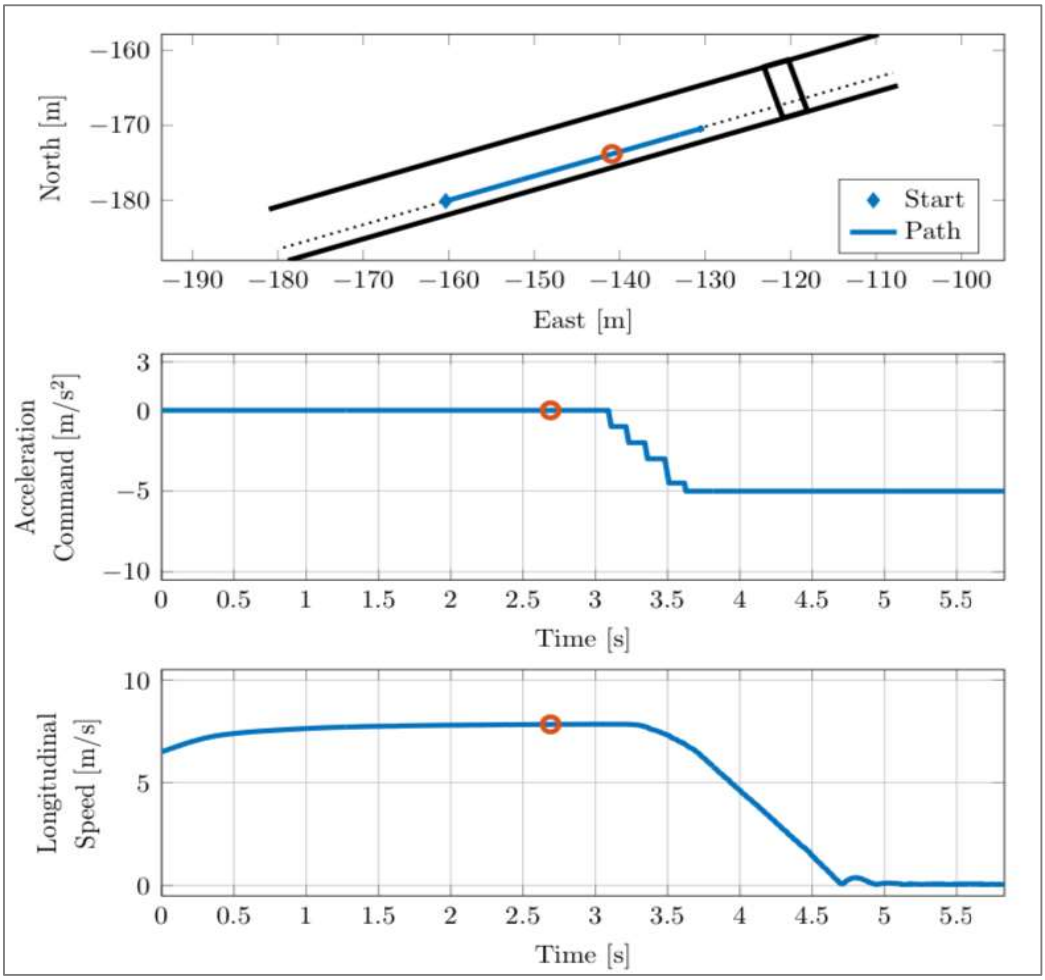

Figure 4.10: Overhead walking pedestrian POMDP trajectory, acceleration command, and speed profile based on pedestrian crossing belief (circle shows when the pedestrian was spotted).

The normalised legality term returns to ζ_n = 0.125 for the halted pedestrian, but efficiency and smoothness fall to λ_n = 0.3 and ξ_n = 0.507. Smoothness is critical in this situation for the self-driving vehicle to exhibit directness about its objectives to travel through the surroundings, therefore it is given the greatest priority. Two situations were examined using these weights. Because of the diminishing assumption that the pedestrian would cross the roadway, the vehicle shown in figure 4.11 accelerates progressively as it approaches the crosswalk. The pedestrian avoids entering the crosswalk because it does not want to create an immediate hazard. Figure 4.12 shows a pedestrian crossing the roadway before the vehicle speeds up too fast in the second halted pedestrian scenario. The self-driving vehicle comes to a complete stop after the pedestrian enters the crosswalk.

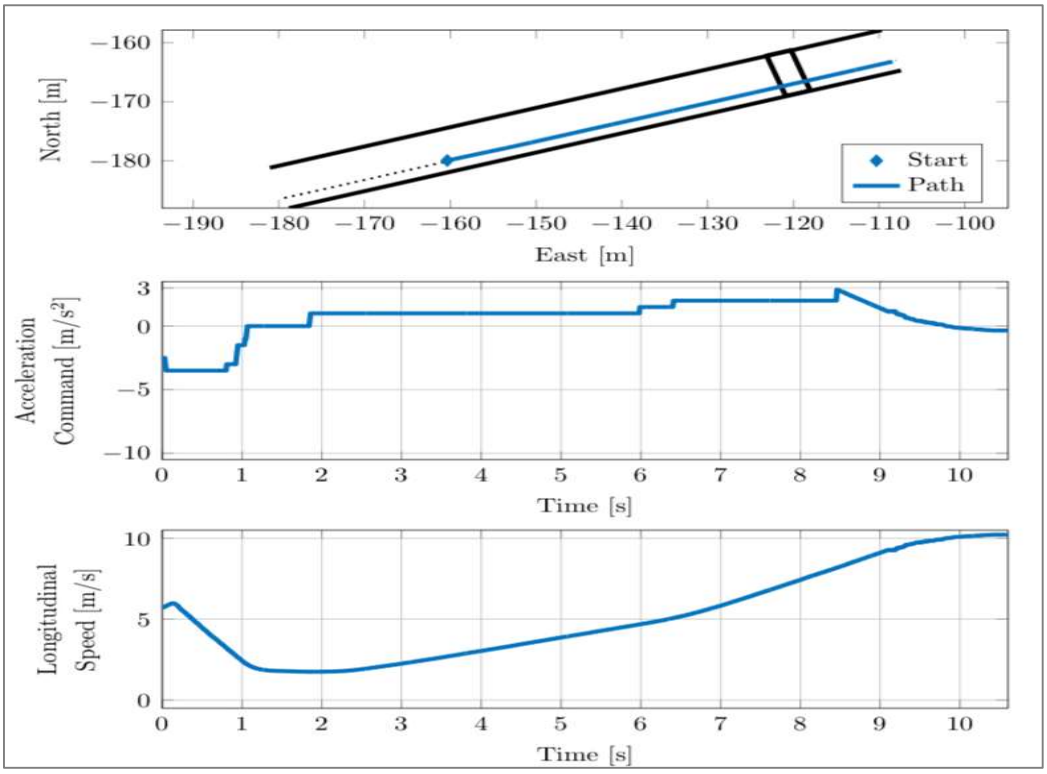

Figure 4.11: Stopped pedestrian POMDP trajectory overhead, acceleration command, and speed profile utilising pedestrian crossing belief. There is no red circle since the pedestrian does not enter the crosswalk.

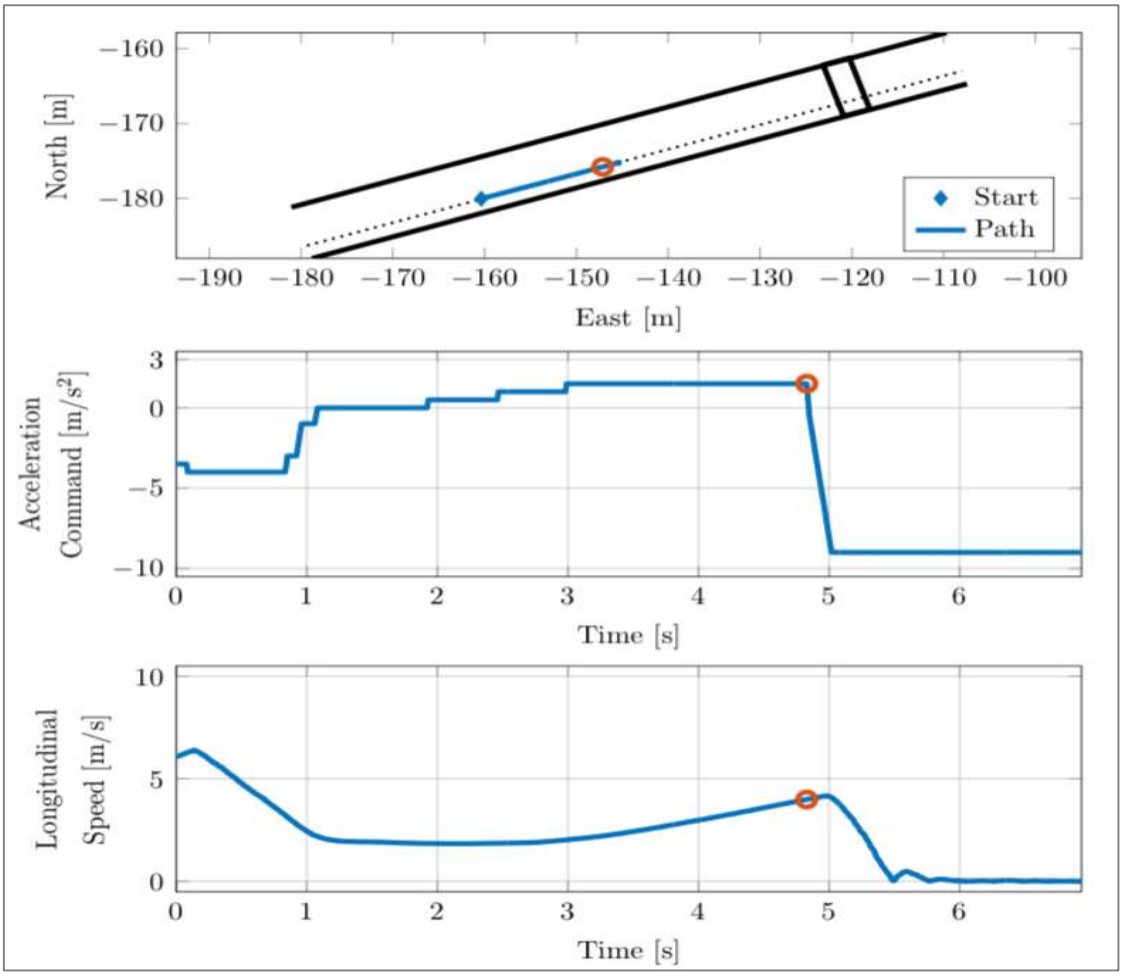

Figure 4.12: Stopped pedestrian POMDP trajectory overhead, acceleration command, and speed profile utilising pedestrian crossing belief (circle shows when the pedestrian was spotted). Pedestrians have the right of way.

The situation automatically becomes more complicated around the value tension between the pedestrian's purpose and the self-driving vehicle's desire to drive along the road with this second iteration's focus on pedestrian behaviour. Over the value statements, the weights chosen are still arbitrary trade-offs. As a result, a further in-depth examination is likely necessary at this stage to evaluate whether a certain design point may address the value tensions. A Pareto optimization, for example, may be used to simulate many scenarios when weights are changed as shown in section 4.5.

4.4.4 Observation

As the technical specifications clarify in terms of the recognised values, this second iteration of the speed control design shows advances in resolving value conflicts. Because this is not the final product, there are still certain aspects of the implementation to emphasise and other things to improve.

4.4.4.1 Successful Outcomes

- ✓ The car was able to successfully yield to the pedestrian in all circumstances by accounting for pedestrian uncertainty. Dynamic programming may be utilised to account for future state information in the policy since the problem was designed as a POMDP.
- ✓ The main information on the pedestrian was whether he or she was in a crosswalk and what posture he or she was in. Fairness and reciprocity were upheld because of this.
- ✓ The choice to represent the problem as a POMDP and solve for an offline policy was maintained throughout the design process, which helped to examine and balance some of the design task's value conflicts. Even though the weights were chosen at random, it revealed the possibility of resolving the value tensions.
- ✓ The penalty on change in acceleration increased smoothness, and the effect was directly proportional to ξ.
- ✓ Efficiency remained consistent with the term λ.
- ✓ The pedestrian was modelled as a function of posture, which revealed information about pedestrian intent and crossing the roadway.
    - ➢ For the halted pedestrian, the car gradually raised its speed as it neared the crossing, signalling to the pedestrian that if he or she enters the crosswalk while simultaneously wanting to proceed down the road, the vehicle will yield.
    - ➢ The car approached the crossing at a very cautious pace because of the random likelihood of the inattentive pedestrian.

4.4.4.2 Improvements to be made

- ✓ Pedestrian modelling must be enhanced further.
  - ➢ The pedestrian's stance and position did not appear to completely comprehend the pedestrian's desire to cross the roadway. Other factors might be examined while keeping the ideals of fairness and reciprocity in mind to reduce the vehicle's usage of biased information or discriminating behaviours.
  - ➢ For the pedestrian, there is likely to be a link between distraction and mobility, which might imply a higher chance of shifting for the distracted posture. Other pedestrian models might be used to investigate the better points of pedestrian posture and mobility.
  - ➢ When a pedestrian enters a crosswalk, the transitions presume that the person will remain within the crosswalk. This is not the case. There should be further research on simulating the transition from the crosswalk to the sidewalk (or a safe distance from the self-driving vehicle's moving lane).
- ✓ The car tends to come to a standstill just outside the crosswalk. This might be due to poor weight selection, or the reward function could be tweaked, for example, by imposing a small penalty on significant decelerations so the vehicle only comes to a complete stop when absolutely required. This will aid in the designing of mobility and efficiency specifications.
- ✓ The high possibility of transitioning in the case of the moving pedestrian enhanced the policy's effect on the parameters of safety and legality. These weights need to be reduced considerably, or a different formulation should be investigated to isolate the influence of safety and legality more effectively.
- ✓ To establish how effectively mobility, safety, and legality can be accomplished with this implementation, more research into the choice of weights in the reward function is also required. For the first iteration, this may be accomplished using a Pareto, or multi-objective, optimization over the weights, as shown in section 4.5.

4.5 Bridging the Gap on Human Values

The policy comparison and experimental results show a speed control algorithm design that is possibly reasonable, but only for a certain set of weights. The choice of weights in the reward function can have a significant impact on the vehicle's behaviour. An analytical approach is required to assess how well more directly the planned technology matches with stakeholder values. To identify which set of weights best aligns with the values, one approach to do this analysis is to use the Pareto (or multi-objective) optimization technique. If one goal cannot be improved without affecting at least one other goal, the design is Pareto optimum. The design objectives are mapped to a criterion space using evaluation criteria to create a frontier of Pareto optimum locations. The identification of Pareto optima helps to shut the loop on the design process, where human values are translated into engineering objectives, engineering objectives are translated into evaluation criteria, and evaluation criteria are translated into human values. As a result, engineers may concentrate on Pareto optimum solutions without having to commit to a certain prioritisation of objectives ahead of time. After that, the Pareto frontier may be presented to a wider group of stakeholders to select the final design to implement.

By changing the weights in the reward function that correspond to the engineering objectives, an example of a Pareto frontier for the first VSD iteration is produced. A separate optimum policy is created for each combination of weights in the reward function. Monte Carlo simulations are conducted for each specified optimal strategy, and the simulation results are compared to evaluation criteria. The vehicle velocity at the crossing is mapped to the objective of safety and legality. The criteria of average time to perform a manoeuvre corresponds to the objective of efficiency. The criteria of average maximum change in acceleration translates to the smoothness objective. When the self-driving car is within 20m of the crossing, the pedestrian appears from behind the occluding vehicle. The pedestrian takes roughly 4 seconds to cross the roadway, according to the simulations.

The resultant Pareto frontier can then be presented to a broader group of stakeholders, such as politicians, policymakers, and public interest organisations, to decide which, set of weights to use on the self-driving vehicle. A slice of the Pareto frontier for safety and legality vs. mobility is shown in figure 4.13. The yield rate for the simulated suddenly emerging pedestrian scenarios also adds colour to the picture. Additional data, such as damage curves (Kröyer, Jonsson et al. 2014), user studies, emissions curves (Mo, Li et al. 2017) and congestion studies (Soriguera, Martínez et al. 2017), can be conferred on the Pareto boundaries by a broader set of stakeholders.

The Pareto optimization isn't the sole technique for bridging the gap between human values. A risk management or cost benefit analysis for a collection of outcomes [104] may be another utilitarian-like analysis technique. Perhaps a deontological approach, in which policymakers or stakeholders establish thresholds and circumstances, would be preferable. This Pareto analysis is the first step in illustrating how proper analytical technique may aid in determining how well a technological implementation incorporates the human values specified during the conceptualization phase.

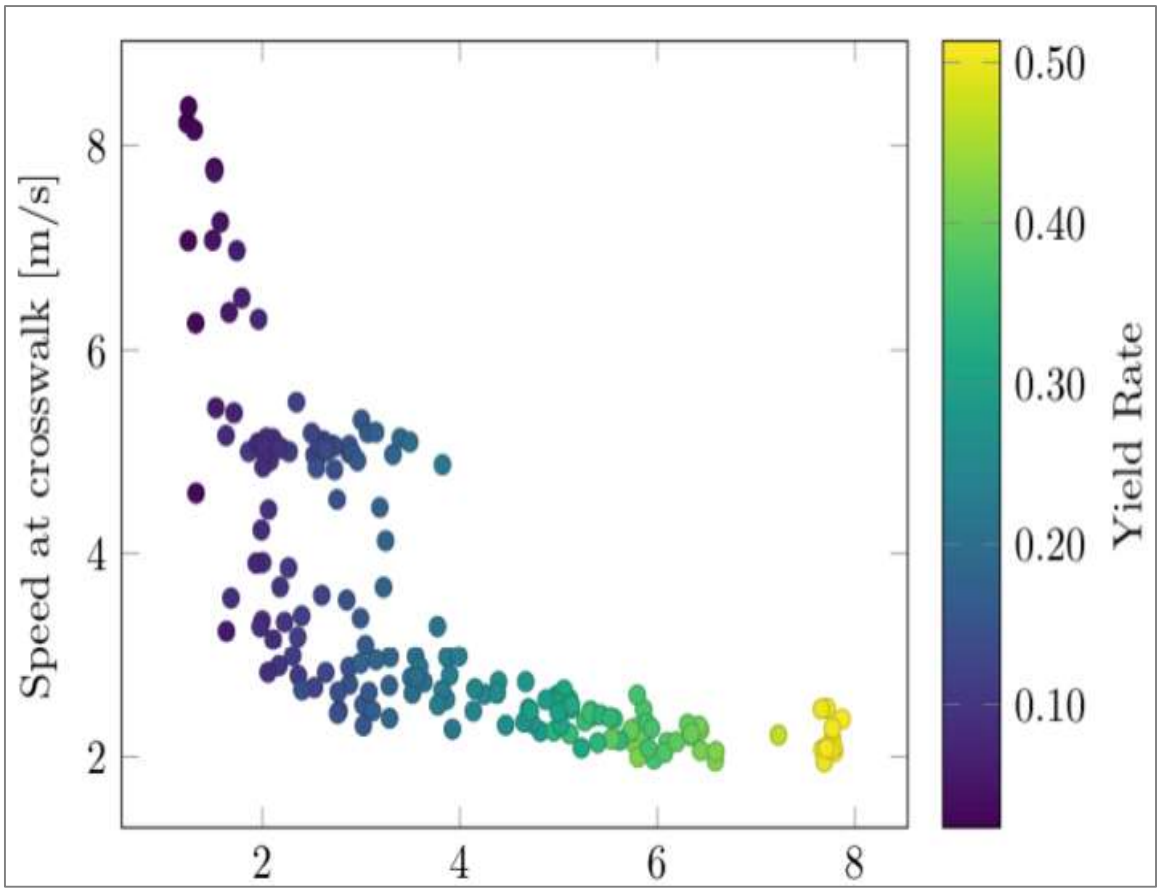

Figure 4.13: POMDP's Pareto frontier with different weights linked to evaluation criteria.

4.6 Summary

Through the conception and technical implementation phases, this chapter illustrates the formal integration of human values into the design of a speed control algorithm. The empirical analysis step aids in the identification of areas for future iterations improvement. In the first iteration, a POMDP is selected to aid in the realisation of the objectives of safety and legality, efficiency and mobility, and smoothness in a situation involving a big vehicle parked in front of a pedestrian crossing. The POMDP assisted in capturing the situation's ambiguity and enabled the vehicle to be proactive by approaching the crossing at an acceptable speed, resulting in a successful yield to an unexpected pedestrian. The second iteration examines a minor adjustment in the situation by eliminating the occlusion and improving the pedestrian model to look more carefully at the value conflict between the pedestrian and self-driving vehicle. Likewise, the technological execution of the values of legality, safety, efficiency, and smoothness were refined. Additional analysis using Pareto optimization offers more information on how well a solution match with the values indicated. Engineers may think more thoroughly about how human values are mixed up in technology as it evolves by iterating using value sensitive design technique.

Although the focus has been on engineers and programmers as designers, VSD allows other stakeholders, such as policymakers and civil society organisation, to participate in the design process. VSD is a useful tool for engineers, however, additional participants from third-party groups can be extensively included to assist guarantee that self-driving cars act in socially acceptable manners.

Chapter 5: Ethical Valence

5.1 Introduction

In the viewpoint of original equipment manufacturers, government organisations, and the public, self-driving vehicles are transitioning from a distant possibility to a near-term reality. This transition is not without threat, as recent and somewhat worrying events have demonstrated. Accidents will continue to happen even as technology advances. The ethics of a self-driving vehicles has thus quickly become a contentious topic, particularly considering the apparent diversity of moral preferences within a given society and the so-called "social dilemma" of choosing a general decisional maxim, even one as benevolent as "minimise casualties" (Hulse, Xie et al. 2018). While the dangers, weaknesses, and problems associated with the transition to a self-driving have been highlighted, a viable, implementable solution has yet to be discovered. What actions must be taken to ensure that the social advantages promised by self-driving cars are realised?

Undoubtedly, the self-driving vehicle’s decision-making is an essential element of the response. The ability of a self-driving vehicles to eliminate human error from the stream of traffic situation is a significant premise and universal discussion in the self-driving vehicle deliberation: no more drunk-driving, texting, sleeping, or otherwise distracted drivers on the road. Furthermore, in inevitable collision scenarios, the self-driving vehicles is expected to make a deliberate decision about how it would crash, thereby replacing human drivers' inefficient and illogical reactions (Martinez, Heucke et al. 2017). This is a difficult task for any artificial decision process, not to mention the one that operates in a complex, dynamic, and unpredictable environment like present-day transportation system. Despite these obstacles, a workable solution for successful and acceptable self-driving vehicle decision making should be discovered. In this chapter, the Ethical Valence is suggested as a strategy for a self-driven vehicle decision making. The Ethical Valence concept describes a self-driving vehicle decision-making as a technique for claim mitigation, in which various road users hold diverse moral claims on the vehicle's conduct, and the vehicle must neutralise these claims as it makes judgments about its surroundings. It must identify an optimum response to these claims in the event of an unavoidable collision, or in so-called "dilemma situations," one that reflects the moral claims and relationships that exist inside the vehicle's decision environment and best fits with user expectations.

5.2 Risk Mitigation in Self-Driving vehicles

Volatility, unpredictability, cultural relativism, and, in certain circumstances, fatality characterise the human trafficking situation. In fact, the number of persons killed or injured on the world's roads remains unacceptably high, with an estimated 1.35 million deaths and up to 50 million injuries each year (Organization 2018). As a result, many stakeholders, institutions, and drivers have hailed autonomous vehicles (AVs) as the next, if not last, step toward accident-free roads. Even the most optimistic long-term predictions of the impact of self-driving cars expect a 90% reduction in traffic-related incidents (Taiebat, Brown et al. 2018, Martínez-Díaz, Soriguera et al. 2019). While this figure is unparalleled and remarkable, the fact remains that fatal, serious, and near-accidents will continue to occur, although less regularly, once autonomous vehicles are on the road, particularly in the early stages of implementation when mixed-feet traffic forces autonomous vehicles to interact with human drivers. Given these projections, it seems irresponsible to think of autonomous cars as merely harmless road users. As self-driving vehicles deployment progresses, physical, if not fatal, damage will continue to be a component of the traffic situation.

As a result of the ongoing existence of damage in mixed fleet traffic situations, the literature has produced two related reactions. First, these dilemma-type decision situations have been compared to the "trolley problem" (McGuire, Langdon et al. 2009, Christensen and Gomila 2012), and second, there has been a plethora of dilemma scenario analysis as to the form of decisional or ethics policies that would encompass how morality requires the AV to act in these types of dilemmas. Obviously, the ideal ethical policy is a complex and open topic that involves numerous meta-ethical and interdisciplinary concerns. Because it appears that any robust decision about the moral content that underpins an ethical policy is implicitly supported by several meta-considerations, such as the source of moral content, whether public and participatory (Arntz, Gregory et al. 2016, Schulz and Dankert 2016), or traditional western ethical paradigms such as utilitarianism (Conway, Goldstein-Greenwood et al. 2018), Rawlsian theories of justice (Holden, Linnerud et al. 2017). The apparent absence of ground-truth ethical principles across various civilizations and the failure of user expectations to closely correspond with any pre-existing moral theory likely to worsen these arguments (Yilmaz and Saribay 2017). Lastly, there are important questions of using the trolley problem as a persuasive policy instrument in the case of self-driving vehicle (König and Neumayr 2017, Kaur and Rampersad 2018), and whether these individual decision-cases cannot be managed through the extensive ethical analysis of a self-driving vehicles as a disruptive technology (Epting 2019).

Consequently, expert discussion has spun an extremely immobilising web around cars whose wheels have already touched public roads in only a few short years, while the topic of the optimal ethical policy remains conspicuously open.

Nevertheless, certain aspects of an ideal ethical policy emerge in this setting of moral ambiguity (Allam and Dhunny 2019). If we accept the assertion that widespread usage and adoption of self-driving cars is a precondition for the many social advantages these vehicles are said to bring (Dechesne, Dignum et al. 2019), then any fair ethical policy for a self-driving vehicle cannot ignore public acceptability. It appears that, for a self-driving car to really become a morally optimum means of transportation, their behaviour must be in line with the varied expectations of the people with whom they interact, as well as the broader communities in which they are deployed. In a nutshell, this constraint ensures the user's happiness and safety, as well as other important design ideals like trust, responsibility, and transparency (Perera, Hussain et al. 2019). However, in terms of the vehicle's ethical policy, this claim appears to provide either a strong reason to prefer moral theories that do not revise what is commonly referred to as "common sense morality," or, less strongly, a reason to reject any account of the moral good that fails to adequately capture widely held moral attitudes. Many popular theories of morality, including most kinds of deontological ethics, utilitarianism, and Rawlsian contractarianism, look to be in danger if this is accurate; and even if they survive, few will have kept the purity of their original structure, motives, and scope. Moral theory appears to be at least beholden to, if not restricted by, prevalent moral views from this standpoint, within the context of self-driving vehicle ethics.

On the other hand, it appears to be just as ethically problematic to ignore moral theory entirely, relying solely on the (moral) "wisdom of the crowd" (Dignum 2019, Tronto 2020). Much of the support for this latter claim comes from a concern about the algorithmic consequences of human moral failure: whether it's due to bias, prejudice, ignorance, irrationality, or plain egoism, human behaviour provides (training) data that is at best morally sub-optimal, and at worst morally unacceptable (Dignum 2019). Some argue that artificial agents should not just avoid replicating these sorts of behaviours in their interactions, but that if they are intended to behave as pure moral reasoners, they may even provide a chance for human moral growth (Cushman, Young et al. 2006, Graham, Nosek et al. 2011, Lind and Wakenhut 2017). In this case, despite their mismatch with user expectations and the introduction of what Lind has dubbed "ethically superior robot villains" into our daily lives, ethics regulations based on pure descriptions of moral theory may be morally justifiable or even needed (Wong 2020). Then,

from this vantage point, public acceptance appears to be a non-issue, until it is enhanced by robotic technology such as driverless cars.

As a result, an ideal ethical policy must resolve the inherent conflict between these two groups to some extent, finding a balance between public acceptability and moral standards. It appears that it must be just acceptable enough to gain human users' confidence and adoption, yet just moral enough to avoid repeating the most heinous of human desires. Similarly, an ideal ethical policy is better characterised as a collection of procedures that should be followed in sacrifice or dilemma circumstances, rather than a computer solution to the trolley problem. These scenarios frequently arise in areas where the law is silent, or where the vehicle cannot offer a comprehensive response to whom it should prioritise or sacrifice via its actions. The ethics of a self-driving cars therefore falls firmly inside these many gaps, and regardless of the ethical policy in place, it is critical that the surrounding decisional architecture reflect and support this complexity, rather than skirting or denying its depth. In this vain, the next section will aim to develop an architecture that is adaptable enough to accept a variety of different sorts of ethical rules while leaving the question of which ethical policy is the "correct" one open.

5.3 The Ethical Valence Concept's Claims and Foundations

The Ethical Valence conceptual approach is best understood as a type of moral claim mitigation. Every road user in the vehicle's surroundings has a claim on the vehicle's behaviour as a condition of existing in the decision context, in accordance with the underlying concept. To put it another way, every individual—from pedestrian to passenger—has a distinct expectation of how the car will treat them throughout its planning, which, when backed up by data on human well-being, provides the vehicle a reason to operate in a certain way. The ethical valence concept depicts self-driving cars as an ecological organism (Wilson 1993, Haidt and Joseph 2004, Gray, Young et al. 2012), whose agency is directly impacted by its surroundings' claims. The strength of a claim might vary. A pedestrian's claim to safety, for example, may be stronger than a passenger's claim if the former is more likely to be badly harmed because of a self-driving collision. As it drives through its surroundings, the self-driving vehicle's objectives is to (excellently) accomplish as many claims as possible, reacting in proportion to the strength of each claim.

Individual claims might be regarded analytically as contributing or pro tanto causes for the vehicle's behaviour (Appiah 2008, Dempsey 2016). Each claim is a contributory "ought," which implies that the strength of a claim is proportional to how strongly its "ought" to respond

to the individual's claim, or his "moral pull" (Shapiro 2018). in normal driving conditions, for example, a self-driving car has every incentive to prioritise its passenger's claim to safety in its tactical decision-making. When a dilemma scenario develops and an inevitable collision is forthcoming, the vehicle will be confronted with extra reasons to prioritise the safety claims of other road users, such as pedestrians or bicycles, all things considered. Because both variables are antagonistic, the vehicle must choose which is the more powerful and act on that factor. Thus, by reacting to the strongest attraction in its surroundings, the vehicle is doing what its "ought" to do morally.

The purpose of claim mitigation in the ethical valence concept is to capture the role of normative ethics in the decision-making of a self-driving car. In other words, claims allow the vehicle to decide what morality requires in critical situations by assessing how changes in road user welfare affect the rightness or wrongness of a self-driving car's actions. In many respects, this approach is influenced by the distributive ethics "competing claims" paradigm (Sovacool, Burke et al. 2017, Symons 2019), and when viewed in this perspective, the ethical valence notion appears to be fundamentally utilitarian. However, this perspective is flawed. In practise, because the EVT's main goal is to provide a public-acceptance-sensitive account of AV ethical decision-making, we must seriously consider the idea that other potentially normatively relevant factors, such as agent-relative constraints and options (Kalajtzidis 2019), should be included in the theory's foundational structure. We must also resist the seductive temptation to believe that this requires self-driving cars to have some form of moral status or to meet common definitions of personhood such as intentionality, subjectivity, or free will (Alfano, Loeb et al. 2014, Frischmann and Selinger 2018). In this perspective, it's probable that if agent-relative constraints and choices are normatively significant in the setting of self-driving vehicles at all, it's because they reflect the expectations that certain road users may have about the partiality of the self-driving vehicle. In this light, the passenger is likely to expect her self-driving car to prioritise her and her family's needs, particularly when she is in danger of being badly hurt or killed (Nascimento, Vismari et al. 2019).

The function of claim mitigation in the ethical valence concept is to capture the contribution that normative ethics might make to a self-driving vehicle's decision-making. In other words, claims enable the vehicle to determine what morality necessitates in crucial circumstances by measuring how changes in the welfare of road users impact the rightness or wrongness of a self-driving vehicle's conduct. In many ways, this approach is inspired by the "competing claims" paradigm prevalent in distributive ethics (Dignum 2017, Dogan, Costantini et al. 2020), and seen in this light, the ethical valence concept would appear to be foundationally utilitarian.

This vision, however, is imperfect. In practise, because the ethical valence concept's primary objective is to provide an account of a self-driving vehicle ethical decision-making that is sensitive to public acceptance, we must seriously consider the idea that other potentially normatively relevant factors, such as agent-relative constraints and options (Kalajtzidis 2019), must be included in the model's foundational structure. We must also avoid the alluring temptation of supposing that this necessitates self-driving vehicles having some sort of moral position or possessing popular criteria for personhood like intentionality, subjectivity, or free choice (Talbot, Jenkins et al. 2017, Vanderelst and Winfield 2018). In this light, it's likely that if agent-relative restrictions and choices are normatively important in the context of self-driving cars at all, it's because they mirror the expectations that some road users may have about the self-driving vehicle's partiality. In this perspective, the passenger is likely to expect her self-driving vehicle to prioritise her and her family's interests, especially in situations when she is at risk of being seriously injured or killed (Keeling 2020).

Because the self-driving car serves as a proxy or substitute for her practical activity in the road traffic scenario, it is possible that this assumption is accurate (Cunneen, Mullins et al. 2020, Keeling 2020). As a result, if her self-driving car fails to safeguard her and her loved ones from harm, it may behave in an unpleasant manner, i.e., as a "morally superior robot villain," disregarding the importance of the connections she has with her loved ones. Significantly, the inclusion of a form of morally admirable partiality on the part of the self-driving vehicle toward its passengers does not necessarily imply the adoption of a form of passenger-centric exclusivism, in which the passenger's interest is the only normatively relevant factor determining the rightness of the self-driving vehicle's actions, as is frequently implied in popular culture. The fact that a specific person is a passenger in a self-driving car is one normatively important aspect that must be handled together with information about the individual well-being of all road users is adequate, rather than going into detail.

Due to this odd mixture at the factoral level of the EVT, we are moving away from utilitarianism and toward contractarian forms of foundational theory—and perhaps specifically Scanlonian contractualism—because these types of theory typically view the correct list of normatively relevant factors as those that would be agreed upon, consented to, or reasonably unobjectionable for suitably disposed and informed individuals in the first place (Scanlon 2000, Wallace 2019). To be sure, there are substantial disadvantages to utilising (Scanlonian) contractualism as the EVT's fundamental theory. Furthermore, the concept of claim mitigation as given in this chapter is predicated on the idea that an autonomous vehicle role in ethical decision-making is to directly assess the claims and interests of individual road users in its

surroundings. A contractualist explanation of moral deliberation, in which the agent evaluates the reasons people have for rejecting the agent's current motivating principles, then chooses a motivating principle that provides reasons for action that no one could possibly reject, contrasts with the agent's present motivating principles. The "focal point" of contractualist theories, rather than the acts they inspire, is norms and principles, and as a result, the viewpoint from which moral debate takes place is the standpoint from which contractualist theories take place (Habermas 2018). Although this step is not included in the actual choice procedure of the Ethical Valence Theory, it is carried out by the human decision-makers who are involved in the design process throughout the development of the theory. It is possible to interpret this divergence as a departure from Scanlon's original theory in this way, because it requires the division of cognitive labour between the designer and the machine, which is something that Scanlon's original thesis glaringly fails to provide. According to the Value Sensitive Design method (Friedman, Kahn et al. 2013), we may consider this deliberative step to be the major responsibility of the so-called "conceptual phase" of intelligent artefact design. As soon as it is installed in an autonomous vehicle, the EVT does not deliberate "across" principles to find the most acceptable (or least rejectable) option; rather, it simply acts on the principle that its human designers have chosen to implement, which may or may not satisfy the explicitly Scanlonian conditions of reasonable rejection. The following are the three most significant analogies between contractualism and the fundamental framework of Ethical Valence Theory: First and foremost, unlike contractualism, the EVT does not seek to alter common sense morality by providing a metric explanation of what constitutes moral worth. Instead, the EVT seeks to clarify what constitutes moral worth. As opposed to this, it is a pluralist account of morality, which includes space in its description of normatively significant components for both the quality of outcomes and the unique and universal duties of all individuals. A second constraint adhered to by the EVT, which is related to (Scanlonian) contractualism, is that "...in rejecting a moral principle, we cannot turn to assertions about the impersonal goodness or badness of occurrences" (Wallace 2019). In addition, both contractualism and the Ethical Valence Concept adhere to a further 'individualist' restriction, according to which all moral reasons for conduct "...must appeal entirely to the principle's implications for ourselves and other single individuals...." (Scanlon 2000, Wallace 2019) both suggest that They are advantageous to the EVT because they restrict the aggregation of claims in the vehicle's ethical deliberation, guaranteeing that the vehicle only considers direct changes in individual welfare or everyone’s degree of claim fulfilment, rather than aggregated claims. Essentially, we have something that

is similar in appearance to an ecumenical variation of action consequentialism that adheres to contractualist limitations and norms.

5.3.1 Valence as a Concept

It is argued that the contractualist motivations of Ethical Valence Concept serve primarily to track common sense morality as it applies to the case of autonomous vehicles, whereas the concept of "valence," as well as the theory's use of it, serve primarily to track (empirical) estimates of public acceptability. The strength of each road user's absolute right not to be hurt by an autonomous vehicle has already been established. The strength of this right grows in proportion to the severity of the injury experienced because of an accident, as previously established. The most important thing to remember about road users is that they each have a distinct valence, which fluctuates in strength in proportion to how that specific user's identity connects to several various sets of traits and behaviours. criteria outside of the technical limitations of identification, data collection, and processing, there are no specific criteria that should inform a valence, but they can include features such as different age groups, socio-economic levels, or professions in the vein of the Moral Machines Experiment (Awad, Dsouza et al. 2018), or they can include forms of morally admirable partiality that might exist between the self-driving vehicle and its passenger(s) (Keeling 2020). Even though the bulk of the criteria used to identify valences remain controversial, it would be absurd to attempt to summarise the whole extent of the issue in this space. In our opinion, while attempts to define and define the moral limits of valence features fall squarely under the jurisdictional purview of moral philosophy, such efforts must also engage with the burgeoning world of so-called ethical design principles (Dignum 2019) and emerging ethico-legal doctrines from various political institutions (Bonnefon, Shariff et al. 2019), which taken together at the very least, may impose very strict limitations on the use of such In his 2017 article which points out that the German parliament's ethics commission on automated and connected driving expressly prohibits "...any distinction based on personal features (age, gender, physical or mental constitution)...", which appears to exclude all but a circumstantial categorization of individuals, discriminating only "cyclists" from "pedestrians" and other such "types" of road users. Accordingly, categorization of road users according to their relative 'vulnerability,' such as is commonly achieved in the subject of traffic psychology, may be successful in satisfying such stringent standards.
Valences' attractiveness and function in nature may both appear to be questionable when confronted with such tight informational limitations. It is likely that two more issues provided

by the selection of valence criteria may lend greater credibility to this point of view. First and foremost, at least from the standpoint of acceptability, there is a clear relationship between decisional accuracy and robustness of information, which simply means that the finer the valence criteria are, the more closely the resulting decisions of the self-driving vehicle will be able to track public acceptability (Martin 2019). In an ideal world, valences would incorporate all the essential facts about a specific situation to enable for the best-informed decision possible. Various sources of information, including explicit input from the passenger, empirical studies, data retrieved from environmental perception, and vehicle-to-vehicle or vehicle-to-device communication, are asserted to be available for gathering these facts. These facts are said to include several traits that are comparable across different road users including health status, age, income, and occupation. The inability to clearly distinguish between which qualities are significant adds to the already tough process of recognising which characteristics are relevant. Consider the idea of age, for example. In certain societies, the old are valued more highly than the young, for reasons like as knowledge, perspective, or other less utilitarian reasons, but others appear to worship the cult of youth, and as a result, may be more ready to sacrifice the young in the case of a self-driving vehicle crash. It is possible to record preferred gerontophobia in both scenarios by empirical investigation of the surrounding environment, and the valences will reflect these fluctuations because of the findings. When it comes to the complete examination of an individual's welfare (and, subsequently, his 'claim,' in the sense that the elderly is more likely than younger people to suffer damage or death because of a collision), age can be an important factor to consider (Liu, Hainen et al. 2019) When it comes to a complete examination of an individual's welfare (and, hence, his 'claim,') age might be an important factor to consider. As a result, it is essential to examine the nature of the facts that serve as the foundation for valences when formulating empirical research and surveys, as well as when interpreting the data, to avoid ambiguity.

The unintended or collateral implications that the selection of certain characteristics may have on the overall traffic environment are a second hurdle that the concept of valence, as we have mentioned, must overcome before it can be deemed effective in its application. A vehicle that is designed to discriminate between motorcyclists who are wearing helmets and motorcyclists who are not wearing helmets in the vehicle's environment is an example of a self-driving vehicle that is designed to discriminate between motorcyclists who are not wearing helmets in the vehicle's environment, according to (Arkin 2016) in his paper on autonomous vehicle ethics. While it may be tough to sacrifice a helmet-wearing rider in an eventual accident, the vehicle lowers harm by basically targeting the most vulnerable road user on the highway. It is

possible that this will result in an unpleasant trade-off in the case of a motorbike collision. However, by doing so, it disincentivizes other motorcycle riders from wearing helmets, therefore indirectly raising the amount of danger in the overall traffic scenario. While choosing to sacrifice the motorcyclist who does not wear a helmet, the government makes another type of mistake: by placing a high value on the safety of helmet-wearing motorcyclists, the government unfairly targets illegal road users, displaying an uncomfortable form of technological paternalism that may appear to 'punish' those who do not adhere to the letter of the law. The government should reconsider its decision to sacrifice the illegal road user. This is another case in which the interplay between an impartial claim to safety and a valence may make the selection of a self-driving vehicles more difficult to comprehend. A "helmet or no-helmet" criteria may be eliminated from consideration because of the detrimental implications that employing such a criterion might have on the traffic environment. This is even though it plainly violates the right to personal safety of road users. By the same token, if the problem is handled more broadly than as a concern of public safety, and more precisely as a question of public acceptability, empirical study may be able to absorb the consequences of this influence more effectively. So, it shouldn't be too surprising that the development of valence criteria is an extremely contentious topic, in part because it requires policymakers and engineers to determine which seemingly inconsequential information might one day turn out to be the deciding factor in a potentially life-threatening situation. Although it is feasible to alleviate some of this pressure by developing a deliberative process that does not rely solely on valence-type considerations, this is especially true when moral claims and social acceptability are considered separately during the design process.

5.3.2 An Ethical Profile

In the ethical valence concept, the idea of an "ethical profile" refers to a specific decision method or strategy that mitigates the numerous claims and valences of road users and other road users. This is the final conceptual aspect of the theory. At its core, each moral profile provides a different criterion of rightness: a maxim or rule that decides whether a certain action option is right or wrong in the given circumstances. Accordingly, a moral profile describes which claims the self-driving vehicle is sensitive to, when those claims are sensitive to those claims, and how those claims are influenced by a given individual's valence strength. It is possible to organise the mitigation process between valences and claims in a variety of ways, but a preliminary categorical distinction between users who are within the self-driving vehicle

and those who are outside of it can be utilised to honour the unique duty the self-driving vehicle may have in relation to its passengers. To do this, claim mitigation should be conceptualised as the weighing of a passenger's claim against those in the self-driving vehicle's surrounding environment. We may then investigate a range of potential mitigations across these two areas of interest, including the following: A risk-averse altruist moral profile, for example, may prioritise the user who has the greatest valence in the event of a collision, provided that the threat to the s self-driving vehicle's passenger is not severe. In this sort of profile, it appears that the passenger in a self-driving vehicle may be prepared to endure some degree of injury to respond properly in the traffic environment, but not to the point that he or she would die or suffer significantly debilitating injuries because of his or her decisions. A profile like this might help to reduce public concern about so-called "killer cars" and to enhance user confidence in autonomous vehicles by reducing public anxiety. However, a self-driving vehicle's passenger would be prioritised by a threshold egoist type profile if there is no considerable threat of damage to another user who has a greater valence than the passenger, which is unlikely. Perhaps a profile like this could increase public acceptance of autonomous vehicles if the criteria for their use were designed to address (and eventually prioritise) the needs of especially vulnerable road users such as children under the age of six and people with disabilities who are unable to walk or stand on their own.

There will never be a single profile that will be able to resolve the moral and societal problem that autonomous automobiles offer once and for all, since there is no such thing. The selection of a moral profile, as well as the selection of valence criteria, are available as multiple entry points for human control in an autonomous machine, although this is not the case with autonomous machines. The public is also given a say in the choices made by a self-driving vehicle, which is crucial because it guarantees, among other things, that the self-driving vehicle is sensitive to moral urgency and that its deliberation process is fair and well-organized on the side of the public. By focusing on flexibility rather than rigidity, we can assure that the actions of an autonomous vehicle are both acceptable and moral, rather than merely moral.

As previously mentioned, the validity of the Ethical Valence Theory is jeopardised by a well-known computational weakness: the theory's reliance on computations of the degree of harm that is expected to be suffered by certain road users. Since many ethics policies that aim at damage or risk minimization do not have the informational certainty necessary to successfully predict the harmfulness of individual accidents this has become a recurring source of concern in the literature on robot ethics (Arkin 2016, Lin, Abney et al. 2017). As a result, the Ethical Valence Theory is susceptible to similar objections in this regard as well. Because of a lack of

specific knowledge about elements such as the posture of people or the structural integrity of individual cars, estimating the possibility of injury to road users will remain approximative and imprecise. This is exacerbated by ambiguity in the vehicle's assessment of the road itself. Although it has this shortcoming, the ethical valence concept has the advantage of not relying on the possibility of harm in a disproportionate amount when making ethical decisions. There are a variety of elements that impact what the vehicle will do, including the potential for injury and the possibility of an impartial claim being filed against it, among other considerations.

The ability of the ethical valence concept to anticipate and respond to the emergence of a second dilemma scenario that may arise because of its original morally optimal action choice is one last theoretical difficulty. For autonomous cars, this relates to the problem of temporal horizons in the decision-making process, and as a result, it constitutes both a technology barrier for self-driving vehicles and a fundamental theoretical error in consequentialist ethics. If the self-driving vehicles has perfect foresight, it should be able to anticipate and mitigate any negative externalities that may arise because of its ethical decision-making in the future. Indeed, it's likely that this is one of the implicit assumptions that underpin the concept of autonomous vehicles behaving as superhuman drivers, as has been suggested by some. Because ‘short-sighted' autonomous vehicles may appear to be immoral in comparison, it may appear that designers have a techno-ethical duty to incorporate lengthy time horizons into vehicle decision-making. When elucidating appropriate moral profiles, the moral significance of a temporal discount rate and the ostensibly relevant characteristics of that rate become important considerations. This is especially true if these considerations are seen to be important when it comes to public acceptability in a contractualist spirit. However, these considerations are not the only considerations. In the absence of such information, it is not abundantly clear that the functional moral agency of autonomous vehicles will be required to extend much further into the future than is currently anticipated, particularly if this process threatens the real-time performance of the decision-making algorithm. If an autonomous vehicle is confronted with these types of issues, the boundary between its decisional ethics policy and its larger society policy for autonomous vehicles is usually blurred, if not completely erased. Two terrains exist inside the action space of unavoidable accidents that do not necessarily have to be completely consistent with one another to function properly. Briefly stated, when faced with a dilemma, it is critical that the vehicle make the ethically optimal choice within its environment, and in an ideal world, it would also account for any additional harm that it might cause. However, it is possible that the need for optimization will extend beyond this horizon in the case of autonomous vehicles due to wishful thinking.

Because of this, ethical valence concept is meant to serve as an adaptive response to the many types of uncertainty that are characteristic of the early stages in the adoption of self-driving vehicles from a theoretical viewpoint. ethical valence theory, which was developed in response to moral ambiguity and a lack of universal ethical consensus, presents customizable and flexible moral profiles in a claim-based framework that is not closely linked to traditional moral theory or any one idea of what is right or wrong. ethical valence concept are used to help vehicles in deciding on which facts and features should be considered when making decisions in ambiguous situations. This information has been backed up by empirical investigation and is available when the vehicle is operating in ambiguous situations. While it does not want to create "morally superior robot villains" or to replicate criminal behaviour in human drivers, it does not rule out the possibility. Instead, it seeks to provide a satisfactory solution to the moral and societal issues posed by self-driving vehicle, as well as a practical tool for engineers to utilise in their daily job.

5.4 Ethical Valence Theory Computational Implementation

Described in this section is the process of putting the ethical valence concept into practise. The topics mentioned in the preceding sections will be examined again, but this time from the perspective of computational standpoint. As it drives the thousands of kilometres it will traverse, a self-driving vehicle will encounter problematic scenarios on public highways at various points along the way. In these situations, every potential response will result in the harm (perhaps fatal) of a road user. We found that the emergence of a dilemma situation triggers the activation of an ethically constrained deliberation model, the ethical valence concept, which is a deliberation model with ethical restrictions in our self-driving vehicle's simulation. To distinguish between this model and the one that is used in regular settings, where performance and efficiency constraints impact the decision-making process, it is necessary to distinguish between them.

5.4.1 The MDP Algorithms

It will be addressed in further depth in this section about the various components of a Markov Decision Process (MDP). A brief and abbreviated introduction will be provided; however, comprehending this introduction will be necessary for understanding the future sections.

According to (Sigaud and Buffet 2013), the technique is composed of five components, which are as follows:

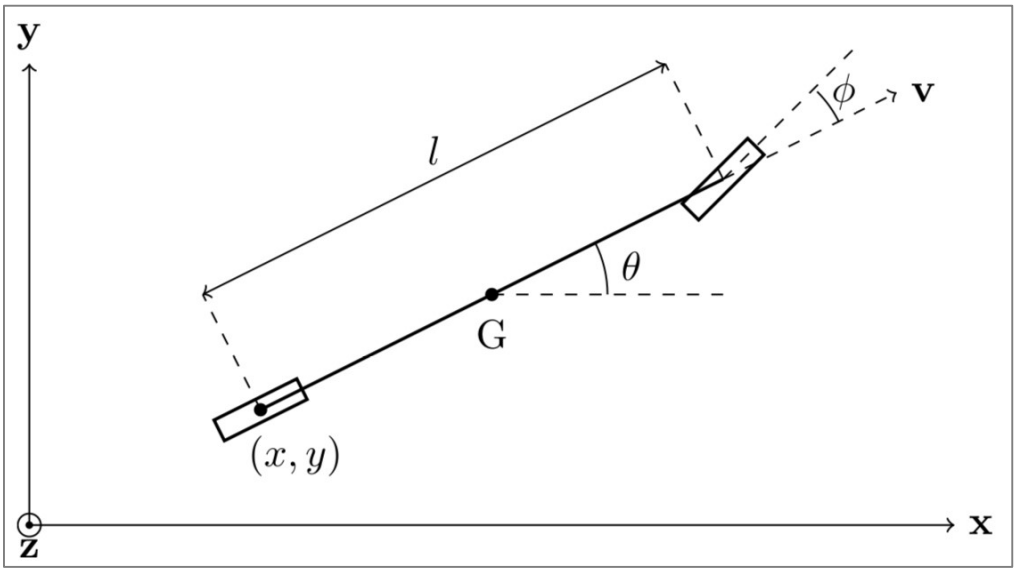

Figure 5.1: Self-driving vehicle's state representation

- ✓ ***The state space ($s_i \in S$):*** collection of all possible self-driving vehicle configurations, and the behaviour of the system is described by a series of states that are repeated throughout time.
- ✓ ***The action set ($a_i \in A$):*** collection of all possible actions available to the self-driving vehicle. It is responsible for initiating the shift from one state to another in the self-driving vehicle's internal state transition system.
- ✓ ***Transition probability (T):*** whenever a self-driving vehicle is in one state, the transition probability (T) represents the likelihood that performing an action will result in the self-driving vehicle shifting to another state; it is represented by the formula $p(s_{t+1} | s_t, a_t)$ when the self-driving vehicle is in another state.
- ✓ ***The reward function (R):*** the performance of a function in relation to the global goal can be quantified.
- ✓ ***The discount constant (γ):*** is a variable that may be used to change the value of utility between time $(t + 1)$ and the present time (t). It is described as between the numbers [0, 1].

According to the example that will be used in the application section, the state may be defined as $(x, y, \theta, v, \emptyset)$, with the only source of information being the self-driving vehicle's configuration (the configuration of all other road users is already accounted for in the reward

function). Each of the variables in the equation (x, y) represents the position of the centre point of the rear-axis, as well as the direction of the vehicle, the scalar velocity, and the steering angle.

The outcome of a MDP algorithm is a policy (π^*) that, for each state, identifies the most optimal action to be performed and then implements that action in that state. As shown in eq.5.1, when this action is done at the state s_t, it increases the value V (s_t, a_t) at the state s_t to the greatest extent possible.

$$V^*(s) = \mathbb{E}^{\pi} [\sum_{i=0}^{\infty} \Upsilon^i \cdot r_i (s, a) | s_0 = s_i] = max_{a \in A} [r (s, a) + \sum_{s' \in S} p(s'|s, a) V(s')] \quad \text{Equation (5.1)}$$

By simply creating a connection between the actions that maximise $V(s_i)$ and s_i, then the policy is retrieved from eq.5.1 as shown in eq.5.2

$$\pi (s) \in argmax_{a \in A} [r (s, a) + \sum_{s' \in S} p(s'|s, a) V(s')] \quad \text{Equation (5.2)}$$

5.4.2 Dilemma Situations classifications

During each stage of the self-driving vehicle's trip, it should be possible to assess whether a circumstance poses a moral problem that should be investigated and contemplated in further depth. It is necessary to categorise circumstances to determine if the self-driving is required to act in line with ethical limitations or just to achieve certain objectives. Self-driving vehicles are expected to comply to three principles that explain their responsibilities towards other road users in their near area. These rules are as follows: One or more of these rules may be violated across all possible actions, indicating that the non-dilemma portion of the Self-driving vehicle's decision-making will be unable to cope with the consequences of all possible actions and that ethical deliberation will be required for the Self-driving vehicle to act in an acceptable manner. Harm is defined as the unpleasant consequences that a person experiences because of colliding with another road user, regardless of the type of collision that occurred. Below are the responsibilities of Self-driving vehicles in relation to other road users:

- ✓ It is critical that the lives of the passengers are not jeopardised in any way.
- ✓ It is not acceptable to put the lives of road users and those in the surrounding environment at danger.
- ✓ It is essential to follow all traffic regulations.

It is important to note that both the first and second criterion apply to interactions between other road users and the self-driving vehicle in question. Vehicles are represented as rectangles, whilst humans are represented as squares, to make their implementation as simple as possible. It is judged that a collision has occurred if these structures come into touch with one another because of the performance of a certain activity. Based on the findings of (de Moura et al 2010), a safe border may be established around the self-driving vehicle to prohibit the execution of actions that would eliminate the possibility of breaking without swerving to avoid an accident from occurring. This circumstance would not necessarily be deemed a conundrum that needed to be resolved since there are other viable choices available.

So far, there has been no concern given to the compliance with traffic rules and regulations. The vehicle, on the other hand, should be cognizant of the interplay between ethical and legal activity, which is a desirable characteristic in this context. Another way of putting it is that where there is an inherent conflict between preventing human injury on the one hand and complying to traffic laws on the other, it is better to prioritise avoiding the former over the latter. Therefore, it is important to describe the MDP algorithm such that this priority may be stated in a manner that is independent of the effect of the temporal discount rate on the priority expression. Nevertheless, following a violation, the self-driving vehicle must return to a 'safe' state to prevent a second collision from occurring as a direct result of the action it performed in the first place. All activities result in a collision if this is the case, as determined by the decision-making process of the self-driving vehicle.

Compliance with traffic rules from the outset of the self-driving vehicle's development is an issue that has gotten little attention in the literature. Put another way, legal conformity presents challenges related to the interpretation of laws that may be ambiguous, allow for exceptions, or be internally inconsistent, and the resolution of all these challenges may necessitate the application of common sense thinking to be successfully resolved (Rizaldi and Althoff 2015, Prakken 2017). As a result of complying to traffic rules, it is important to include somewhat abstract criteria, which are used in legislation to map real-world behaviour, into a self-driving vehicle and, more broadly, into an autonomous system to ensure that it operates safely (Leenes and Lucivero 2014, Boden, Bryson et al. 2017). In certain cases, such as (Esterle, Gressenbuch et al. 2020), researchers have already attempted to incorporate some aspects of various traffic codes particularly those relating to circulation and behaviour into a self-driving vehicle. Most of these endeavours have been carried out utilising logic-based techniques to simulate restrictions, with logic-based approaches representing just the procedural demands that are commonly included in traffic regulations, as opposed to other approaches.

Consequently, while the obligation requiring traffic code observance is properly specified, it may not be required to fully apply the whole traffic code to its full extent because of these factors. To avoid going beyond the scope of this thesis and discussing the methods by which all traffic codes should be implemented within a self-driving vehicle, a set of logical rules will be used to represent the procedural rules that are present in each traffic code. This will avoid going beyond the scope of this thesis and discussing the methods by which all traffic codes should be implemented within a self-driving vehicle. If this set of criteria is followed, the self-driving vehicle should be able to cruise in a legal way virtually all the time. In this part, we will not discuss exceptions to the code or how to settle disagreements amongst rules, as they are considered ethical decisions rather than legal decisions (even if ideally the procedures to solve conflicts between rules present in traffic codes should be used where possible).
Consider the following logic rules in a straight-line domain, with no pedestrian strips or semaphores, and a solid double line between each of the logic rules:

- ✓ Do not cross into the opposite lane of traffic unless necessary.
- ✓ Avoid parking your vehicle on the sidewalk.
- ✓ Drivers are not permitted to exceed the specified speed limit.

It should be noted that the previous example contains a simplification that is only valid in a small number of unique instances. In general, the self-driving vehicle should be intended to target the 4th or 5th level of automation in typical contexts, thus the real set of rules will be considerably wider in scope than these three rules.

5.4.3 An Algorithm for Ethical Deliberation

All action that the self-driving vehicle can do is evaluated considering the collection of tasks that have been specified. The ethical valence concept must still decide on which action to take and which action to carry out if, at a certain point, there is no acceptable alternative available. Specifically, it accomplishes this using two variables: *valence* and *injury*. It is comparable in general terms to that of (Mittelstadt 2019) in that it takes into consideration world states as well as decisions (formerly referred to as 'acts') and the repercussions of those decisions, as well as the consequences of those decisions. For its part, our method differs somewhat from the others in that it provides a quantification of the effects of prospective actions as well as the inclusion of uncertainties in the execution of those acts, which is particularly essential.

5.4.3.1 Injury

It is necessary to take "injury" into account in ethical reasoning to estimate the danger for self-driving vehicle occupants and other road users engaged in a hypothetical collision, and therefore to establish whether they have a legal claim against the vehicle. To determine the severity of an accident, for many years, the difference in velocity between two involved road users (Δv) has served as the key variable (Evans 1994, Evans 2001, Jurewicz, Sobhani et al. 2016, Organization 2017).

It is the bulk of vehicle crash research performed in the field of accident data that is used to establish the function of (Δv) in collision outcomes. This is because historical accident data is easier to get by than current accident data. There are two metrics that may be used to evaluate injury: the risk of fatality (as calculated by the risk of death calculator) and the Abbreviated Injury Scale (AIS) (MacKenzie, Shapiro et al. 1985, Hsu, Wu et al. 2019). It is necessary to consider not only fatal collisions but also collisions that can result in severe injury (referred to as MAIS3+, which indicates that at least one injury in some region of the body is greater than AIS3, on a scale that ranges from zero to six) in this context, so we will use the latter term. Whenever the European Union measures road traffic accidents, this number is used as a baseline to compare performance between member states (Weijermars, Bos et al. 2018).

Table 5.1: The collision threshold that is utilised in fatality collisions (Δv)

Classification	Contact point	Δv value (m/s)	source
Pedestrian collision	–	6.94	(Kröyer 2015)
Vehicle collision	Rear	10.56	(Jurewicz, Sobhani et al. 2016)
	Front	7.78	(Jurewicz, Sobhani et al. 2016)
	Far side	6.39	(Jurewicz, Sobhani et al. 2016)
	Near side	5.56	(Jurewicz, Sobhani et al. 2016)

There are several different types of severe injury thresholds, all of which are included in table 5.1 above. Typically, an injury is deemed "severe" if it reflects an MAIS3+ injury risk of 10% or more. Injury severity was calculated using definition of severe injury, which is more than 9

(defined as the squared sum of AIS for the three most seriously damaged body areas). This definition is tougher than MAIS3+ and was used to compute the ISS for the pedestrian instance as well. Insurance policies cover both the near-side (driver's side) and far-side (passenger's side) of a lateral collision. It is applied the same penalty that is used for collisions between vehicles when a single vehicle collides with another single vehicle. The data presented in (Jurewicz, Sobhani et al. 2016) was collected by the National Highway Traffic Safety Administration (NHTSA) and was previously published in (Bahouth, Graygo et al. 2014). It considered injuries in the front seat, with a seat belt, without rollover, with a passenger age ranging from 16 to 55, involving passenger vehicles and heavy vehicles, and it considered injuries in the front seat, with a seat belt, without rollover, with a passenger age ranging from 16 to 55, Retrospective analysis has certain drawbacks, as will be discussed below. (Rosen, Stigson et al. 2011) feel that the data is biased because it is being collected from a limited number of nations. Aside from that, age is a key component in the pedestrian scenario (Kröyer 2015), and as a result, the distribution of age in the investigated population plays a role in the resultant curve that is not considered in the analysis. In addition, underreporting of non-dilemma cases (Hyder, Paichadze et al. 2017), estimation of collision velocities (Rosen, Stigson et al. 2011), neglect of a vehicle's mass and geometry (Mizuno and Kajzer 1999, Martin and Wu 2018), and the use of different methodologies to evaluate AIS scores (Weijermars, Bos et al. 2018) all reduce the precision of this approach (Rosen, Stigson et al. 2011). Taking into consideration contextual information is important since the preceding technique has limits when applied to specific situations (even though it generalises well across a population). This type of system may be used to simulate the collision interaction between two vehicles, where the starting velocity of each vehicle is projected onto the axes n (normal to the contact plane between the two vehicles) and t (tangential to the contact plane).

Evaluating the collision velocity was accomplished using the conservation of linear momentum, which is represented in eq.5.3. It is represented by the variable v_f, which reflects the collision velocity for both road users (k and l), as expressed by the variable v_f. The masses m_k and m_l correspond to the entire mass of the road user (if it is a vehicle, then the vehicle's mass plus the passengers' mass), while the velocities and impacts of k and l are represented by v_l^i and v_k^i respectively.

$$m_k v_k^i + m_l v_k^i = (m_k + m_l)\, v_f \quad \text{Equation (5.3)}$$

Because a vehicle's mass is far greater than that of any pedestrians, we assume that there is no change in the self-driving vehicle's velocity in accidents with those involved. Considering that the most common factors used to predict damage for pedestrians are the kind of vehicle involved (due to the height of bonnet leading edge) (Mizuno and Kajzer 1999, Lefler and Gabler 2004) and the vehicle's collision velocity (Mizuno and Kajzer 1999), this simplification was made. As a result, the ultimate velocity of the pedestrian is regarded to be identical to the self-driving vehicles. It is the same rationale that is used for vehicle-to-vehicle collisions that is used when a vehicle collides with a static object, with the exception that v_f is set to zero instead of one.

Eq.5.4 below defines harm as the quantification of the severity of an accident, which is the result of the accident (Esterle, Gressenbuch et al. 2020). It is computed for each road user based on the velocity variation caused by the collision, with the velocity at contact for the road user k, v_k^i, and the forward velocity v_f as inputs. The coefficient of structural vulnerability (v_{vul}^k,), which is determined later in eq.5.4, is taken into consideration. This arrangement takes into consideration the impact force as well as the structural susceptibility to such a force, among other things.

$$h^k\ (s_t,\ s_t',\ a_t) = v_{vul}^k \cdot (\|v_f - v_k^i\|) \quad \text{Equation (5.4)}$$

Whether or not two vehicles with varying dimensions and masses give the same level of protection to their occupants is defined as compatibility (Mizuno and Kajzer 1999). In the opinion of (Mizuno and Kajzer 1999, Petzoldt, Schleinitz et al. 2018), sport utility vehicles, for example, are designed to protect their occupants while being hostile against other cars. For pedestrians, the height of the bonnet leading edge explains why some cars are more dangerous for pedestrians than others. This is because the site of harm is determined by whatever portion of the body the vehicle contacts when it hits a person. According to (Crocetta, Piantini et al. 2015) pedestrian can impact the hood in a variety of locations, which alters the way they are projected onto the ground, resulting in either more or lesser damage. c_{vul} is a constant that reflects all the intrinsic features of a system or item (including its state). Preferably, one would calculate $h^k\ (s_t,\ s_t',\ a_t)$ using the same process used to determine the probability of MAIS3+ injury versus Δv plot (logistic regression with weighting), but velocities at the impact v_k^i are not available in publicly available databases of vehicle collisions, which makes this an impractical option. Aside from that, it would be beneficial to categorise collisions according to

the type of vehicle involved (SUV, sedan, minivan, etc.) as well as the direction of the collision (frontal collisions, near side collisions, far side collisions, against a static object, etc), which are not always available in public databases. Because the determination of v_k^i for collisions is the topic of itself, it is beyond the scope of this thesis to go into greater detail about it. A linear function was used to simplify the $h^k\ (s_t,\ s_t',\ a_t) = f\ (c_{vul},\ \Delta v^k)$ equation, and c_{vul} will be estimated in the application section because of this.

5.4.3.2 Ethical Valences

The goal of a valence is to indicate the degree of social acceptability that is associated to the claims of road users in the vehicle's surroundings, as stated in earlier sections. In this way, certain road users' claims may be more or less "acceptable" to fulfil through the vehicle's action option. The valences track various physical characteristics that are seen to carry social importance, such as height, age, gender, helmet-wearing-cyclist, or stroller-pushing-adult, all of which are detectable by the self-driving vehicle's object classification algorithms in the sense that they are rooted in the phenomenal signature of individuals. Importantly, the intensity of these valences is determined by a sort of ranking or hierarchisation, as shown in table 5.2 below, which ties a road user's claim with a certain class or category of valence.

Table 5.2: Potential valence hierarchical structure

1st Highlight	2nd Highlight	Categorization
Young (0 – 18) years	Pedestrian	A
Adult (18 – 65) years	Pedestrian	B
Old (65+) years	Pedestrian	C
Young	Vehicle passenger	D
Adult	Vehicle passenger	E
Old	Vehicle passenger	F

As a result, there might be more or fewer valence groups depending on the amount or detail of the valence traits under examination. Two characteristics are utilised in this example: age and road user type. According to new study, western cultures prefer to protect the young and vulnerable (as indicated by their susceptibility to injury) in self-driving vehicle collisions (Awad, Dsouza et al. 2018). Considering the findings, the category was devised. When there

are a lot of people, automobiles, or pedestrians, the entity with the most users and highest categorisation gets priority. When comparing a self-driving vehicle with passengers D and E to one with passengers D and F, the latter has a higher valence. When the valence factors are limited to a bare minimum or are clear (as in the example above), the likelihood of numerous road users having the same valence but competing claims increases dramatically. In this sense, there may be situations when determining the level of harm is the most important factor in selecting what action to take. In these cases, the vehicle satisfies the strongest claim in its surroundings, protecting the person whose welfare is jeopardised the greatest, either because of a dangerous situation (high velocity difference) or because of a vehicle vulnerability (detected by the structural vulnerability constant). The operational moral profile, which governs the claim mitigation method between passengers in cars and road users outside of them, complicates this basic maximisation of welfare. Table 5.3 shows two possible moral profiles for achieving this aim. The danger is deemed substantial if the value of Δv exceeds the limits set out in table 5.1.

Table 5.3: Potential moral profiles for a self-driving vehicle

Moral profiles	Interpretation
Altruism with a low risk tolerance	If the risk to the self-driving vehicle occupant is not extreme, then protects the road user with the highest valence.
Egoism at the threshold	If the risk to the road user with the highest valence is not extreme, then protects the self-driving vehicle occupant.

These profiles don't quite match any established moral theories, and they reflect a wider range of egoistic rationality perspectives than any traditional moral theories (Hedden 2015). These compromises are meant to represent differing degrees of agreement between the claims and valences of self-driving vehicle occupant and those of other agents operating inside the self-driving vehicle's environment. These profiles typically confirm the view that a certain degree of morally praiseworthy bias in self-driving vehicle behaviour is conceivable, and possibly even desirable, to best correspond with user expectations or earn user confidence, according to (Lin 2016, Martin 2019). Furthermore, the profiles in table 5.3 are not exhaustive, and they depict somewhat factually confusing interpretations of the numerous different profile types that the Ethical Valence Concept may accommodate. In these versions, the damage calculation is

the major element that informs the different repercussions of the self-driving vehicle's actions, which are based on trade-offs between the passenger's claims and those of other actors in the vehicle's surroundings.

5.4.3.3 Consideration of Ethical Issues

After being informed on the values and hazards, the self-driving vehicle can decide on what to do. The operational moral profile, which is discussed in further detail below, influences this decision greatly. Each moral profile, as shown in the example, reflects a different style of deliberation, as in table 5.4 below. It's worth repeating that the moral profiles and, for that matter, ethical deliberation itself appear in the vehicle's tactical planning only when it's in a dilemma situation; otherwise, they're missing. Aside from that, clear, goal-oriented planning is in place, with standard decision-making criteria being used. Each profile calls for a different approach to implementation.

Table 5.4: Technique for optimization depending on the selected moral profile.

Moral profiles	Consideration
Altruism with a low risk tolerance	Reduce the projected injury from the highest-valence road user until the self-driving vehicle's collision becomes serious.
Egoism at the threshold	Reduce the projected injury to the self-driving vehicle occupant until the risk to the highest-valence road user becomes serious.

The implementation process considers the self-driving vehicle state (s_i, which is represented by (x_i, y_i) position, θ_i direction, v_i velocity, and $\emptyset_i$ steering angle), the environment state (e, which includes the location and velocity of all agents in the environment), the highest road user valence (η), and the maximum Δv as the input. The action that should be performed (a_η) is the output in this example using the altruism with a low risk tolerance as a case study. To begin, calculate all damage measures for all potential actions and subsequent states (represented by the state space s', which is made up of states reached following a single transition). The decisional horizon is comparable to one transition in this situation because the accident will occur shortly afterward. Using equations 5.3 & 5.44 as the first step. The self-driving vehicle is solely responsible for the behaviour of one road user in an ideal accident. Transition

uncertainty, represented as, $p(s_i'|s_i, a_j)$, is used to account for all other road users, given the current state (s_i) and an action (a_j).

```
1 for all a_i ∈ A do
2     for all s_i' ∈ S' do
3         v_f ← calculate final velocities (equations 5.3)
4         h^k(s_i, s_i', a_t) ← calculate harm for all road users
                                (including self − driving vehicle, equation 5.4)
5     end
6 end
```

1st algorithm: Estimation of all potential damages

If all possible outcomes result in a velocity differential larger than Δv (the road user's velocity minus the self-driving vehicle's expected velocity), the accident is regarded extreme, and the self-driving vehicle's passenger's safety takes precedence over the road user's safety. In the setting of the profile under evaluation, the selected action decreases the expected damage to the self-driving vehicle. The value of (Δv) varies based on the sort of collision, to emphasise (as seen by the values in table 5.1). The expected harm ($h_{exp}(s_i, a_j), as\ in\ eq.5.5$) is derived using the transition probability, which provides a mean harm value for a road user - k, given that many states s_i' might be reached for a single state s_i and action a_j, resulting in distinct collisions. In this case, it is thought that the position of all road users, as well as the surveillance of the self-driving vehicle's state, are optimal (no uncertainty in these measures).

$$h^k_{exp}(s_i, a_j) = \sum_{s_i' \in s_i} p(s_i'|s_i, a_j) h(s_i, s_i', a_t) \quad \text{Equation (5.5)}$$

It's likely that the transition probability indicates uncertainty in assessing other road users' behaviour, among other sources of uncertainty in the circumstance. The transition probability will have static values that are dependent on the action and the current situation since the MDP approach given here does not bother itself with such calculations. Each action has an 80% probability of success and a 20% risk of moving the self-driving vehicle to one of the neighbouring states (10% for each). The expectation of r (s, a, s', o) over all s' and o is needed to make optimal decisions.

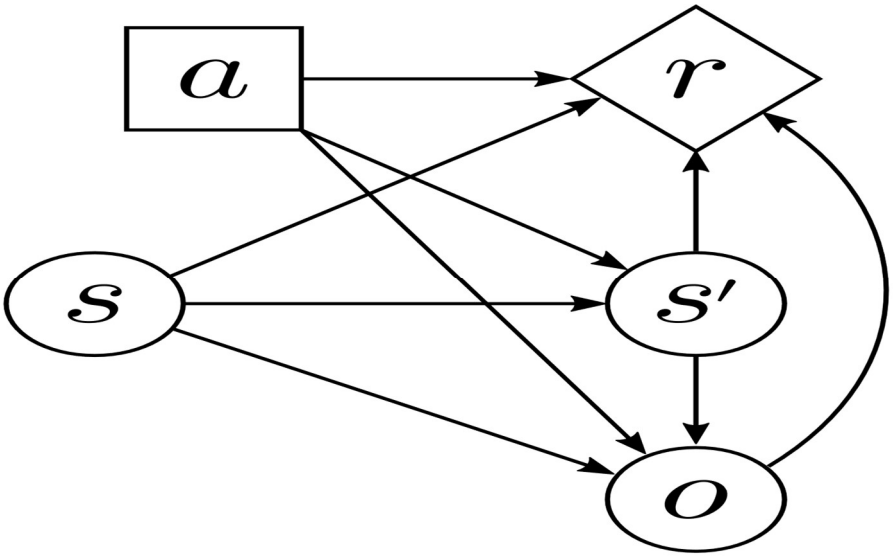

Figure 5.2: State transition uncertainty for POMDP

For extreme activities, success is more likely than failure, with a probability of 90% versus a probability of 10% (as seen in figure 5.1). If the set of allowed actions according to Δv, A_η is not empty, the selected action is the one that minimises the anticipated harm to the road user while maintaining the maximum valence for the actions $\in A_\eta$. When there are several minimal actions to choose from, the one that decreases the self-driving vehicle's expected harm is chosen as shown in below in the 2nd algorithm.

$A_\eta \leftarrow all\ actions\ in\ A\ that\ (\|v_f - v_i^{AV}\| \leq \Delta v)$
if $A_\eta = \emptyset$ *then*
    $a_\eta = argmin_{a \in A}\ h_{exp}^{Av}(s_i,\ a_j)$
else
    $a_c \leftarrow argmin_{a \in A_\eta}\ h_{exp}^{RU}(s_i,\ a_j)$
    if Multiple a_c *exists then*
        $a_\eta = argmin_{a_c}\ h_{exp}^{Av}(s_i,\ a_c)$
    else
        $a_\eta = a_c$
    end
end

2nd algorithm: Decision Making

While this position may look severe when compared to other options, such as the possibility of minimising both values, it is quite reasonable. The goal of achieving both damage reduction and road user harm minimising at the same time is not difficult to attain. However, in our

scenario, both moral profiles are in direct conflict with one another to maximise the safety of one road user over the other, even though there are an endless number of compromises that may be foreseen between the self-driving vehicle and road users. When considering the profile of egoism at the threshold, the only difference between it and the other profiles would be the action deliberation process represented by the 2nd algorithm.

5.4.3.4 Ethical Valence Concept in a Hypothetical Situation

A simplified dilemma situation in an urban context is shown in figure 5.2, which demonstrates how to cope with it. In comparison to the other options, three acts stand out: swerving to the left and colliding with a yellow vehicle; continuing straight and colliding with a pedestrian; and swerving to the right and colliding with the wall. This is accomplished by searching the action space, and in this case just three actions have statistically significant differences in their consequences. Therefore, to aid in the decision-making process, the ethical valence concept must be turned on and engaged.

Scenario 1 displays the collision simulation while the self-driving vehicle is in its initial condition (10, 3.25, 0, 15, 0). The collision simulation is as shown in figure 5.3 (x, y coordinates of the vehicle, direction, longitudinal velocity, and steering angle). For simulating the self-driving vehicles' behaviour, a non-holonomic single-track model developed by (Qian et al. 2016) was used. The collision happens during a decision iteration, which divides the self-driving vehicle's trajectory into periods of 0.5seconds.

To estimate the vulnerability constant, c_{vul}, the data from (Kröyer 2015, Jurewicz, Sobhani et al. 2016) are combined with eq. 5.6 below, $prob_{MAIS3+}$ (Δv), where ProbMAIS3+ (Δv) is the probability of MAIS3+ injury given a Δv, or the difference between the starting and ending velocities prior to the collision, respectively. As previously stated in the preceding sections, this is an ineffective approach of accounting for such circumstances, but it will suffice for the purposes of the example being offered.

$$c_{vul} = \frac{1}{1 - prob_{MAIS3+}\,(\Delta v)} \quad \text{Equation (5.6)}$$

The preference order is shown in table 5.5, based on the valences for each road user in fig. 5.3. In scenario 1, Δv equals 23.1m/s for a self-driving vehicle-vehicle (frontal collision),

14.1m/s for a self-driving vehicle-pedestrian (pedestrian collision), and 14.2m/s for a self-driving vehicle-wall (frontal collision).

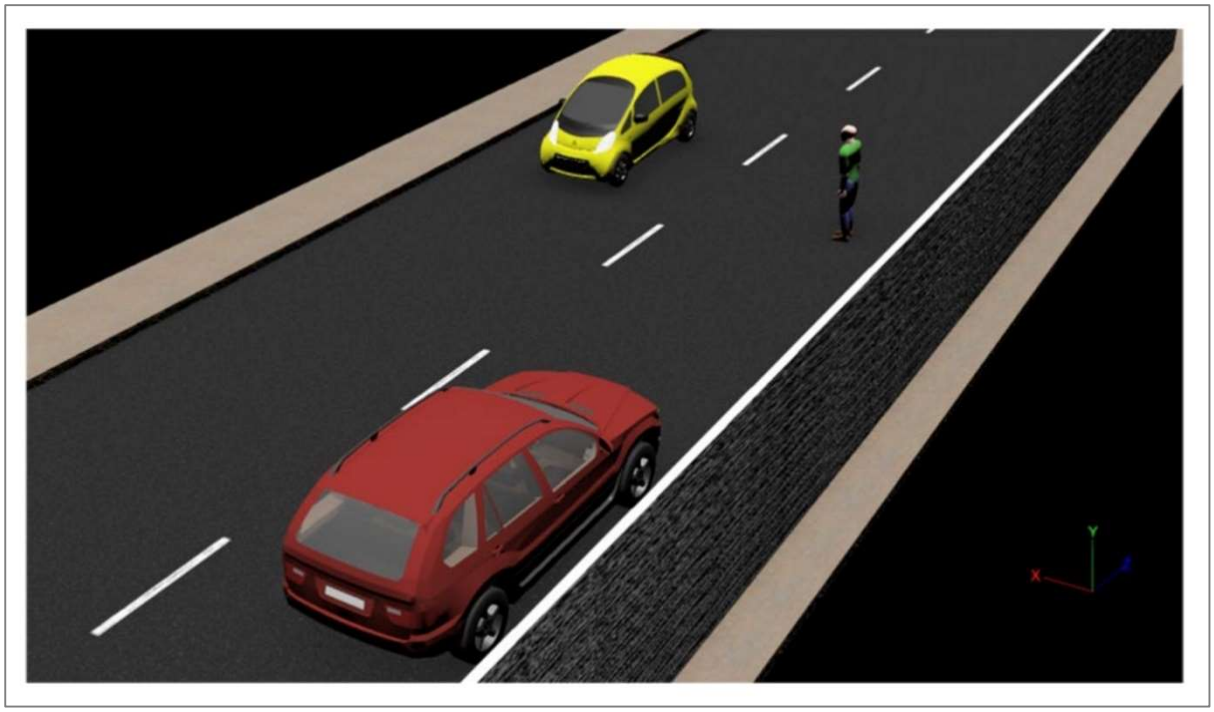

Figure 5.3: Potential dilemma situation

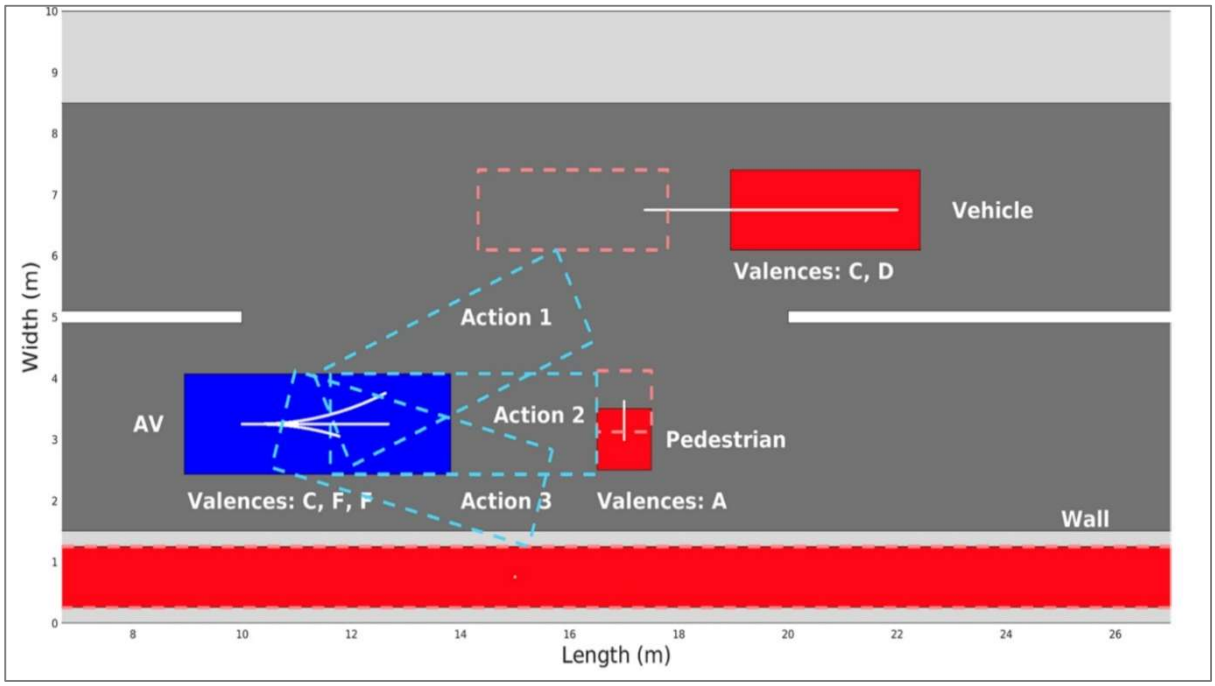

Figure 5.4: Simulation of a collision in scenario 1.

Comparing these values to the constraints established in table 5.1, it is possible to infer that all acts pose a significant danger to the self-driving vehicle as well as to other road users on the highway. Due to the altruism with a low risk tolerance profile being followed, the pedestrian would be run over, since the self-driving vehicle must be prioritised (Δv is over the limit, therefore the self-driving vehicle's damage is decreased, choosing the value highlighted in red from table 5.6; such an approach is seen in the 2nd algorithm), and the pedestrian would be murdered in the process. Every possible collision scenario is illustrated in table 5.6, along with the resulting damage and expected damage (total of damages weighted by transition probability as in equation 5.5) that were computed for each self-driving vehicle's potential collision.

Table 5.5: Valence Classification

Road user	Valences	Classification
Self-driving vehicle	C, F, F	3°
Vehicle	C, D	2°
Pedestrian	A	1°

Table 5.6: The harm to self-driving vehicle for each potential collision in scenario 1

Collision type	Self-driving vehicle's harm	Self-driving vehicle's expected harm
Vehicle collision	8.77	7.02
Pedestrian collision	0	2.46
Wall collision	15.80	12.64

As demonstrated in table 5.5, if the self-driving vehicle is programmed to have egoism at the threshold as its operational moral profile, a collision with a wall would be the favoured choice since the valences of both the pedestrian and the vehicle are greater than they are for a collision with another object (because both Δv are above the limit, the expected harm of road users with valences higher than the self-driving vehicle is minimised, as shown in red values from table 5.7). For instance, as demonstrated in table 5.7, the nominal road user's harm is provided in the first column, while the predicted vehicle harm and pedestrian harm are provided in the second and third columns, respectively. The predicted vehicle harm and pedestrian harm are attained using the transition probability from equation 5.5 and shown in the first and second columns, respectively. Considering that the wall is a static object, both the actual harm it does and the

expected harm it causes are equal to zero (only human safety is considered; historical, cultural, or affective values of a static object such as a tree or a monument are ignored). The relative locations of road users are shown in figure 5.3, and table 5.11 demonstrates that in scenario 2, where the initial self-driving vehicle's state is (10, 3.25, 0, 7.5, 0), damages and velocity variations would cause the selected actions to become unintended consequences. Collisions involving pedestrians result in velocity differences of 14.87m/s, 5.63m/s, and 6.07m/s. This shows that in this case, neither the pedestrian nor the wall would result in a serious threshold crash.

However, using altruism with a low risk tolerance as the operative moral profile results in the wall collision action being executed (in this case, the expected harm of the road user with the highest valence is minimised), resulting in the action highlighted in red value of table 8, while using egoism at the threshold as the operative moral profile results in the pedestrian collision action being executed (in this case, the expected harm of the self-driving vehicle is minimised), resulting in the action highlighted in red value of table 8 and using the tables 5.8 and 5.9 are structurally similar to tables 5.6 and 5.7, respectively, in terms of their organisation.

Table 5.7: The harm to road users for each potential collision in scenario -1

Collision type	Road user's h	Vehicle's h_{exp}	Pedestrian h_{exp}
Vehicle collision	16.80	15.12	1.57
Pedestrian collision	15.71	1.68	12.57
Wall collision	0	0	1.57

Table 5.8: Quantification of collisions for scenario -2

Collision type	Self-driving vehicle's harm	Self-driving vehicle's expected harm
Vehicle collision	5.10	4.08
Pedestrian collision	0	1.12
Wall collision	6.07	4.86

Table 5.9: Quantification of collisions for other road users in scenario-2

Collision type	Road user's h	Vehicle's h_{exp}	Pedestrian h_{exp}
Vehicle collision	10.85	9.76	0.56
Pedestrian collision	5.63	1.08	4.51
Wall collision	0	0	0.56

Table 5.10: The initial state of road users

Road user	States	Scenario 1	Scenario 2
Vehicle	$(x, y, \theta, v, \emptyset)$	$(25, 6.75, 180, 15, 0)$	$(22, 6.75, 180, 7.5, 0)$
Pedestrian	(x, y, θ, v)	$(17, 3, 90, 1.5)$	$(17, 3, 90, 1.5)$
Wall	(x, y, θ, v)	$(15, 0.75, 0, 0)$	$(15, 0.75, 0, 0)$

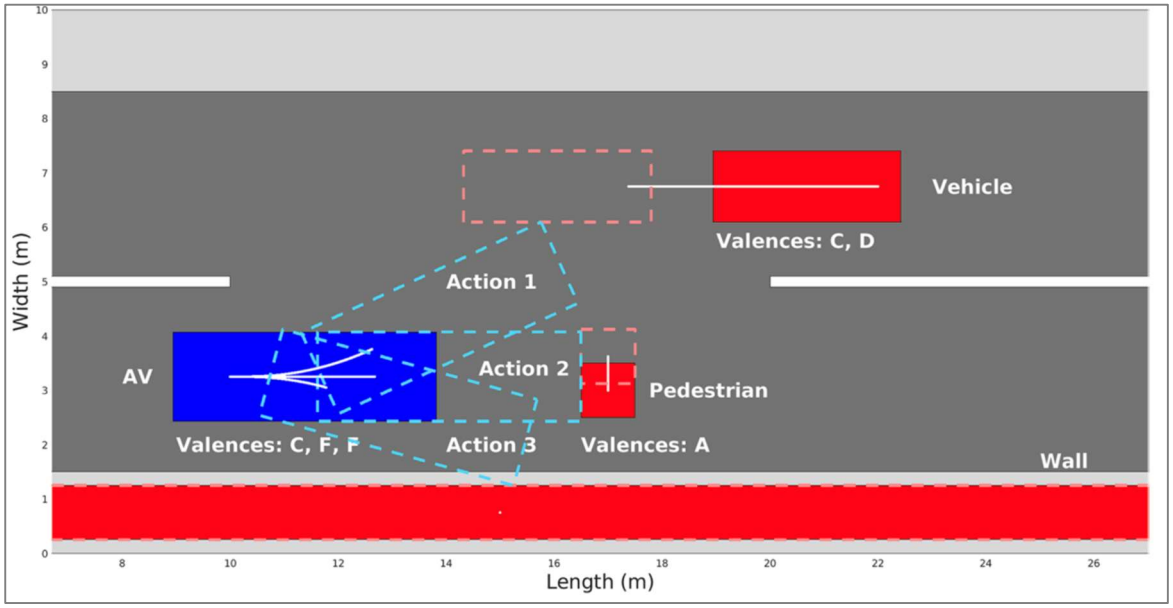

Figure 5.5: Simulation of a collision for scenario 2.

5.5 Summary

While the research detailed in this chapter, as well as the moral and computational approach that reinforces it, should not be the "ultimate" normative answer to the problem of behaviour in self-driving vehicles, they really constitute that solution. However, as will be described further below, there are a variety of reasons why the ethical valence concept may fall short of the expectations of some stakeholders involved in the development of a self-driving vehicles. The ethical valence concept distinguishes between various road users in several ways. For example, it distinguishes between "passengers" and other vulnerable road users, or it distinguishes between the types of vehicles that might be involved in an ethical dilemma situation. It is possible that this positioning will be viewed negatively by some because it does not adhere to some of the prominent normative doctrines that have been proposed in recent years, the majority of which condemn the practise of discrimination between potential victims of a self-driving vehicle's actions (Luetge 2017). The ambiguity around the valences in question may serve to increase this concern even further. In what ways can we ensure that the information they get is accurate and representative of the information we collect? So, what should we do in the case that the information we've obtained threatens to jeopardise civic or human rights? In the future years, it is projected that the design of the decision process for highly autonomous systems, such as self-driving vehicles, would continue to be a heated topic of discussion. As a result, a high degree of interdisciplinary cooperation between scientific fields that have traditionally enjoyed independence will be required, as will a steep learning curve on the part of users, governments, and other institutions in the societies in which these technologies will be implemented. Although it appears that when technology makes autonomous decisions that have implications for human lives and well-being, designers have an additional responsibility to ensure that the outcomes are acceptable, ethical, and respectful rather than just efficient. It is possible to handle a component of this problem through the application of law, and a bigger percentage of the problem can be addressed through ethical considerations and moral philosophy. However, the final judgments must be representative of the people who will be affected by them, as well as their views, claims, and perceptions of what is right. To address this multi-disciplinary and urgent requirement for public engagement and acceptance, the Ethical Valence Concept seeks to provide the foundation for the development of an ethical and acceptable self-driving vehicle for use on the world's highways and other public transportation systems. The major goal of this theory is to do just that.

Chapter 6: Conclusion and Further work

6.1 Conclusion

Self-driving vehicle decision-making algorithms are a common design choice that many engineers overlook because of the ethical ambiguity they introduce into the process. For platooning autonomous cars, the following distance appears to be an easy decision. But it could affect fuel efficiency, response time, and even the flow of nearby traffic. For engineers, this thesis provides a framework for understanding how these factors and various algorithm implementations might impact vehicle behaviour and society in general.

As a result, in many circumstances, engineering decisions are centred on a single objective, such as safety. To ensure the safety of the vehicle when driving laterally, Chapter 3 focuses on modelling steering system delays in a decision-making algorithm. Various technical trade-offs, such as algorithm complexity and implementation overhead, must be considered in this endeavour.

The two techniques proposed in this thesis for a self-driving vehicle decision-making process that take ethical issues into account are milestones in the direction of better engineers and better self-driving vehicle decision-making algorithms. The first technique goes straight to philosophy, using connections between philosophical and mathematical frameworks to support design decisions for a self-driving vehicle steering control system. The use of rule-based mathematical techniques such as set theory and constraints may be justified using deontological reasoning, a philosophy founded on rules.

The use of cost-based mathematical paradigms, such as optimization, can be explained using consequential reasoning, a cost-based philosophy. Model predictive control (MPC), which solves a constrained optimization problem, is one strategy for leveraging the benefits of both deontology and consequentialism, according to chapter 3. Because the choice of weights might lead to a range of vehicle behaviours, a third philosophy known as virtue ethics is used to gain a better understanding of the situation. Engineers can use this mapping to justify their design decisions in terms of cost, constraint, and weight when building a self-driving vehicle decision-making algorithm.

Engineers may be better qualified to ethically programme a self-driving vehicle decision-making algorithm if a diverse range of stakeholders (preferably representative of a society) and explicit consideration of human values are included in the design process. The stakeholders and the values that have been established give advice to the engineers and bring viewpoints to

the table that they may not have considered at the outset in isolation. In Chapter 4, a modified value-sensitive design technique is used to link human values with engineering standards. To help in the accomplishment of the stated human values, many analysis approaches might be used. This method is demonstrated with an example design challenge of a self-driving vehicle speed control system for travelling over a pedestrian crossing. Even if the implementation is ethically designed, there may be a variety of design possibilities to choose from. Extra analysis, such as Pareto optimization, can help with communication to a larger set of stakeholders, as well as additional rationale resources, when deciding which design point to deploy.

6.1.1 Contributions

This thesis contributes the following:

- ✓ ***An MPC formulation to account for steering actuation latency on a self-driving vehicle platform***. The formulation made use of straightforward, computationally efficient models to enhance a self-driving vehicle's capability (Chapter 3).
- ✓ ***Incorporation of traffic regulations governing lane dividers and crosswalks in an MPC and POMDP respectively***. The traffic regulation (§21460) is incorporated into the formulation of the model predictive steering control to examine the adherence of an autonomous vehicle to double yellow lines when manoeuvring around an obstacle (Chapter 3). The speed of an autonomous vehicle passing over a crosswalk is controlled by the traffic regulation (§21950), which is integrated in the model of a POMDP (Chapter 4).
- ✓ ***A mathematical mapping of philosophical principles***. The philosophical principles of deontology, consequentialism, and virtue ethics are mapped to the mathematical concepts of constraints, costs, and choice of weights (Chapter 3). Engineers can better understand the consequences of using a cost, constraint, or even weights by using this method.
- ✓ ***Designing ethical decision-making algorithms using a modified value sensitive design technique***. VSD is used to identify human values that may be involved in the design of a speed controller for crossing scenarios. Iterating through the design task reveals conflicts between the values until the technology is created to match with the specified human values.
- ✓ ***Implementation of an ethical valence concept for claim mitigation in a self-driving decision-making algorithm***. Ethical valence concept describes self-driving vehicle

decision-making as a sort of claim mitigation where various road users hold different moral claims on the vehicle's behaviour, and the vehicle must mitigate these claims as it makes judgments about its surroundings. Actions' consequent damage and uncertainty are assessed and accounted for, leading to an ethical implementation that is realistic. These algorithms are designed to accommodate a variety of "moral perspectives" regarding what morality needs and what road users may anticipate, providing an evaluation tool for a self-driving vehicle's ethical decision-making process.

6.2 Further work

Ethical implications should be addressed at every stage of the self-driving vehicle development process, from sensors to perception algorithms to testing and deployment. While this thesis focuses on the decision layer of the self-driving vehicle stack, human values may be influenced at all levels of the stack.

6.2.1 Generalizability

Scenarios were used to show the effectiveness of the methodologies discussed in the thesis. It is thus unknown how successfully these techniques will scale or generalise to the actual world since, as discussed in Chapter 5, summary, the real world contains an infinite number of possibilities to deal with. Scalability and generalizability are expected to be essential considerations for those building self-driving vehicles and integrating them into the design process might assist in explaining the viability of such a concept.

Some strategies exist for scaling single-user problem formulations, even if scalability was not considered at the outset of the design. For instance, a speed control POMDP for an autonomous vehicle to travel along a roadway with a single pedestrian nearby with an uncertain purpose has been proposed by (Christensen and Gomila 2012, Sigaud and Buffet 2013). For each pedestrian encountered in the experiment, they launched a new instance of the POMDP. Each instance produces a result. The final course of action was the most cautious of the bunch. Utility fusion (Martinez, Heucke et al. 2017) is another method in which the utilities of each encountered pedestrian are added together or decreased. The policy's final action is taken after the utilities are reconciled.

6.2.2 Utilizing VSD to Quantify Engineering Improvements

Engineers' engineering practises might be transformed through value-sensitive design. So, the programmers are compelled to evaluate how the conditional statement or reward function they are coding will contribute to achieving goals outlined in the conceptualization. There are no quantitative measurements here. Such an approach should be evaluated by a user research done by engineers. The identical design task might be undertaken by two different teams of engineers: one that is taught about VSD, the other that is not. An interview with engineers can be conducted after completing a design task so that their thoughts and justifications can be documented.

6.2.3 VSD and Philosophical Principles

The philosophical framework technique, despite its limitations in Chapter 3, might nevertheless be useful in the engineering process. Even though the focus of Chapter 4 was on engineering analysis, a more thorough philosophical investigation may be presented as well. An engineer who understands the link between these philosophical and mathematical frameworks can perform some exploratory analysis or assess basic design consequences. Philosophers, on the other hand, are individuals who are genuinely trained to make moral judgments. Therefore, VSD might be a fantastic approach to further involving them in the design process by allowing them to do a philosophical analysis concurrently with the technical study. (For a legal analysis, a similar argument may be made.)

6.2.4 Strategy

Government authorities recognise the critical role of ethical considerations in the development of autonomous vehicle technology (Lin, Abney et al. 2017, Organization 2017). This thesis asserts that a technique such as VSD may assist policymakers and regulators. This is primarily due to a conversation focused on human values. However, it is critical to involve policymakers directly to ascertain if they find this approach effective.

6.2.5 Prospects

Society must be able to participate in the discussion about self-driving vehicle technology before it can reap the advantages of it. Engaging a wider set of stakeholders allows for more diverse viewpoints to be incorporated into the design process. As a result, everyone will benefit from enhanced technological capabilities.

Reference

Ahrens, D. M. (2020). "RETROACTIVE LEGALITY." The Journal of Criminal Law and Criminology (1973-) **110**(3): 379-440.

Alfano, M., et al. (2014). "Experimental moral philosophy."

Allam, Z. and Z. A. Dhunny (2019). "On big data, artificial intelligence and smart cities." Cities **89**: 80-91.

Althoff, M., et al. (2017). CommonRoad: Composable benchmarks for motion planning on roads. 2017 IEEE Intelligent Vehicles Symposium (IV), IEEE.

Appiah, K. A. (2008). Experiments in ethics, Harvard University Press.

Applbaum, A. I. (2000). Ethics for adversaries, Princeton University Press.

Applin, S. A. and M. D. Fischer (2015). New technologies and mixed-use convergence: How humans and algorithms are adapting to each other. 2015 IEEE International Symposium on Technology and Society (ISTAS).

Aripin, M. K., et al. (2014). A Review of Active Yaw Control System for Vehicle Handling and Stability Enhancement.

Arkin, R. C. (2016). "Ethics and autonomous systems: Perils and promises [point of view]." Proceedings of the IEEE **104**(10): 1779-1781.

Arntz, M., et al. (2016). "The risk of automation for jobs in OECD countries: A comparative analysis."

Arslan, O., et al. (2017). Sampling-based algorithms for optimal motion planning using closed-loop prediction. 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE.

Awad, E., et al. (2018). "The moral machine experiment." Nature **563**(7729): 59-64.

Bahouth, G., et al. (2014). "The benefits and tradeoffs for varied high-severity injury risk thresholds for advanced automatic crash notification systems." Traffic injury prevention **15**(sup1): S134-S140.

Bai, H., et al. (2015). Intention-aware online POMDP planning for autonomous driving in a crowd. 2015 ieee international conference on robotics and automation (icra), IEEE.

Bayerlein, H., et al. (2018). Trajectory optimization for autonomous flying base station via reinforcement learning. 2018 IEEE 19th International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), IEEE.

Beal, C. E. and J. C. Gerdes (2012). "Model predictive control for vehicle stabilization at the limits of handling." IEEE Transactions on Control Systems Technology **21**(4): 1258-1269.

Berntorp, K., et al. (2019). "Motion planning of autonomous road vehicles by particle filtering." IEEE Transactions on Intelligent Vehicles **4**(2): 197-210.

Bharadwaj, S., et al. (2021). "Decentralized control synthesis for air traffic management in urban air mobility." IEEE Transactions on Control of Network Systems.

Binns, R. (2018). "What Can Political Philosophy Teach Us about Algorithmic Fairness?" IEEE Security & Privacy **16**(3): 73-80.

Bobier, C. G. and J. C. Gerdes (2013). "Staying within the nullcline boundary for vehicle envelope control using a sliding surface." Vehicle System Dynamics **51**(2): 199-217.

Boden, M., et al. (2017). "Principles of robotics: regulating robots in the real world." Connection Science **29**(2): 124-129.

Bonnefon, J.-F., et al. (2019). "The trolley, the bull bar, and why engineers should care about the ethics of autonomous cars [point of view]." Proceedings of the IEEE **107**(3): 502-504.

Borning, A. and M. Muller (2012). Next steps for value sensitive design. Proceedings of the SIGCHI conference on human factors in computing systems.

Bouton, M., et al. (2018). Scalable decision making with sensor occlusions for autonomous driving. 2018 IEEE international conference on robotics and automation (ICRA), IEEE.

Brazell, S., et al. (2019). "A Machine-Learning-Based Approach to Assistive Well-Log Correlation." Petrophysics **60**(04): 469-479.

Brown, M. and J. C. Gerdes (2019). "Coordinating tire forces to avoid obstacles using nonlinear model predictive control." IEEE Transactions on Intelligent Vehicles **5**(1): 21-31.

Brüdigam, T., et al. (2018). "Legible Model Predictive Control for Autonomous Driving on Highways." IFAC-PapersOnLine **51**(20): 215-221.

Cai, J., et al. (2018). "Feature selection in machine learning: A new perspective." Neurocomputing **300**: 70-79.

Carvalho, A., et al. (2015). "Automated driving: The role of forecasts and uncertainty—A control perspective." European Journal of Control **24**: 14-32.

Chae, H., et al. (2017). Autonomous braking system via deep reinforcement learning. 2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC).

Chen, K., et al. (2019). "Active steering control for autonomous vehicles based on a driver-in-the-loop platform: A case study of collision avoidance." Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering **233**(10): 1422-1437.

Chen, S., et al. (2019). "A reinforcement learning based approach for multi-projects scheduling in cloud manufacturing." International Journal of Production Research **57**(10): 3080-3098.

Cheng, S., et al. (2019). "Longitudinal collision avoidance and lateral stability adaptive control system based on MPC of autonomous vehicles." IEEE Transactions on Intelligent Transportation Systems **21**(6): 2376-2385.

Christensen, J. F. and A. Gomila (2012). "Moral dilemmas in cognitive neuroscience of moral decision-making: A principled review." Neuroscience & Biobehavioral Reviews **36**(4): 1249-1264.

Claussmann, L., et al. (2019). "A review of motion planning for highway autonomous driving." IEEE Transactions on Intelligent Transportation Systems **21**(5): 1826-1848.

Conway, P., et al. (2018). "Sacrificial utilitarian judgments do reflect concern for the greater good: Clarification via process dissociation and the judgments of philosophers." Cognition **179**: 241-265.

Crocetta, G., et al. (2015). "The influence of vehicle front-end design on pedestrian ground impact." Accident Analysis & Prevention **79**: 56-69.

Cunneen, M., et al. (2020). "Autonomous vehicles and avoiding the trolley (dilemma): vehicle perception, classification, and the challenges of framing decision ethics." Cybernetics and Systems **51**(1): 59-80.

Cunningham, A. G., et al. (2015). MPDM: Multipolicy decision-making in dynamic, uncertain environments for autonomous driving. 2015 IEEE International Conference on Robotics and Automation (ICRA).

Cushman, F., et al. (2006). "The role of conscious reasoning and intuition in moral judgment: Testing three principles of harm." Psychological science **17**(12): 1082-1089.

Damm, L. (2012). Moral Machines: Teaching Robots Right from Wrong, Routledge. **25:** 149-153.

Datta, S. (2019). 3-Dimensional Path Planning of An Unmanned Aerial Vehicle, Texas A&M University-Kingsville.

Davies, S. (1996). "Multidimensional triangulation and interpolation for reinforcement learning." Advances in neural information processing systems **9**: 1005-1011.

Dechesne, F., et al. (2019). "AI & Ethics at the Police: Towards Responsible use of Artificial Intelligence in the Dutch Police." AI & Ethics at the Police: Towards Responsible use of Artificial Intelligence in the Dutch Police.

Del Giudice, M. and B. J. Crespi (2018). "Basic functional trade-offs in cognition: An integrative framework." Cognition **179**: 56-70.

Demirel, B., et al. (2017). "Optimal control of linear systems with limited control actions: Threshold-based event-triggered control." IEEE Transactions on Control of Network Systems **5**(3): 1275-1286.

Dempsey, M. M. (2016). "Against Liability: Toward a Reasons-Based Account of Self-Defense." Against Liability: A Reasons-Based Account of Self-Defense, in Ethics of Self Defense, Christian Coons and Michael Weber, eds.(Oxford University Press, 2016), Villanova Law/Public Policy Research Paper(2016-1033).

Deori, L., et al. (2018). "4-D flight trajectory tracking: A receding horizon approach integrating feedback linearization and scenario optimization." IEEE Transactions on Control Systems Technology **27**(3): 981-996.

Di, X. and R. Shi (2021). "A survey on autonomous vehicle control in the era of mixed-autonomy: From physics-based to AI-guided driving policy learning." Transportation research part C: emerging technologies **125**: 103008.

Dignum, V. (2017). "Responsible autonomy." arXiv preprint arXiv:1706.02513.

Dignum, V. (2019). Responsible artificial intelligence: how to develop and use AI in a responsible way, Springer Nature.

Dogan, E., et al. (2020). "Ethical issues concerning automated vehicles and their implications for transport." Policy Implications of Autonomous Vehicles. Elsevier, The Netherlands: 215-233.

Du, X. and K. K. Tan (2015). "Autonomous Reverse Parking System Based on Robust Path Generation and Improved Sliding Mode Control." IEEE Transactions on Intelligent Transportation Systems **16**(3): 1225-1237.

Enke, B. (2020). "Moral values and voting." Journal of Political Economy **128**(10): 3679-3729.

Epting, S. (2019). "Automated vehicles and transportation justice." Philosophy & Technology **32**(3): 389-403.

Erlien, S. M., et al. (2016). "Shared Steering Control Using Safe Envelopes for Obstacle Avoidance and Vehicle Stability." IEEE Transactions on Intelligent Transportation Systems **17**(2): 441-451.

Esterle, K., et al. (2020). Formalizing traffic rules for machine interpretability. 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS), IEEE.

Evans, L. (1994). "Driver injury and fatality risk in two-car crashes versus mass ratio inferred using Newtonian mechanics." Accident Analysis & Prevention **26**(5): 609-616.

Evans, L. (2001). "Causal influence of car mass and size on driver fatality risk." American Journal of Public Health **91**(7): 1076.

Evans, N. (2017). "Virtue Ethics in Knowledge Management." Handbook of Virtue Ethics in Business and Management: 1231-1243.

Faulhaber, A. K., et al. (2019). "Human decisions in moral dilemmas are largely described by utilitarianism: Virtual car driving study provides guidelines for autonomous driving vehicles." Science and Engineering Ethics **25**(2): 399-418.

Friedman, B., et al. (2013). Value sensitive design and information systems. Early engagement and new technologies: Opening up the laboratory, Springer**:** 55-95.

Frischmann, B. and E. Selinger (2018). Re-engineering humanity, Cambridge University Press.

Funke, J., et al. (2015). Prioritizing collision avoidance and vehicle stabilization for autonomous vehicles. 2015 IEEE Intelligent Vehicles Symposium (IV).

Funke, J., et al. (2017). "Collision Avoidance and Stabilization for Autonomous Vehicles in Emergency Scenarios." IEEE Transactions on Control Systems Technology **25**(4): 1204-1216.

Gallardo, D. I., et al. (2017). "A simplified estimation procedure based on the EM algorithm for the power series cure rate model." Communications in Statistics-Simulation and Computation **46**(8): 6342-6359.

Gao, Y., et al. (2014). A tube-based robust nonlinear predictive control approach to semiautonomous ground vehicles.

Gao, Y., et al. (2010). "Predictive Control of Autonomous Ground Vehicles With Obstacle Avoidance on Slippery Roads." (44175): 265-272.

Garip, Z., et al. (2017). Path Planning for Multiple Mobile Robots Using A* Algorithm.

Ghazal, T. M., et al. (2021). "Internet of vehicles and autonomous systems with AI for medical things." Soft computing: 1-13.

Giacomin, J. (2014). "What is human centred design?" The Design Journal **17**(4): 606-623.

Gogarty, B. and M. Hagger (2008). "The laws of man over vehicles unmanned: The legal response to robotic revolution on sea, land and air." JL Inf. & Sci. **19**: 73.

Goodall, N. J. (2014). "Ethical Decision Making during Automated Vehicle Crashes." Transportation Research Record **2424**(1): 58-65.

Graham, J., et al. (2011). "Mapping the moral domain." Journal of personality and social psychology **101**(2): 366.

Gray, A., et al. (2013). Robust Predictive Control for semi-autonomous vehicles with an uncertain driver model. 2013 IEEE Intelligent Vehicles Symposium (IV).

Gray, A., et al. (2013). Stochastic predictive control for semi-autonomous vehicles with an uncertain driver model. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013).

Gray, K., et al. (2012). "Mind perception is the essence of morality." Psychological inquiry **23**(2): 101-124.

Greene, J. and W. Sinnott-Armstrong (2008). "Moral psychology." The secret joke of Kant's soul **3**: 35-80.

Grüne, L. and J. Pannek (2017). Nonlinear model predictive control. Nonlinear model predictive control, Springer**:** 45-69.

Guanetti, J., et al. (2018). "Control of connected and automated vehicles: State of the art and future challenges." Annual Reviews in Control **45**: 18-40.

Gupta, S., et al. (2019). "Negotiation Between Vehicles and Pedestrians for the Right of Way at Intersections." IEEE Transactions on Intelligent Transportation Systems **20**(3): 888-899.

Habermas, J. (2018). Inclusion of the other: Studies in political theory, John Wiley & Sons.

Haidt, J. and C. Joseph (2004). "Intuitive ethics: How innately prepared intuitions generate culturally variable virtues." Daedalus **133**(4): 55-66.

Halalae, I. and C.-O. Miclosina (2019). "THE IMPACT OF ISAAC ASIMOV'S IDEAS ON THE INTELLIGENT ROBOTS EVOLUTION." Robotica & Management **24**(2).

Hedden, B. (2015). Reasons without persons: Rationality, identity, and time, OUP Oxford.

Hibbard, B. (2012). Avoiding Unintended AI Behaviors. Artificial General Intelligence, Berlin, Heidelberg, Springer Berlin Heidelberg.

Holden, E., et al. (2017). The imperatives of sustainable development: needs, justice, limits, Routledge.

Houjie, J., et al. (2016). Obstacle avoidance of autonomous vehicles with CQP-based model predictive control. 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC).

Hsu, S.-Y., et al. (2019). "Impact of adapting the Abbreviated Injury Scale (AIS)-2005 from AIS-1998 on injury severity scores and clinical outcome." International journal of environmental research and public health **16**(24): 5033.

Huang, Y., et al. (2019). "A motion planning and tracking framework for autonomous vehicles based on artificial potential field elaborated resistance network approach." IEEE Transactions on Industrial Electronics **67**(2): 1376-1386.

Hulse, L. M., et al. (2018). "Perceptions of autonomous vehicles: Relationships with road users, risk, gender and age." Safety science **102**: 1-13.

Hyder, A. A., et al. (2017). "Monitoring the decade of action for global road safety 2011–2020: an update." Global public health **12**(12): 1492-1505.

Iftekhar, L. and R. Olfati-Saber (2012). Autonomous driving for vehicular networks with nonlinear dynamics. 2012 IEEE Intelligent Vehicles Symposium.

Jalalmaab, M., et al. (2015). Model predictive path planning with time-varying safety constraints for highway autonomous driving. 2015 International Conference on Advanced Robotics (ICAR).

Jie, C., et al. (2018). "Stochastic Optimization in a Cumulative Prospect Theory Framework." IEEE Transactions on Automatic Control: 1-1.

Joa, E., et al. (2019). "Estimation of the tire slip angle under various road conditions without tire–road information for vehicle stability control." Control Engineering Practice **86**: 129-143.

Jurewicz, C., et al. (2016). "Exploration of vehicle impact speed–injury severity relationships for application in safer road design." Transportation Research Procedia **14**: 4247-4256.

Kalajtzidis, J. (2019). "Ethics of Social Consequences as a Hybrid Form of Ethical Theory?" Philosophia **47**(3): 705-722.

Kaur, K. and G. Rampersad (2018). "Trust in driverless cars: Investigating key factors influencing the adoption of driverless cars." Journal of Engineering and Technology Management **48**: 87-96.

Keeling, G. (2020). "Why trolley problems matter for the ethics of automated vehicles." Science and Engineering Ethics **26**(1): 293-307.

Kenwright, B. (2018). "Virtual Reality: Ethical Challenges and Dangers [Opinion]." IEEE Technology and Society Magazine **37**(4): 20-25.

Kitchener, K. S. (2016). Ethical Issues and Professional Standards in Psychotherapy. Encyclopedia of Mental Health (Second Edition). H. S. Friedman. Oxford, Academic Press**:** 132-142.

Kong, J., et al. (2015). Kinematic and dynamic vehicle models for autonomous driving control design. 2015 IEEE Intelligent Vehicles Symposium (IV).

König, M. and L. Neumayr (2017). "Users' resistance towards radical innovations: The case of the self-driving car." Transportation research part F: traffic psychology and behaviour **44**: 42-52.

Kowalczuk, Z. and M. Czubenko (2017). "Emotions Embodied in the SVC of an Autonomous Driver System." IFAC-PapersOnLine **50**(1): 3744-3749.

Krotov, V. and L. Silva (2018). "Legality and ethics of web scraping."

Kröyer, H. R. (2015). "Is 30 km/ha 'safe'speed? Injury severity of pedestrians struck by a vehicle and the relation to travel speed and age." IATSS research **39**(1): 42-50.

Kröyer, H. R., et al. (2014). "Relative fatality risk curve to describe the effect of change in the impact speed on fatality risk of pedestrians struck by a motor vehicle." Accident Analysis & Prevention **62**: 143-152.

LaValle, S. M. (2006). Planning algorithms, Cambridge university press.

Lăzăroiu, G., et al. (2020). "Connected and Autonomous Vehicle Mobility: Socially Disruptive Technologies, Networked Transport Systems, and Big Data Algorithmic Analytics." Contemporary Readings in Law and Social Justice **12**(2): 61-69.

Leenes, R. and F. Lucivero (2014). "Laws on robots, laws by robots, laws in robots: regulating robot behaviour by design." Law, Innovation and Technology **6**(2): 193-220.

Lefler, D. E. and H. C. Gabler (2004). "The fatality and injury risk of light truck impacts with pedestrians in the United States." Accident Analysis & Prevention **36**(2): 295-304.

Leikas, J., et al. (2020). Good Life Ecosystems–Ethics and Responsibility in the Silver Market. International Conference on Human-Computer Interaction, Springer.

Li, S., et al. (2017). "Formation control of heterogeneous discrete-time nonlinear multi-agent systems with uncertainties." IEEE Transactions on Industrial Electronics **64**(6): 4730-4740.

Li, Z., et al. (2019). Predictable Trajectory Planner in Time-domain and Hierarchical Motion Controller for Intelligent Vehicles in Structured Road. 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE.

Lin, P. (2016). Why Ethics Matters for Autonomous Cars. Autonomous Driving: Technical, Legal and Social Aspects. M. Maurer, J. C. Gerdes, B. Lenz and H. Winner. Berlin, Heidelberg, Springer Berlin Heidelberg**:** 69-85.

Lin, P., et al. (2017). Robot ethics 2.0: From autonomous cars to artificial intelligence, Oxford University Press.

Lind, G. and R. Wakenhut (2017). Testing for moral judgment competence. Moral judgments and social education, Routledge**:** 25-48.

Liu, J., et al. (2019). "Pedestrian injury severity in motor vehicle crashes: an integrated spatio-temporal modeling approach." Accident Analysis & Prevention **132**: 105272.

Luetge, C. (2017). "The German ethics code for automated and connected driving." Philosophy & Technology **30**(4): 547-558.

Luo, Q., et al. (2019). "Localization and navigation in autonomous driving: Threats and countermeasures." IEEE Wireless Communications **26**(4): 38-45.

MacKenzie, E. J., et al. (1985). "The Abbreviated Injury Scale and Injury Severity Score: levels of inter- and intrarater reliability." Medical care: 823-835.

Maguire, M. (2001). "Methods to support human-centred design." International journal of human-computer studies **55**(4): 587-634.

Malle, B. F., et al. (2016). Which robot am I thinking about? The impact of action and appearance on people's evaluations of a moral robot. 2016 11th ACM/IEEE International Conference on Human-Robot Interaction (HRI).

Manders-Huits, N. (2011). "What values in design? The challenge of incorporating moral values into design." Science and Engineering Ethics **17**(2): 271-287.

Manenti, F. (2011). Considerations on Nonlinear Model Predictive Control Techniques.

Martin, J.-L. and D. Wu (2018). "Pedestrian fatality and impact speed squared: Cloglog modeling from French national data." Traffic injury prevention **19**(1): 94-101.

Martin, K. (2019). "Ethical implications and accountability of algorithms." Journal of Business Ethics **160**(4): 835-850.

Martínez-Díaz, M., et al. (2019). "Autonomous driving: a bird's eye view." IET intelligent transport systems **13**(4): 563-579.

Martinez, C. M., et al. (2017). "Driving style recognition for intelligent vehicle control and advanced driver assistance: A survey." IEEE Transactions on Intelligent Transportation Systems **19**(3): 666-676.

Martinho, A., et al. (2021). "Ethical issues in focus by the autonomous vehicles industry." Transport Reviews: 1-22.

Mattingley, J. and S. Boyd (2012). "CVXGEN: A code generator for embedded convex optimization." Optimization and Engineering **13**(1): 1-27.

McBride, N. and R. R. Hoffman (2016). "Bridging the Ethical Gap: From Human Principles to Robot Instructions." IEEE Intelligent Systems **31**(5): 76-82.

McGuire, J., et al. (2009). "A reanalysis of the personal/impersonal distinction in moral psychology research." Journal of Experimental Social Psychology **45**(3): 577-580.

McNamara, R. A., et al. (2019). "Weighing outcome vs. intent across societies: How cultural models of mind shape moral reasoning." Cognition **182**: 95-108.

Meghjani, M., et al. (2019). Context and intention aware planning for urban driving. 2019 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE.

Mehrara Molan, A. and K. Ksaibati (2021). "Factors impacting injury severity of crashes involving traffic barrier end treatments." International journal of crashworthiness **26**(2): 202-210.

Merat, N., et al. (2018). "What externally presented information do VRUs require when interacting with fully Automated Road Transport Systems in shared space?" Accident Analysis and Prevention **118**: 244-252.

Milfont, T. L., et al. (2019). "The Moral Foundations of Environmentalism." Social Psychological Bulletin **14**(2): 1-25.

Miller, K. W., et al. (2017). "This “Ethical Trap” Is for Roboticists, Not Robots: On the Issue of Artificial Agent Ethical Decision-Making." Science and Engineering Ethics **23**(2): 389-401.

Miller, R. and F. Cushman (2018). "Moral values and motivations: How special are they." The atlas of moral psychology: Mapping good and evil: 1-13.

Mittelstadt, B. (2019). "Principles alone cannot guarantee ethical AI." Nature Machine Intelligence **1**(11): 501-507.

Mizuno, K. and J. Kajzer (1999). "Compatibility problems in frontal, side, single car collisions and car-to-pedestrian accidents in Japan." Accident Analysis & Prevention **31**(4): 381-391.

Mladenovic, M. N. and M. Abbas (2014). Priority-based intersection control framework for self-driving vehicles: Agent-based model development and evaluation. 2014 International Conference on Connected Vehicles and Expo (ICCVE).

Mo, B., et al. (2017). "Speed profile estimation using license plate recognition data." Transportation research part C: emerging technologies **82**: 358-378.

Mohseni, F., et al. (2020). "Distributed Cooperative MPC for Autonomous Driving in Different Traffic Scenarios." IEEE Transactions on Intelligent Vehicles **6**(2): 299-309.

Morris, E. A., et al. (2020). "Are drivers cool with pool? Driver attitudes towards the shared TNC services UberPool and Lyft Shared." Transport Policy **94**: 123-138.

Muehlhauser, L. and L. Helm (2013). The Singularity and Machine Ethics.

Nar, K., et al. (2017). Learning prospect theory value function and reference point of a sequential decision maker. 2017 IEEE 56th Annual Conference on Decision and Control (CDC).

Nascimento, A. M., et al. (2019). "A systematic literature review about the impact of artificial intelligence on autonomous vehicle safety." IEEE Transactions on Intelligent Transportation Systems **21**(12): 4928-4946.

Navet, N. and F. Simonot-Lion (2017). Automotive embedded systems handbook, CRC press.

Nay, J. L. and J. P. Zagal (2017). Meaning without consequence: virtue ethics and inconsequential choices in games. Proceedings of the 12th International Conference on the Foundations of Digital Games.

Nilsson, J., et al. (2016). "Longitudinal and Lateral Control for Automated Yielding Maneuvers." IEEE Transactions on Intelligent Transportation Systems **17**(5): 1404-1414.

Operto, F. (2011). "Ethics in Advanced Robotics." IEEE Robotics & Automation Magazine **18**(1): 72-78.

Organization, W. H. (2017). Global status report on road safety 2018. Geneva: World Health Organization; 2018.

Organization, W. H. (2018). Global status report on road safety 2018: summary, World Health Organization.

Paden, B., et al. (2016). A Survey of Motion Planning and Control Techniques for Self-driving Urban Vehicles.

Paden, B., et al. (2016). "A survey of motion planning and control techniques for self-driving urban vehicles." IEEE Transactions on Intelligent Vehicles **1**(1): 33-55.

Pakusch, C., et al. (2018). "Unintended effects of autonomous driving: A study on mobility preferences in the future." Sustainability (Switzerland) **10**(7).

Pan, Y., et al. (2020). "The validation of a semi-recursive vehicle dynamics model for a real-time simulation." Mechanism and Machine Theory **151**: 103907.

Perera, H., et al. (2019). Towards integrating human values into software: Mapping principles and rights of GDPR to values. 2019 IEEE 27th International Requirements Engineering Conference (RE), IEEE.

Petzoldt, T., et al. (2018). "Potential safety effects of a frontal brake light for motor vehicles." IET intelligent transport systems **12**(6): 449-453.

Peysakhovich, A. and J. Naecker (2017). "Using methods from machine learning to evaluate behavioral models of choice under risk and ambiguity." Journal of Economic Behavior & Organization **133**: 373-384.

Pimentel, J. and J. Bastiaan (2018). Characterizing the safety of self-driving vehicles: A fault containment protocol for functionality involving vehicle detection. 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), IEEE.

Ploeg, J. (2017). Cooperative Vehicle Automation: Safety Aspects and Control Software Architecture. 2017 IEEE International Conference on Software Architecture Workshops (ICSAW).

Plyley, K. (2018). Tolerated illegality and intolerable legality: from legal philosophy to critique.

Point, I., et al. "Traffic Safety Facts."

Pouya, P. and A. M. Madni (2020). "Expandable-Partially Observable Markov Decision-Process Framework for Modeling and Analysis of Autonomous Vehicle Behavior." IEEE Systems Journal.

Prakken, H. (2017). "On the problem of making autonomous vehicles conform to traffic law." Artificial Intelligence and Law **25**(3): 341-363.

Purves, D., et al. (2015). "Autonomous Machines, Moral Judgment, and Acting for the Right Reasons." Ethical Theory and Moral Practice **18**(4): 851-872.

Qian, X., et al. (2016). A hierarchical Model Predictive Control framework for on-road formation control of autonomous vehicles. 2016 IEEE Intelligent Vehicles Symposium (IV).

Rahman, M. H., et al. (2019). A Deep Learning Based Approach to Predict Sequential Design Decisions. ASME 2019 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference.

Rasekhipour, Y., et al. (2017). "A Potential Field-Based Model Predictive Path-Planning Controller for Autonomous Road Vehicles." IEEE Transactions on Intelligent Transportation Systems **18**(5): 1255-1267.

Redelmeier, D. A. and S. Raza (2017). "Life-threatening motor vehicle crashes in bright sunlight." Medicine **96**(1): e5710-e5710.

Rehman, S. and J. Dzionek-Kozłowska (2018). "The trolley problem revisited. An exploratory study."

Riaz, F., et al. (2018). "A collision avoidance scheme for autonomous vehicles inspired by human social norms." Computers & Electrical Engineering **69**: 690-704.

Rizaldi, A. and M. Althoff (2015). Formalising traffic rules for accountability of autonomous vehicles. 2015 IEEE 18th international conference on intelligent transportation systems, IEEE.

Rosen, E., et al. (2011). "Literature review of pedestrian fatality risk as a function of car impact speed." Accident Analysis & Prevention **43**(1): 25-33.

Rouse, W. B. (2017). "The Systems, Man, and Cybernetics of Driverless Cars: Challenges and Opportunities for the SMCS." IEEE Systems, Man, and Cybernetics Magazine **3**(3): 6-8.

Sadat, A., et al. (2020). Perceive, predict, and plan: Safe motion planning through interpretable semantic representations. European Conference on Computer Vision, Springer.

Sakib, N. (2020). "Highway Lane change under uncertainty with Deep Reinforcement Learning based motion planner."

Sarathy, V., et al. (2017). Learning behavioral norms in uncertain and changing contexts. 2017 8th IEEE International Conference on Cognitive Infocommunications (CogInfoCom).

Scanlon, T. (2000). What we owe to each other, Belknap Press.

Schratter, M., et al. (2019). Pedestrian collision avoidance system for scenarios with occlusions. 2019 IEEE Intelligent Vehicles Symposium (IV), IEEE.

Schroeder, B., et al. (2014). Empirically-based performance assessment & simulation of pedestrian behavior at unsignalized crossings, Southeastern Transportation Research, Innovation, Development and Education ….

Schulz, W. and K. Dankert (2016). "'Governance by Things' as a challenge to regulation by law." Internet Policy Review **5**(2): 2017-2001.

Schwarting, W., et al. (2018). "Planning and decision-making for autonomous vehicles." Annual Review of Control, Robotics, and Autonomous Systems **1**: 187-210.

Schwarting, W., et al. (2018). "Planning and Decision-Making for Autonomous Vehicles." Annual Review of Control, Robotics, and Autonomous Systems **1**(1): 187-210.

Schwarting, W., et al. (2019). "Social behavior for autonomous vehicles." Proceedings of the National Academy of Sciences **116**(50): 24972-24978.

Shaoshan, L., et al. (2017). Creating Autonomous Vehicle Systems, Morgan & Claypool.

Shapiro, I. (2018). Democracy's place, Cornell University Press.

Sheth, P. D. and A. J. Umbarkar (2015). Constrained Optimization Problems Solving Using Evolutionary Algorithms: A Review. 2015 International Conference on Computational Intelligence and Communication Networks (CICN).

Siampis, E., et al. (2017). "A real-time nonlinear model predictive control strategy for stabilization of an electric vehicle at the limits of handling." IEEE Transactions on Control Systems Technology **26**(6): 1982-1994.

Sigaud, O. and O. Buffet (2013). Markov decision processes in artificial intelligence, John Wiley & Sons.

Singh, S., et al. (2018). "Robust tracking with model mismatch for fast and safe planning: an SOS optimization approach." arXiv preprint arXiv:1808.00649.

Soin, A. and M. Chahande (2017). Moving vehicle detection using deep neural network. 2017 International Conference on Emerging Trends in Computing and Communication Technologies (ICETCCT).

Son, S. H., et al. (2020). "Idle speed control with low-complexity offset-free explicit model predictive control in presence of system delay." arXiv preprint arXiv:2012.02859.

Soriguera, F., et al. (2017). "Effects of low speed limits on freeway traffic flow." Transportation research part C: emerging technologies **77**: 257-274.

Sovacool, B. K., et al. (2017). "New frontiers and conceptual frameworks for energy justice." Energy Policy **105**: 677-691.

Stower, H. (2019). "Investigating the moral machine." Nature Medicine **25**(1): 19-19.

Suh, S. H. and A. B. Bishop (1988). "Collision-avoidance trajectory planning using tube concept: Analysis and simulation." Journal of robotic systems **5**(6): 497-525.

Sunberg, Z. N. and M. J. Kochenderfer (2018). Online algorithms for POMDPs with continuous state, action, and observation spaces. Twenty-Eighth International Conference on Automated Planning and Scheduling.

Symons, X. (2019). Meeting needs and respecting persons: An ethical framework for the allocation of lifesaving healthcare interventions, Australian Catholic University.

Taeihagh, A. and H. S. M. Lim (2019). "Governing autonomous vehicles: emerging responses for safety, liability, privacy, cybersecurity, and industry risks." Transport Reviews **39**(1): 103-128.

Taiebat, M., et al. (2018). "A review on energy, environmental, and sustainability implications of connected and automated vehicles." Environmental science & technology **52**(20): 11449-11465.

Takapoui, R., et al. (2020). "A simple effective heuristic for embedded mixed-integer quadratic programming." International journal of control **93**(1): 2-12.

Talbot, B., et al. (2017). "When robots should do the wrong thing." Robot ethics **2**: 258-273.

Talhelm, T., et al. (2015). "Liberals think more analytically (more "WEIRD") than conservatives." Personality and Social Psychology Bulletin **41**(2): 250-267.

Tanelli, M., et al. (2018). "Guest Editorial: Multifaceted Driver–Vehicle Systems: Toward More Effective Driving Simulations, Reliable Driver Modeling, and Increased Trust and Safety." IEEE Transactions on Human-Machine Systems **48**(1): 1-5.

Taramov, A. and N. Shilov (2017). A systematic review of proactive driver support systems and underlying technologies. 2017 20th Conference of Open Innovations Association (FRUCT).

Terziyan, V., et al. (2018). "Patented intelligence: Cloning human decision models for Industry 4.0." Journal of Manufacturing Systems.

Tjolleng, A., et al. (2017). "Classification of a Driver's cognitive workload levels using artificial neural network on ECG signals." Applied ergonomics **59**: 326-332.

Tronto, J. C. (2020). Moral boundaries: A political argument for an ethic of care, Routledge.

Tuncali, C. E. and G. Fainekos (2019). Rapidly-exploring random trees for testing automated vehicles. 2019 IEEE Intelligent Transportation Systems Conference (ITSC), IEEE.

Turri, V., et al. (2013). Linear model predictive control for lane keeping and obstacle avoidance on low curvature roads. 16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013).

Umbrello, S. and A. F. De Bellis (2018). "A value-sensitive design approach to intelligent agents." Artificial Intelligence Safety and Security (2018) CRC Press (. ed) Roman Yampolskiy.

Urooj, S., et al. (2018). Systematic literature review on user interfaces of autonomous cars: Liabilities and responsibilities. 2018 International Conference on Advancements in Computational Sciences (ICACS).

Van den Hoven, J., et al. (2012). "Engineering and the Problem of Moral Overload." Science and Engineering Ethics **18**(1): 143-155.

Van Wynsberghe, A. (2013). "Designing robots for care: Care centered value-sensitive design." Science and Engineering Ethics **19**(2): 407-433.

Van Wynsberghe, A. and S. Robbins (2014). "Ethicist as designer: a pragmatic approach to ethics in the lab." Science and Engineering Ethics **20**(4): 947-961.

Vanderelst, D. and A. Winfield (2018). The dark side of ethical robots. Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society.

Vasile, C.-I., et al. (2017). Minimum-violation scLTL motion planning for mobility-on-demand. 2017 IEEE International Conference on Robotics and Automation (ICRA), IEEE.

Wagner, A., et al. (2019). Kantian one day, Consequentialist the next: Moral emotions as mediators between ethical frameworks for robots. 2019 Conference of the International Association for Computing and Philosophy (IACAP 2019).

Wallace, R. J. (2019). The moral nexus, Princeton University Press.

Wang, Q., et al. (2018). Predictive Maneuver Planning for an Autonomous Vehicle in Public Highway Traffic.

Wang, Y., et al. (2012). On Studying Relationship between Altruism and the Psychological Phenomenon of Self-Deception in Rational and Autonomous Networks. 2012 32nd International Conference on Distributed Computing Systems Workshops.

Wei, J., et al. (2010). A prediction-and cost function-based algorithm for robust autonomous freeway driving. 2010 IEEE Intelligent Vehicles Symposium, IEEE.

Weijermars, W., et al. (2018). "Serious road traffic injuries in europe, lessons from the eu research project safetycube." Transportation Research Record **2672**(32): 1-9.

Wilson, J. Q. (1993). "The moral sense." American Political Science Review **87**(1): 1-11.

Wintersberger, P., et al. (2017). Do moral robots always fail? Investigating human attitudes towards ethical decisions of automated systems. 2017 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN).

Wong, D. B. (2020). Moral relativity, University of California Press.

Wulfmeier, M., et al. (2016). Watch this: Scalable cost-function learning for path planning in urban environments. 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE.

Xu, J., et al. (2019). "Destination Prediction A Deep Learning based Approach." IEEE Transactions on Knowledge and Data Engineering: 1-1.

Xu, P., et al. (2018). "System Architecture of a Driverless Electric Car in the Grand Cooperative Driving Challenge." IEEE Intelligent Transportation Systems Magazine **10**(1): 47-59.

Yi, B., et al. (2016). Real time integrated vehicle dynamics control and trajectory planning with MPC for critical maneuvers. 2016 IEEE Intelligent Vehicles Symposium (IV).

Yilmaz, O. and S. A. Saribay (2017). "Activating analytic thinking enhances the value given to individualizing moral foundations." Cognition **165**: 88-96.

Yoon, Y., et al. (2009). "Model-predictive active steering and obstacle avoidance for autonomous ground vehicles." Control Engineering Practice **17**(7): 741-750.

Yurtsever, E., et al. (2020). "A survey of autonomous driving: Common practices and emerging technologies." IEEE Access **8**: 58443-58469.

Zachko, O., et al. (2019). Models of safety management in development projects. 2019 IEEE 14th International Conference on Computer Sciences and Information Technologies (CSIT), IEEE.

Zawieska, K. (2020). "Disengagement with ethics in robotics as a tacit form of dehumanisation." AI & SOCIETY **35**(4): 869-883.

Zhang, H., et al. (2019). "A novel machine learning based approach for iPS progenitor cell identification." PLOS Computational Biology **15**(12): e1007351.

Zhang, K., et al. (2015). Computationally aware control of autonomous vehicles: a hybrid model predictive control approach.

Zhang, X., et al. (2018). Behavioral cloning for driverless cars using transfer learning. 2018 IEEE/ION Position, Location and Navigation Symposium (PLANS).

Ziegler, J., et al. (2014). Trajectory planning for Bertha—A local, continuous method. 2014 IEEE intelligent vehicles symposium proceedings, IEEE.

Appendixes

Appendix-1: Source code for simulation Id

```
classdef getSimId < matlab.System & matlab.system.mixin.Propagates
    % Get Simulation Id

    % Public, tunable properties
    properties
        %NumberOfVehicles Maximum number of vehicles
        NumberOfVehicles = 0;

        %NumEachVehicle Maximum number of each type of vehicles
        NumEachVehicle = zeros (6, 1);
    end

    properties (DiscreteState)

    end

    % Pre-computed constants
    Properties (Access = private)
        adasData_;
    end

    methods (Access = protected)
        function Id = getAvaiableId(obj, vehType)
            for i = 1: obj.NumberOfVehicles
                if ((obj.adasData_(i).Type == vehType) && ...
                        (obj.adasData_(i).Id <= 0))
                    Id = i;
                    return;
                end
            end
            disp('Too many vehicles');
            return;
        end
        function setupImpl(obj)
            % Perform one-time calculations, such as computing constants
            obj.adasData_(obj.NumberOfVehicles).Type = VehType.None;
            obj.adasData_(obj.NumberOfVehicles).Id = 0;
            for i = obj.NumberOfVehicles : -1 : 1
                obj.adasData_(i) = obj.adasData_(end);
                if i <= obj.NumEachVehicle(1)
                    obj.adasData_(i).Type = VehType(1);
                elseif i <= sum (obj.NumEachVehicle(1 : 2))
                    obj.adasData_(i).Type = VehType(2);
                end
            end

        end

        function y = stepImpl (obj, vehType, vehId)
            % Implement algorithm. Calculate y as a function of input u and
            % discrete states.
            if (vehId < 0)
                idx = -vehId;
                obj.adasData_(idx).Id = -idx;
                y = idx;
```

```
            else
                idx = obj.getAvaiableId(vehType);
                obj.adasData_(idx).Id = idx;
                y = idx;
            end
        end

        function sz = getOutputSizeImpl(obj)
            sz = 1;
        end

        function c = isOutputFixedSizeImpl(obj)
            c = true;
        end

        function c = isOutputComplexImpl(obj)
            c = false;
        end

        function c = getOutputDataTypeImpl(obj)
            c = 'double';
        end

        function resetImpl(obj)
            % Initialize / reset discrete-state properties
        end
    end
end
```

Appendix-2: Source code for vehicle type

```
classdef VehType < Simulink.IntEnumType
  enumeration
    None (0)
    Veh1(1)
    Veh2(2)
  end
end
```

Appendix-3: Source code for safety check

```
classdef SafetyCheck < matlab.System & matlab.system.mixin.Propagates
    & matlab.system.mixin.CustomIcon

    % This template includes the minimum set of functions required
    % to define a System object with discrete state.
    % Public, tunable properties
    properties
        % Safety Distance [m]
        safeDistance = 75;

        % Average Vehicle Length [m]
        aveVehLength = 6;

        % Max Accelerating Rate [m/s^2]
        maxAccelRate = 3;
```

```
    % Max Deaccelerating Rate [m/s^2]
    maxDeaccelRate = 10;

    % Lane Width [m]
    LaneWidth = 7;
end

properties (DiscreteState)

end

% Pre-computed constants
properties (Access = private)

end

methods (Access = protected, Static)
    function header = getHeaderImpl
        % Define header panel for System block dialog
        header = matlab.system.display.Header( ...
            'Title', "Safety Check", ...
            'Text','Check whether it is safe to change lane.');
    end
end

methods (Access = protected)

    function flag = supportsMultipleInstanceImpl(obj)
        flag = true;
    end

    function icon = getIconImpl(~)
        icon = {"Safety Check"};
    end

    function setupImpl(obj)
        % Perform one-time calculations, such as computing constants
    end

    function y = stepImpl(obj, u)
        % Implement algorithm. Calculate y as a function of input u and
        % discrete states.

        if (u.exist == false)
            y = 0;
            return;
        end

        y = 1;
```

```
            % If can switch lane, the time needed is about
            % Assume ay = 1/3*a;
            t_est = sqrt(3*obj.LaneWidth/obj.maxAccelRate);

            %% check front vehicle
            if (u.frontDistance < obj.safeDistance)
                %false if front vehicle exists
                a_front_mx = 2*(u.frontDistance + u.frontVelocity*t_est -
obj.safeDistance/5)/t_est^2;
                if a_front_mx < -obj.maxDeaccelRate/1 || u.frontDistance <
1.5*obj.aveVehLength
                    % if current vehicle does not have enough brake
                    y = 0;
                    return;
                end
            end

            %% check rear vehicle
            if (u.rearDistance < obj.safeDistance)
                % if rear vehicle exists
                a_rear_mn = 2*(-u.rearDistance + u.rearVelocity*t_est +
obj.safeDistance/5)/t_est^2;
                if a_rear_mn > obj.maxAccelRate/1.5 || u.rearDistance <
1.5*obj.aveVehLength
                    % if current vehicle does not have enough power
                    y = 0;
                    return;
                end
            end

        end

        function sz_1 = getOutputSizeImpl(obj)
            sz_1 = 1;
        end

        function c1 = isOutputFixedSizeImpl(obj)
            c1 = true;
        end

        function c1 = isOutputComplexImpl(obj)
            c1 = false;
        end

        function c1 = getOutputDataTypeImpl(obj)
            c1 = 'double';
        end

        function resetImpl(obj)
            % Initialize / reset discrete-state properties
```

```
        end
    end
end
```

Appendix-4: Source code for change lane

```
classdef ChangeLane < Simulink.IntEnumType
  enumeration
    None (0)
    Left (1)
    Right (2)
  end
end
```

Appendix-5: Source code for helper plot lane curve

```
classdef (StrictDefaults)helperPlotLaneCurve < matlab.System &
matlab.system.mixin.CustomIcon
    % This is a helper block and may be modified in the future.

    %#codegen

    Properties (Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [0 200]

        %YLim Y range
        YLim = [-100 50]
    end

    properties (Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties (Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneCurve(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set(obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);

            theta = linspace(pi/2, 3*pi/4, 1000);
```

```
            radius = 100;
            laneWidth = 7;
            center = [100, -100];

            plot (obj.hAxis, center(1) + (radius - laneWidth/2)*cos(theta),
center(2) + (radius - laneWidth/2)*sin(theta), 'linewidth', 2, 'color',
'k');
            plot (obj.hAxis, center(1) + (radius + laneWidth/2)*cos(theta),
center(2) + (radius + laneWidth/2)*sin(theta), 'linewidth', 2, 'color',
'k');
            plot (obj.hAxis, [100 200], [-laneWidth/2 -laneWidth/2],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [100 200], [laneWidth/2 laneWidth/2],
'linewidth', 2, 'color', 'k')

            mPosition_X = varargin{1};

            for i = length(mPosition_X): -1: 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1: length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
```

```
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                        continue;
                    end

                    if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
                        continue;
                    end

                    curPosition = zeros (2, 1);
                    curPosition(1) = mPosition_X(i) - 5/2;
                    curPosition(2) = mPosition_Y(i) - 3/2;

                    cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                    mbasicPos = rotationMatrix * obj.basicPos;

                    markerFaceColor = 'g';

                    switch mType(i)
                        case VehType(1)
                            markerFaceColor = 'b';
                        case VehType(2)
                            markerFaceColor = 'r';
                    end

                    set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                        'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                    if obj.showVelocity
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                    end

                    if obj.showAcceleration
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
```

```
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                    end
                end
            end
        end
    end

    % Simulink interface
    methods(Access=protected)
        function str = getIconImpl(~)
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
    end
end
```

```
function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-6: Source code for helper plot lane merge

```
classdef (StrictDefaults)helperPlotLaneMerge < matlab.System &
matlab.system.mixin.CustomIcon
    %
    % This is a helper block and may be removed or modified in the future.

    %#codegen

    properties(Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [0 300]

        %YLim Y range
        YLim = [-150  100]
    end

    properties(Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties(Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneMerge(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set (obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);
```

```
            plot (obj.hAxis, [0, 173 - 7], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 + 7 300], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 - 7 173 + 7], [0 0], 'linewidth', 2,
'color', 'k', 'linestyle', '--')
            plot (obj.hAxis, [0 300], [7 7], 'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [0 173 - 7], [-100 + 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [0 173 + 7], [-100 - 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')
            mPosition_X = varargin{1};

            for i = length(mPosition_X) : -1 : 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1 : length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
```

```
                        'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                    continue;
                end

                if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
                    continue;
                end

                curPosition = zeros(2, 1);
                curPosition(1) = mPosition_X(i) - 5/2;
                curPosition(2) = mPosition_Y(i) - 3/2;

                cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                mbasicPos = rotationMatrix * obj.basicPos;

                markerFaceColor = 'g';

                switch mType(i)
                    case VehType(1)
                        markerFaceColor = 'b';
                    case VehType(2)
                        markerFaceColor = 'r';
                end

                set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                    'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                if obj.showVelocity
                    set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                        'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                end

                if obj.showAcceleration
                    set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                        'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                end
            end
        end
```

```
        end
    end

    % Simulink interface
    Methods (Access=protected)
        function str = getIconImpl(~)
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
    end
end

function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-7: Source code for helper plot lane switch

```
classdef (StrictDefaults)helperPlotLaneSwitch < matlab.System &
matlab.system.mixin.CustomIcon
    %
    % This is a helper and may be removed or modified in the future.

    %#codegen

    properties (Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [0 120]

        %YLim Y range
        YLim = [-50 50]
    end

    properties (Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties (Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneSwitch(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods(Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set (obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);
```

```
            plot (obj.hAxis, [0 120], [-7 -7], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [0 120], [0 0], 'linewidth', 2, 'color', 'k',
'linestyle', '--')
            plot (obj.hAxis, [0 120], [7 7], 'linewidth', 2, 'color', 'k')

            mPosition_X = varargin{1};

            for i = length(mPosition_X) : -1 : 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1 : length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                        continue;
                    end

                    if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
```

```
                        continue;
                    end

                    curPosition = zeros(2, 1);
                    curPosition(1) = mPosition_X(i) - 5/2;
                    curPosition(2) = mPosition_Y(i) - 3/2;

                    cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                    mbasicPos = rotationMatrix * obj.basicPos;

                    markerFaceColor = 'g';

                    switch mType(i)
                        case VehType(1)
                            markerFaceColor = 'b';
                        case VehType(2)
                            markerFaceColor = 'r';
                    end

                    set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                        'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                    if obj.showVelocity
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                    end

                    if obj.showAcceleration
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                    end
                end
            end
        end
    end

    % Simulink interface
    methods (Access=protected)
        function str = getIconImpl(~)
```

```
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
    end
end

function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-8: Source code for helper plot lane switch and merge

```
classdef (StrictDefaults)helperPlotLaneSwitchAndMerge < matlab.System &
matlab.system.mixin.CustomIcon

    % This is a helper block and may be removed or modified in the future.

    %#codegen

    Properties (Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [0 300]

        %YLim Y range
        YLim = [-150 100]
    end

    properties (Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties (Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneSwitchAndMerge(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set(obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);
```

```
            plot (obj.hAxis, [0 173 - 7], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 + 7 300], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 - 7 173 + 7], [0 0], 'linewidth', 2,
'color', 'k', 'linestyle', '--')
            plot (obj.hAxis, [0 300], [7 7], 'linewidth', 2, 'color', 'k',
'linestyle', '--')
            plot (obj.hAxis, [0 300], [14 14], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [0 173 - 7], [-100 + 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [0 173 + 7], [-100 - 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')

            mPosition_X = varargin {1};

            for i = length(mPosition_X): -1: 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1: length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
```

```
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                        continue;
                    end

                    if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
                        continue;
                    end

                    curPosition = zeros (2, 1);
                    curPosition(1) = mPosition_X(i) - 5/2;
                    curPosition(2) = mPosition_Y(i) - 3/2;

                    cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                    mbasicPos = rotationMatrix * obj.basicPos;

                    markerFaceColor = 'g';

                    switch mType(i)
                        case VehType(1)
                            markerFaceColor = 'b';
                        case VehType(2)
                            markerFaceColor = 'r';
                    end

                    set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                        'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                    if obj.showVelocity
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                    end

                    if obj.showAcceleration
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
```

```
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                    end
                end
            end
        end
    end

    % Simulink interface
    Methods (Access=protected)
        function str = getIconImpl(~)
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
    end
end
```

```
function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-9: Source code for helper plot lane curve-1

```
classdef (StrictDefaults)helperPlotLaneCurve1 < matlab.System &
matlab.system.mixin.CustomIcon

    % This is a helper block and may be removed or modified in the future.

    %#codegen

    Properties (Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [-220, 220]

        %YLim Y range
        YLim = [-200, 130]
    end

    properties (Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties (Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneCurve1(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set(obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);
```

```
            theta = linspace (0, pi, 1000);
            radius = 100;
            laneWidth = 7;

            center1 = [-radius - laneWidth, 0];
            center2 = [radius + laneWidth, 0];

            plot (obj.hAxis, center1(1) + (radius -
laneWidth/2)*cos(theta), center1(2) + (radius - laneWidth/2)*sin(theta),
'linewidth', 2, 'color', 'k');
            plot (obj.hAxis, center1(1) + (radius +
laneWidth/2)*cos(theta), center1(2) + (radius + laneWidth/2)*sin(theta),
'linewidth', 2, 'color', 'k');
            plot (obj.hAxis, center2(1) + (radius -
laneWidth/2)*cos(theta), center2(2) + (radius - laneWidth/2)*sin(theta),
'linewidth', 2, 'color', 'k');
            plot (obj.hAxis, center2(1) + (radius +
laneWidth/2)*cos(theta), center2(2) + (radius + laneWidth/2)*sin(theta),
'linewidth', 2, 'color', 'k');

            plot (obj.hAxis, [-laneWidth/2 -laneWidth/2], [0 120],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [laneWidth/2 laneWidth/2], [0 120],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [-laneWidth/2 -laneWidth/2], [-200 0],
'linewidth', 2, 'color', 'k', 'linestyle', '--')
            plot (obj.hAxis, [laneWidth/2 laneWidth/2], [-200 0],
'linewidth', 2, 'color', 'k', 'linestyle', '--')
            plot (obj.hAxis, [-3*laneWidth/2 -3*laneWidth/2], [-200 0],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [3*laneWidth/2 3*laneWidth/2], [-200 0],
'linewidth', 2, 'color', 'k')

            mPosition_X = varargin{1};

            for i = length(mPosition_X): -1: 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
```

```
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1: length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                        continue;
                    end

                    if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
                        continue;
                    end

                    curPosition = zeros (2, 1);
                    curPosition (1) = mPosition_X(i) - 5/2;
                    curPosition (2) = mPosition_Y(i) - 3/2;

                    cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                    mbasicPos = rotationMatrix * obj.basicPos;

                    markerFaceColor = 'g';

                    switch mType(i)
                        case VehType(1)
                            markerFaceColor = 'b';
                        case VehType(2)
                            markerFaceColor = 'r';
                    end
```

```
                    set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                        'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                    if obj.showVelocity
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                    end

                    if obj.showAcceleration
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                    end
                end
            end
        end
    end

    % Simulink interface
    methods(Access=protected)
        function str = getIconImpl(~)
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
```

```
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
    end
end

function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-10: Source code for helper plot lane switch and merge-2

```
classdef (StrictDefaults)helperPlotLaneSwitchAndMerge2 < matlab.System &
matlab.system.mixin.CustomIcon

    % This is a helper block and may be removed or modified in the future.

    %#codegen

    Properties (Nontunable, SetAccess = immutable)
        %XLim X range
        XLim = [0 300]

        %YLim Y range
        YLim = [-150 100]
    end

    properties (Nontunable, Logical)
        %ShowVelocity Display Velocity
        showVelocity = false

        %ShowAcceleration Display Acceleration
        showAcceleration = false
    end

    properties (Access=private)
        hFig
        hAxis
        hGlyph
        hText
        hTexta

        basicPos = [-2.5, 2.5, 2.5, -2.5; -1.5, -1.5, 1.5, 1.5]
    end

    methods
        function obj = helperPlotLaneSwitchAndMerge2(varargin)
            % Constructor
            setProperties(obj,nargin,varargin{:});
        end
    end

    methods (Access=protected)
        function setupImpl(obj,varargin)
            obj.hFig = figure;
            obj.hAxis = axes;
            set (obj.hFig, 'position', [100 100 840 640])
            hold on;
            axis equal
            xlim(obj.XLim);
            ylim(obj.YLim);
```

```
            plot (obj.hAxis, [0 173 - 7], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 + 7 300], [0 0], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [173 - 7 173 + 7], [0 0], 'linewidth', 2,
'color', 'k', 'linestyle', '--')
            plot (obj.hAxis, [0 300], [7 7], 'linewidth', 2, 'color', 'k',
'linestyle', '--')
            plot (obj.hAxis, [0 300], [14 14], 'linewidth', 2, 'color',
'k')
            plot (obj.hAxis, [0 173 - 7], [-100 + 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [0 173 + 7], [-100 - 7/sqrt(3) 0],
'linewidth', 2, 'color', 'k')
            plot(obj.hAxis, [30/tan(pi/6) 30/tan(pi/6) + 70/tan(pi/9) +
7/(2*sin(pi/9))], [-70 - 7/(2*cos(pi/9)) 0], 'linewidth', 2, 'color', 'k')
            plot (obj.hAxis, [88.085 30/tan(pi/6) + 70/tan(pi/9) -
7/(2*sin(pi/9))], [-53.127 0], 'linewidth', 2, 'color', 'k')

            mPosition_X = varargin{1};

            for i = length(mPosition_X): -1: 1
                obj.hGlyph(i) = patch('XData', [0, 5, 5, 0], 'YData', [-1.5
-1.5 1.5 1.5], 'visible', 'off');

                markerFaceColor = 'g';

                set(obj.hGlyph(i), 'FaceColor', markerFaceColor);
                obj.hText(i) = text(0, 0, '0', 'visible', 'off');
                obj.hTexta(i) = text(0, 0, '0', 'visible', 'off');
            end
        end

        function stepImpl(obj,varargin)
            %Update the Simulink control toolbar

            mPosition_X = varargin{1};
            mPosition_Y = varargin{2};
            mVelocity_X = varargin{3};
            mVelocity_Y = varargin{4};
            mAcceleration_X = varargin{5};
            mAcceleration_Y = varargin{6};
            mId = varargin{7};
            mType = varargin{8};

            % Update the plot if it is visible
            if strcmp(get(obj.hFig,'Visible'),'on')
                for i = 1: length(mPosition_X)
                    if (mId(i) <= 0)
                        set(obj.hGlyph(i), 'XData', obj.basicPos(1, :) +
mPosition_X(i), ...
```

```
                            'YData', obj.basicPos(2, :) + mPosition_Y(i),
'visible', 'off');
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'off');
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'off');
                        continue;
                    end

                    if (sqrt(mVelocity_X(i)^2 + mVelocity_Y(i)^2) < 5e-2)
                        continue;
                    end

                    curPosition = zeros (2, 1);
                    curPosition(1) = mPosition_X(i) - 5/2;
                    curPosition(2) = mPosition_Y(i) - 3/2;

                    cosTheta = mVelocity_X(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    sinTheta = mVelocity_Y(i)/sqrt(mVelocity_X(i)^2 +
mVelocity_Y(i)^2);
                    rotationMatrix = [cosTheta, -sinTheta; sinTheta,
cosTheta];

                    mbasicPos = rotationMatrix * obj.basicPos;

                    markerFaceColor = 'g';

                    switch mType(i)
                        case VehType(1)
                            markerFaceColor = 'b';
                        case VehType(2)
                            markerFaceColor = 'r';
                    end

                    set(obj.hGlyph(i), 'XData', mbasicPos(1, :) +
mPosition_X(i), ...
                            'YData', mbasicPos(2, :) + mPosition_Y(i),
'visible', 'on', 'FaceColor', markerFaceColor);

                    if obj.showVelocity
                        set(obj.hText(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 5 0], ...
                            'String', [num2str(round(mVelocity_X(i), 2)), '
', num2str(round(mVelocity_Y(i), 2))], 'visible', 'on');
                    end
```

```
                    if obj.showAcceleration
                        set(obj.hTexta(i), 'Position', [mPosition_X(i)
mPosition_Y(i) + 10 0], ...
                            'String', [num2str(round(mAcceleration_X(i),
2)), ' ', num2str(round(mAcceleration_Y(i), 2))], 'visible', 'on');
                    end
                end
            end
        end
    end

    % Simulink interface
    Methods (Access=protected)
        function str = getIconImpl(~)
            str = sprintf('Simulation\n\nVisualization');
        end

        function num = getNumInputsImpl(obj)
            num = 8;
        end

        function varargout = getInputNamesImpl(obj)
            varargout = {};

            varargout = {varargout{:} 'Position_X'};
            varargout = {varargout{:} 'Position_Y'};
            varargout = {varargout{:} 'Velocity_X'};
            varargout = {varargout{:} 'Velocity_Y'};
            varargout = {varargout{:} 'Acceleration_X'};
            varargout = {varargout{:} 'Acceleration_Y'};
            varargout = {varargout{:} 'Id'};
            varargout = {varargout{:} 'Type'};
        end
    end

    methods (Access = protected, Static)
        function header = getHeaderImpl
            % Define header panel for System block dialog
            header = matlab.system.display.Header(...
                'Title', 'SimulationVisualization', ...
                'Text', getHeaderText());
        end

        function simMode = getSimulateUsingImpl
            % Return only allowed simulation mode in System block dialog
            simMode = 'Interpreted execution';
        end

        function flag = showSimulateUsingImpl
            % Return false if simulation mode hidden in System block dialog
            flag = false;
        end
```

```
    end
end

function str = getHeaderText
str = sprintf([...
    'Plot simulation results.']);
end
```

Appendix-11: Source code for utilities forcing step

```
function MinimalDistance = fcn(u)

idx = 1;
tx = zeros(length(u), 1);
ty = zeros(length(u), 1);

for i = 1: length(u)
    if (u(i). Id > 0)
        tx(idx) = u(i). Position(1);
        ty(idx) = u(i). Position(2);
        idx = idx + 1;
    end
end

x = tx(1:idx - 1);
y = ty(1:idx - 1);

if (isempty(x))
    MinimalDistance = 0;
    return;
end

len = length(x);
distance = zeros(len*(len-1)/2,1);

idx = 1;
for i = 1: len
    for j = i + 1: len
        distance(idx) = sqrt((x(i) - x(j))^2 + (y(i) - y(j))^2);
        idx = idx + 1;
    end
end

if isempty(distance)
    MinimalDistance = 1000;
else
    MinimalDistance = min(distance);
end
```

Appendix-12: Source code of sensor model for single lane curve scenario

```
%
% Sensor model for single lane curve scenario

LaneWidth = 7;

centerPoint = [100; -100];
radius = 100;

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo

if mCurVehInfo.Position(1) <= 100
    mLocalVehInfo.dynamicInfo.lateralDeviation = norm(mCurVehInfo.Position
- centerPoint) - radius;
    tRelPos = mCurVehInfo.Position - centerPoint;
    tRelPos = tRelPos/norm(tRelPos);
    mLocalVehInfo.curLane.curvature = [tRelPos(2); -tRelPos(1)];
else
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
0;
    mLocalVehInfo.curLane.curvature = [1; 0];
end

%% VehInput.curLane/leftLane/rightLane

mCurRadian = atan2(mCurVehInfo.Position(2) - centerPoint(2),
mCurVehInfo.Position(1) - centerPoint(1));

for i = 1: length(mGlobalInfo)
    if mGlobalInfo(i). Id <= 0
        % Vehicle is disabled
        continue;
    end

    if mGlobalInfo(i).Position(1) <= radius && mCurVehInfo.Position(1) <=
radius

        mSensoredRadian = atan2(mGlobalInfo(i).Position(2) -
centerPoint(2), mGlobalInfo(i).Position(1) - centerPoint(1));

        if mSensoredRadian < mCurRadian ...
                && radius*(mCurRadian - mSensoredRadian) <
mLocalVehInfo.curLane.frontDistance
            % Sensored vehicle is in front of the current vehicle & with a
```

```
            % smaller distance
            mLocalVehInfo.curLane.frontDistance = radius*(mCurRadian -
mSensoredRadian);
            mLocalVehInfo.curLane.frontVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

        elseif mSensoredRadian > mCurRadian ...
                && radius*(mSensoredRadian - mCurRadian) <
mLocalVehInfo.curLane.rearDistance
            % Sensored vehicle is in rear of the current vehicle and with a
            % smaller distance
            mLocalVehInfo.curLane.rearDistance = radius*(mSensoredRadian -
mCurRadian);
            mLocalVehInfo.curLane.rearVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

        end
    elseif mGlobalInfo(i).Position(1) > radius && mCurVehInfo.Position(1) >
radius
        if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                && (mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1)) <
mLocalVehInfo.curLane.frontDistance
            % Vehicle in front of current vehicle and with a smaller
            % distance
            mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
            mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
        elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                && (mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1)) <
mLocalVehInfo.curLane.rearDistance
            % Vehicle in rear of current vehicle and with a smaller
            % distance
            mLocalVehInfo.curLane.rearDistance = mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1);
            mLocalVehInfo.curLane.rearVelocity = mGlobalInfo(i).Velocity(1)
- mCurVehInfo.Velocity(1);
        end
    end

end
```

Appendix-13: Source code of sensor model for two lane merge scenarios

```
%
% Sensor model for two lane merge scenarios

LaneWidth = 7;

mergePoint = [173 + LaneWidth/2*cos(pi/6); LaneWidth/2*sin(pi/6)];
mergeRange = 173 + LaneWidth/2/sin(pi/6)*[-1, 1];

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo
if mCurVehInfo.Destination > 0
    % Vehicle's destination is the first lane
    if mCurVehInfo.Position(2) < 0
        % Vehicle is in the side lane -> change left
        mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
    else
        mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
    end
end

if mCurVehInfo.Position(2) >= 0
    % Vehicle is in the main lane
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
LaneWidth/2;
else
    % Vehicle is in the side lane
    % assume side lane is 30 degree with main lane
    mLocalVehInfo.curLane.curvature = [cos(pi/6); sin(pi/6)];

    vNormal = [-mLocalVehInfo.curLane.curvature(2);
mLocalVehInfo.curLane.curvature(1)];
    vLateralDeviation = dot(mCurVehInfo.Position - mergePoint,
vNormal)*vNormal;
    temp = cross([mLocalVehInfo.curLane.curvature; 0], [vLateralDeviation;
0]);
    mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(vLateralDeviation)*temp(3);
end

%% VehInput.curLane/leftLane/rightLane
if mCurVehInfo.Position(2) < 0 && mCurVehInfo.Position(2) > -LaneWidth ...
        && mCurVehInfo.Position(1) >= mergeRange(1) &&
mCurVehInfo.Position(1) <= mergeRange(2)
    mLocalVehInfo.leftLane.exist = true;
end
```

```
for i = 1: length(mGlobalInfo)
    if mGlobalInfo(i). Id <= 0
        % Vehicle is disabled
        continue;
    end

    if mCurVehInfo.Position(2) >= 0
        % Vehicle is in the first lane
        if mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the first lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        else
            % Comment below codes because vehicles on main lane only
            % consider those on the main lane
        end
    else
        % Vehicle in the second lane

        % treat the mergePoint as front objection(vehicle) with 0 velocity

        if norm(mCurVehInfo.Position - mergePoint) <
mLocalVehInfo.curLane.frontDistance
            mLocalVehInfo.curLane.frontDistance = norm(mCurVehInfo.Position
- mergePoint);
            mLocalVehInfo.curLane.frontVelocity = -
norm(mCurVehInfo.Velocity);
        end

        if mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the first lane

            sensoredVehToMerge = mGlobalInfo(i).Position(1) -
mergePoint(1);
```

```
            curVehToMerge = -norm(mCurVehInfo.Position - mergePoint);

            if sensoredVehToMerge > curVehToMerge ...
                    && (sensoredVehToMerge - curVehToMerge) <
mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance = sensoredVehToMerge -
curVehToMerge;
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif sensoredVehToMerge < curVehToMerge ...
                    && (curVehToMerge - sensoredVehToMerge) <
mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance = curVehToMerge -
sensoredVehToMerge;
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        else
            % Sensored vehicle is in the second lane too

            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && norm(mGlobalInfo(i).Position - mCurVehInfo.Position)
< mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
norm(mGlobalInfo(i).Position - mCurVehInfo.Position);
                mLocalVehInfo.curLane.frontVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && norm(mCurVehInfo.Position - mGlobalInfo(i).Position)
< mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
norm(mCurVehInfo.Position - mGlobalInfo(i).Position);
                mLocalVehInfo.curLane.rearVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        end
    end
end
```

Appendix-14: Source code of sensor model for two lane switch scenarios

```
%
% Sensor model for two lane switch scenarios

LaneWidth = 7;

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo
if mCurVehInfo.Destination > 0
    if mCurVehInfo.Destination == 1
        % Vehicle's destination is the first lane
        if mCurVehInfo.Position(2) < 0
            % Vehicle is in the second lane -> change right
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    else
        % Vehicle's destination is the second lane
        if mCurVehInfo.Position(2) > 0
            % Vehicle is in the first lane -> change right
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Right;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    end
end

if mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
LaneWidth/2;
else
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) +
LaneWidth/2;
end

%% VehInput.curLane/leftLane/rightLane
if mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.rightLane.exist = true;
else
    mLocalVehInfo.leftLane.exist = true;
end

for i = 1: length(mGlobalInfo)
    if mGlobalInfo(i). Id <= 0
        % Vehicle is disabled
```

```
        continue;
    end

    if mCurVehInfo.Position(2) >= 0
        % Vehicle is in the first lane
        if mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the first lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        else
            % Sensored vehicle is in the second lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.rightLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.rightLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.rightLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.rightLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.rightLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.rightLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    else
        % Vehicle in the second lane
        if mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the first lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
```

```
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        else
            % Sensored vehicle is in the second lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    end
end
```

Appendix-15: Source code of sensor model for three lane switch and lane merge scenarios

```
%
% Sensor model for three lane switch and lane merge scenarios

LaneWidth = 7;

mergePoint = [173 + LaneWidth/2*cos(pi/6); LaneWidth/2*sin(pi/6)];
mergeRange = 173 + LaneWidth/2/sin(pi/6)*[-1, 1];

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo
if mCurVehInfo.Destination > 0
    if mCurVehInfo.Destination == 1
        % Vehicle's destination is the first lane
        if mCurVehInfo.Position(2) < LaneWidth
            % Vehicle is in the second/third lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    elseif mCurVehInfo.Destination == 2
        % Vehicle's destination is the second lane
        if mCurVehInfo.Position(2) >= LaneWidth
            % Vehicle is in the first lane -> change right
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Right;
        elseif mCurVehInfo.Position(2) < 0
            % Vehicle is in the third lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    end
end

if mCurVehInfo.Position(2) >= LaneWidth
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
3*LaneWidth/2;
elseif mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
LaneWidth/2;
else
    % assume side lane is 30 degree with main lane
    mLocalVehInfo.curLane.curvature = [cos(pi/6); sin(pi/6)];

    vNormal = [-mLocalVehInfo.curLane.curvature(2);
mLocalVehInfo.curLane.curvature(1)];
```

```
    vLateralDeviation = dot(mCurVehInfo.Position - mergePoint,
vNormal)*vNormal;
    temp = cross([mLocalVehInfo.curLane.curvature; 0], [vLateralDeviation;
0]);
    mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(vLateralDeviation)*temp(3);
end

%% VehInput.curLane/leftLane/rightLane
if mCurVehInfo.Position(2) >= LaneWidth
    mLocalVehInfo.rightLane.exist = true;
elseif mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.leftLane.exist = true;
elseif mCurVehInfo.Position(2) < 0 && mCurVehInfo.Position(2) > -
LaneWidth ...
        && mCurVehInfo.Position(1) >= mergeRange(1) &&
mCurVehInfo.Position(1) <= mergeRange(2)
    mLocalVehInfo.leftLane.exist = true;
end

for i = 1: length(mGlobalInfo)
    if mGlobalInfo(i). Id <= 0
        % Vehicle is disabled
        continue;
    end

    if mCurVehInfo.Position(2) >= LaneWidth
        % Vehicle is in the first lane
        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
```

```
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.rightLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.rightLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.rightLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.rightLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.rightLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.rightLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    elseif mCurVehInfo.Position(2) >= 0
        % Vehicle in the second lane
        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
```

```
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    else
        % Vehicle in the third lane

        % treat the mergePoint as front objection(vehicle) with 0 velocity

        if norm(mCurVehInfo.Position - mergePoint) <
mLocalVehInfo.curLane.frontDistance
            mLocalVehInfo.curLane.frontDistance = norm(mCurVehInfo.Position
- mergePoint);
            mLocalVehInfo.curLane.frontVelocity = -
norm(mCurVehInfo.Velocity);
        end

        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane
            % Do nothing
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane

            sensoredVehToMerge = mGlobalInfo(i).Position(1) -
mergePoint(1);
            curVehToMerge = -norm(mCurVehInfo.Position - mergePoint);

            if sensoredVehToMerge > curVehToMerge ...
                    && (sensoredVehToMerge - curVehToMerge) <
mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance = sensoredVehToMerge -
curVehToMerge;
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif sensoredVehToMerge < curVehToMerge ...
                    && (curVehToMerge - sensoredVehToMerge) <
mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance = curVehToMerge -
sensoredVehToMerge;
```

```
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        else
            % Sensored vehicle is in the third lane too

            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && norm(mGlobalInfo(i).Position - mCurVehInfo.Position)
< mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
norm(mGlobalInfo(i).Position - mCurVehInfo.Position);
                mLocalVehInfo.curLane.frontVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && norm(mCurVehInfo.Position - mGlobalInfo(i).Position)
< mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
norm(mCurVehInfo.Position - mGlobalInfo(i).Position);
                mLocalVehInfo.curLane.rearVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        end
    end
end
```

Appendix-16: Source code of sensor model for three lane curve and lane switch scenarios

```
%
% Sensor model for three lane curve and lane scenarios

LaneWidth = 7;
radius = 100;

centerPoint1 = [-radius - LaneWidth; 0];
centerPoint2 = [radius + LaneWidth; 0];

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo
if mCurVehInfo.Destination > 0
    if mCurVehInfo.Destination == 1
        % Vehicle's destination is the first lane
        if mCurVehInfo.Position(1) > -LaneWidth/2
            % Vehicle is in the second/third lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Right;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    elseif mCurVehInfo.Destination == 2
        % Vehicle's destination is the second lane
        if mCurVehInfo.Position(1) < -LaneWidth/2
            % Vehicle is in the first lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        elseif mCurVehInfo.Position(1) > LaneWidth/2
            % Vehicle is in the third lane -> change right
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Right;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    else
        % Vehicle's destination is the third lane
        if mCurVehInfo.Position(1) < LaneWidth/2
            % Vehicle is in the first/second lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    end
end

if mCurVehInfo.Position(1) <= -LaneWidth/2
    % Vehicle is in the left lane
    if mCurVehInfo.Position(2) >= 0
```

```
        mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(mCurVehInfo.Position - centerPoint1) - radius;
        tRelPos = mCurVehInfo.Position - centerPoint1;
        tRelPos = tRelPos/norm(tRelPos);
        mLocalVehInfo.curLane.curvature = [tRelPos(2); -tRelPos(1)];
    else
        mLocalVehInfo.dynamicInfo.lateralDeviation =
mCurVehInfo.Position(1) + LaneWidth;
        mLocalVehInfo.curLane.curvature = [0; -1];
    end
elseif mCurVehInfo.Position(1) >= LaneWidth/2
    % Vehicle is in the right lane
    if mCurVehInfo.Position(2) >= 0
        mLocalVehInfo.dynamicInfo.lateralDeviation = radius -
norm(mCurVehInfo.Position - centerPoint2);
        tRelPos = mCurVehInfo.Position - centerPoint2;
        tRelPos = tRelPos/norm(tRelPos);
        mLocalVehInfo.curLane.curvature = [-tRelPos(2); tRelPos(1)];
    else
        mLocalVehInfo.dynamicInfo.lateralDeviation =
mCurVehInfo.Position(1) - LaneWidth;
        mLocalVehInfo.curLane.curvature = [0; -1];
    end
else
    % Vehicle is in the center lane
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(1) -
0;
    mLocalVehInfo.curLane.curvature = [0; -1];
end

%% VehInput.curLane/leftLane/rightLane

if mCurVehInfo.Position(1) <= -LaneWidth/2
    if mCurVehInfo.Position(2) < 0
        mLocalVehInfo.leftLane.exist = true;
        mLocalVehInfo.leftLane.curvature = [0; -1];
    end
elseif mCurVehInfo.Position(1) >= LaneWidth/2
    if mCurVehInfo.Position(2) < 0
        mLocalVehInfo.rightLane.exist = true;
        mLocalVehInfo.rightLane.curvature = [0; -1];
    end
else
    if mCurVehInfo.Position(2) < 0
        mLocalVehInfo.leftLane.exist = true;
        mLocalVehInfo.rightLane.exist = true;
        mLocalVehInfo.leftLane.curvature = [0; -1];
        mLocalVehInfo.rightLane.curvature = [0; -1];
    end
end

for i = 1 : length(mGlobalInfo)
    if mGlobalInfo(i).Id <= 0
```

```
        % Vehicle is disabled
        continue;
    end

    if mCurVehInfo.Position(1) <= -LaneWidth/2
        % Vehicle is in the left lane
        if mGlobalInfo(i).Position(1) <= -LaneWidth/2
            % Sensored vehicle is in the left lane
            if mCurVehInfo.Position(2) >= 0 &&
mGlobalInfo(i).Position(2) >= 0
                % Both current vehicle and sensored vehicles are in the
                % circle section
                mCurRadian = atan2(mCurVehInfo.Position(2) -
centerPoint1(2), mCurVehInfo.Position(1) - centerPoint1(1));
                mSensoredRadian = atan2(mGlobalInfo(i).Position(2) -
centerPoint1(2), mGlobalInfo(i).Position(1) - centerPoint1(1));

                if mSensoredRadian < mCurRadian ...
                        && radius*(mCurRadian - mSensoredRadian) <
mLocalVehInfo.curLane.frontDistance
                    % Sensored vehicle is in front of the current vehicle
and with a
                    % smaller distance
                    mLocalVehInfo.curLane.frontDistance =
radius*(mCurRadian - mSensoredRadian);
                    mLocalVehInfo.curLane.frontVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

                elseif mSensoredRadian > mCurRadian ...
                        && radius*(mSensoredRadian - mCurRadian) <
mLocalVehInfo.curLane.rearDistance
                    % Sensored vehicle is in rear of the current vehicle
and with a
                    % smaller distance
                    mLocalVehInfo.curLane.rearDistance =
radius*(mSensoredRadian - mCurRadian);
                    mLocalVehInfo.curLane.rearVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

                end
            elseif mCurVehInfo.Position(2) < 0 &&
mGlobalInfo(i).Position(2) < 0
                % Both current vehicle and sensored vehicles are in the
                % straight section
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.curLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
                    mLocalVehInfo.curLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
```

```
                    mLocalVehInfo.curLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.curLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.curLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.curLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                end

            end
        elseif mGlobalInfo(i).Position(1) >= LaneWidth/2
            % Sensored vehicle is in the right lane

            % do nothing since the two vehicles are seperated by the middle
            % lane

        else
            % Sensored vehicle is in the middle lane
            if mCurVehInfo.Position(2) < 0
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.leftLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
                    mLocalVehInfo.leftLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                    mLocalVehInfo.leftLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.leftLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.leftLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.leftLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                end
            end

        end
    elseif mCurVehInfo.Position(1) >= LaneWidth/2
        % Vehicle is in the right lane

        if mGlobalInfo(i).Position(1) <= -LaneWidth/2
```

```
            % Sensored vehicle is in the left lane

            % do nothing since the two lanes are seperated by the middle
            % lane

        elseif mGlobalInfo(i).Position(1) >= LaneWidth/2
            % Sensored vehicle is in the right lane

            if mCurVehInfo.Position(2) >= 0 &&
mGlobalInfo(i).Position(2) >= 0
                % Both current vehicle and sensored vehicles are in the
                % circle section
                mCurRadian = atan2(mCurVehInfo.Position(2) -
centerPoint2(2), mCurVehInfo.Position(1) - centerPoint2(1));
                mSensoredRadian = atan2(mGlobalInfo(i).Position(2) -
centerPoint2(2), mGlobalInfo(i).Position(1) - centerPoint2(1));

                if mSensoredRadian > mCurRadian ...
                        && radius*(mSensoredRadian - mCurRadian) <
mLocalVehInfo.curLane.frontDistance
                    % Sensored vehicle is in front of the current vehicle
and with a
                    % smaller distance
                    mLocalVehInfo.curLane.frontDistance =
radius*(mSensoredRadian - mCurRadian);
                    mLocalVehInfo.curLane.frontVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

                elseif mSensoredRadian < mCurRadian ...
                        && radius*(mCurRadian - mSensoredRadian) <
mLocalVehInfo.curLane.rearDistance
                    % Sensored vehicle is in rear of the current vehicle
and with a
                    % smaller distance
                    mLocalVehInfo.curLane.rearDistance = radius*(mCurRadian
- mSensoredRadian);
                    mLocalVehInfo.curLane.rearVelocity =
norm(mGlobalInfo(i).Velocity) - norm(mCurVehInfo.Velocity);

                end

            elseif mCurVehInfo.Position(2) < 0 &&
mGlobalInfo(i).Position(2) < 0
                % Both current vehicle and sensored vehicles are in the
                % straight section
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.curLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
```

```
                    mLocalVehInfo.curLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                    mLocalVehInfo.curLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.curLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.curLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.curLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                end

            end

        else
            % Sensored vehicle is in the middle lane
            if mCurVehInfo.Position(2) < 0
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.rightLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
                    mLocalVehInfo.rightLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                    mLocalVehInfo.rightLane.frontVelocity =
mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2);
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.rightLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.rightLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.rightLane.rearVelocity =
mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2);
                end
            end

        end

    else
        % Vehicle is in the middle lane
        if mGlobalInfo(i).Position(1) <= -LaneWidth/2
            % Sensored vehicle is in the left lane
            if mGlobalInfo(i).Position(2) < 0
                % Sensored vehicle is in the straight section
```

```
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.rightLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
                    mLocalVehInfo.rightLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                    mLocalVehInfo.rightLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.rightLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.rightLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.rightLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                end
            end
        elseif mGlobalInfo(i).Position(1) >= LaneWidth/2
            % Sensored vehicle is in the right lane
            if mGlobalInfo(i).Position(2) < 0
                if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                        && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.leftLane.frontDistance
                    % Vehicle in front of current vehicle and with a
smaller
                    % distance
                    mLocalVehInfo.leftLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                    mLocalVehInfo.leftLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                elseif mGlobalInfo(i).Position(2) >
mCurVehInfo.Position(2) ...
                        && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.leftLane.rearDistance
                    % Vehicle in rear of current vehicle and with a smaller
                    % distance
                    mLocalVehInfo.leftLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                    mLocalVehInfo.leftLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
                end
            end
        else
            % Sensored vehicle is in the middle lane
            if mGlobalInfo(i).Position(2) < mCurVehInfo.Position(2) ...
                    && (mCurVehInfo.Position(2) -
mGlobalInfo(i).Position(2)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
```

```
                % distance
                mLocalVehInfo.curLane.frontDistance =
mCurVehInfo.Position(2) - mGlobalInfo(i).Position(2);
                mLocalVehInfo.curLane.frontVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
            elseif mGlobalInfo(i).Position(2) > mCurVehInfo.Position(2) ...
                    && (mGlobalInfo(i).Position(2) -
mCurVehInfo.Position(2)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mGlobalInfo(i).Position(2) - mCurVehInfo.Position(2);
                mLocalVehInfo.curLane.rearVelocity = -
(mGlobalInfo(i).Velocity(2) - mCurVehInfo.Velocity(2));
            end

        end

    end

end
```

Appendix-17: Source code of sensor model for four lane switch and lane merge scenarios

```
%
% Sensor model for four lane switch and lane merge scenarios

LaneWidth = 7;

mergePoint = [173 + LaneWidth/2*cos(pi/6); LaneWidth/2*sin(pi/6)];
mergeRange = 173 + LaneWidth/2/sin(pi/6)*[-1, 1];
mergePoint2 = [70/tan(pi/9) + 30/tan(pi/6) + 0*7/(2*sin(pi/9)); 0*3.5];
mergeRange2 = 70/tan(pi/9) + 30/tan(pi/6) + LaneWidth/2/sin(pi/9)*[-1, 1];

if mCurVehInfo.Id <= 0
    % Vehicle is disabled
    return;
end

curId = abs(mCurVehInfo.Id);

%% VehglobalInfoInput.dynamicInfo
if mCurVehInfo.Destination > 0
    if mCurVehInfo.Destination == 1
        % Vehicle's destination is the first lane
        if mCurVehInfo.Position(2) < LaneWidth
            % Vehicle is in the second/third lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    elseif mCurVehInfo.Destination == 2
        % Vehicle's destination is the second lane
        if mCurVehInfo.Position(2) >= LaneWidth
            % Vehicle is in the first lane -> change right
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Right;
        elseif mCurVehInfo.Position(2) < 0
            % Vehicle is in the third lane -> change left
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    else
        % Vehicle's destination is the third lane
        if mCurVehInfo.Position(2) < 0
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.Left;
        else
            mLocalVehInfo.dynamicInfo.changeLane = ChangeLane.None;
        end
    end
end

if mCurVehInfo.Position(2) >= LaneWidth
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
3*LaneWidth/2;
```

```
elseif mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.dynamicInfo.lateralDeviation = mCurVehInfo.Position(2) -
LaneWidth/2;
elseif mCurVehInfo.Destination == 2
    % Vehicle is in the first side lane
    % assume side lane is 30 degree with main lane
    mLocalVehInfo.curLane.curvature = [cos(pi/6); sin(pi/6)];

    vNormal = [-mLocalVehInfo.curLane.curvature(2);
mLocalVehInfo.curLane.curvature(1)];
    vLateralDeviation = dot(mCurVehInfo.Position - mergePoint,
vNormal)*vNormal;
    temp = cross([mLocalVehInfo.curLane.curvature; 0], [vLateralDeviation;
0]);
    mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(vLateralDeviation)*temp(3);
else
    % Vehicle is in the second side lane

    if mCurVehInfo.Position(2) < -70
        mLocalVehInfo.curLane.curvature = [cos(pi/6); sin(pi/6)];
        vNormal = [-mLocalVehInfo.curLane.curvature(2);
mLocalVehInfo.curLane.curvature(1)];
        vLateralDeviation = dot(mCurVehInfo.Position - mergePoint,
vNormal)*vNormal;
        temp = cross([mLocalVehInfo.curLane.curvature; 0],
[vLateralDeviation; 0]);
        mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(vLateralDeviation)*temp(3);
    else
        mLocalVehInfo.curLane.curvature = [cos(pi/9); sin(pi/9)];
        vNormal = [-mLocalVehInfo.curLane.curvature(2);
mLocalVehInfo.curLane.curvature(1)];
        vLateralDeviation = dot(mCurVehInfo.Position - mergePoint2,
vNormal)*vNormal;
        temp = cross([mLocalVehInfo.curLane.curvature; 0],
[vLateralDeviation; 0]);
        mLocalVehInfo.dynamicInfo.lateralDeviation =
norm(vLateralDeviation)*temp(3);
    end

end

%% VehInput.curLane/leftLane/rightLane
if mCurVehInfo.Position(2) >= LaneWidth
    mLocalVehInfo.rightLane.exist = true;
elseif mCurVehInfo.Position(2) >= 0
    mLocalVehInfo.leftLane.exist = true;
elseif mCurVehInfo.Position(2) < 0 && mCurVehInfo.Position(2) > -
LaneWidth ...
        && mCurVehInfo.Position(1) >= mergeRange(1) &&
mCurVehInfo.Position(1) <= mergeRange(2)
```

```
    mLocalVehInfo.leftLane.exist = true;
elseif mCurVehInfo.Position(2) < 0 && mCurVehInfo.Position(2) > -
LaneWidth ...
        && mCurVehInfo.Position(1) >= mergeRange2(1) &&
mCurVehInfo.Position(1) <= mergeRange2(2)
    mLocalVehInfo.leftLane.exist = true;
end

for i = 1 : length(mGlobalInfo)
    if mGlobalInfo(i).Id <= 0
        % Vehicle is disabled
        continue;
    end

    if mCurVehInfo.Position(2) >= LaneWidth
        % Vehicle is in the first lane
        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.rightLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.rightLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.rightLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.rightLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
```

```
                mLocalVehInfo.rightLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.rightLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    elseif mCurVehInfo.Position(2) >= 0
        % Vehicle in the second lane
        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane too
            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && (mGlobalInfo(i).Position(1) -
mCurVehInfo.Position(1)) < mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
mGlobalInfo(i).Position(1) - mCurVehInfo.Position(1);
                mLocalVehInfo.curLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && (mCurVehInfo.Position(1) -
mGlobalInfo(i).Position(1)) < mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
mCurVehInfo.Position(1) - mGlobalInfo(i).Position(1);
                mLocalVehInfo.curLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - mCurVehInfo.Velocity(1);
            end
        end
    elseif mCurVehInfo.Destination == 2
        % Vehicle in the third lane (the first side lane)
```

```
        % treat the mergePoint as front objection(vehicle) with 0 velocity

        if norm(mCurVehInfo.Position - mergePoint) <
mLocalVehInfo.curLane.frontDistance
            mLocalVehInfo.curLane.frontDistance = norm(mCurVehInfo.Position
- mergePoint);
            mLocalVehInfo.curLane.frontVelocity = -
norm(mCurVehInfo.Velocity);
        end

        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane
            % Do nothing
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane

            sensoredVehToMerge = mGlobalInfo(i).Position(1) -
mergePoint(1);
            curVehToMerge = -norm(mCurVehInfo.Position - mergePoint);

            if sensoredVehToMerge > curVehToMerge ...
                    && (sensoredVehToMerge - curVehToMerge) <
mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance = sensoredVehToMerge -
curVehToMerge;
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif sensoredVehToMerge < curVehToMerge ...
                    && (curVehToMerge - sensoredVehToMerge) <
mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.rearDistance = curVehToMerge -
sensoredVehToMerge;
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        else
            % Sensored vehicle is in the third lane too

            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && norm(mGlobalInfo(i).Position - mCurVehInfo.Position)
< mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
norm(mGlobalInfo(i).Position - mCurVehInfo.Position);
```

```
                mLocalVehInfo.curLane.frontVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && norm(mCurVehInfo.Position - mGlobalInfo(i).Position)
< mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
norm(mCurVehInfo.Position - mGlobalInfo(i).Position);
                mLocalVehInfo.curLane.rearVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        end
    else
        % Vehicle in the second side lane

        % treat the mergePoint2 as front objection(vehicle) with 0 velocity

        if norm(mCurVehInfo.Position - mergePoint2) <
mLocalVehInfo.curLane.frontDistance
            mLocalVehInfo.curLane.frontDistance = norm(mCurVehInfo.Position
- mergePoint2);
            mLocalVehInfo.curLane.frontVelocity = -
norm(mCurVehInfo.Velocity);
        end

        if mGlobalInfo(i).Position(2) >= LaneWidth
            % Sensored vehicle is in the first lane
            % Do nothing
        elseif mGlobalInfo(i).Position(2) >= 0
            % Sensored vehicle is in the second lane

            sensoredVehToMerge = mGlobalInfo(i).Position(1) -
mergePoint2(1);
            curVehToMerge = -norm(mCurVehInfo.Position - mergePoint2);

            if sensoredVehToMerge > curVehToMerge ...
                    && (sensoredVehToMerge - curVehToMerge) <
mLocalVehInfo.leftLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.leftLane.frontDistance = sensoredVehToMerge -
curVehToMerge;
                mLocalVehInfo.leftLane.frontVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif sensoredVehToMerge < curVehToMerge ...
                    && (curVehToMerge - sensoredVehToMerge) <
mLocalVehInfo.leftLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
```

```
                % distance
                mLocalVehInfo.leftLane.rearDistance = curVehToMerge -
sensoredVehToMerge;
                mLocalVehInfo.leftLane.rearVelocity =
mGlobalInfo(i).Velocity(1) - dot(mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        else
            % Sensored vehicle is in the side lane too

            if mGlobalInfo(i).Position(2) > -70 && ...
                    mGlobalInfo(i).Position(1) <
(mGlobalInfo(i).Position(2) + 100 + 7/sqrt(3))*sqrt(3)
                % Sensored vehicle is in the first side lane and passes the
                % switching point, ignore

                continue;
            end


            if mGlobalInfo(i).Position(1) > mCurVehInfo.Position(1) ...
                    && norm(mGlobalInfo(i).Position - mCurVehInfo.Position)
< mLocalVehInfo.curLane.frontDistance
                % Vehicle in front of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.frontDistance =
norm(mGlobalInfo(i).Position - mCurVehInfo.Position);
                mLocalVehInfo.curLane.frontVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            elseif mGlobalInfo(i).Position(1) < mCurVehInfo.Position(1) ...
                    && norm(mCurVehInfo.Position - mGlobalInfo(i).Position)
< mLocalVehInfo.curLane.rearDistance
                % Vehicle in rear of current vehicle and with a smaller
                % distance
                mLocalVehInfo.curLane.rearDistance =
norm(mCurVehInfo.Position - mGlobalInfo(i).Position);
                mLocalVehInfo.curLane.rearVelocity =
dot(mGlobalInfo(i).Velocity - mCurVehInfo.Velocity,
mLocalVehInfo.curLane.curvature);
            end
        end
    end
end
```

Appendix -18: Vehicle Model

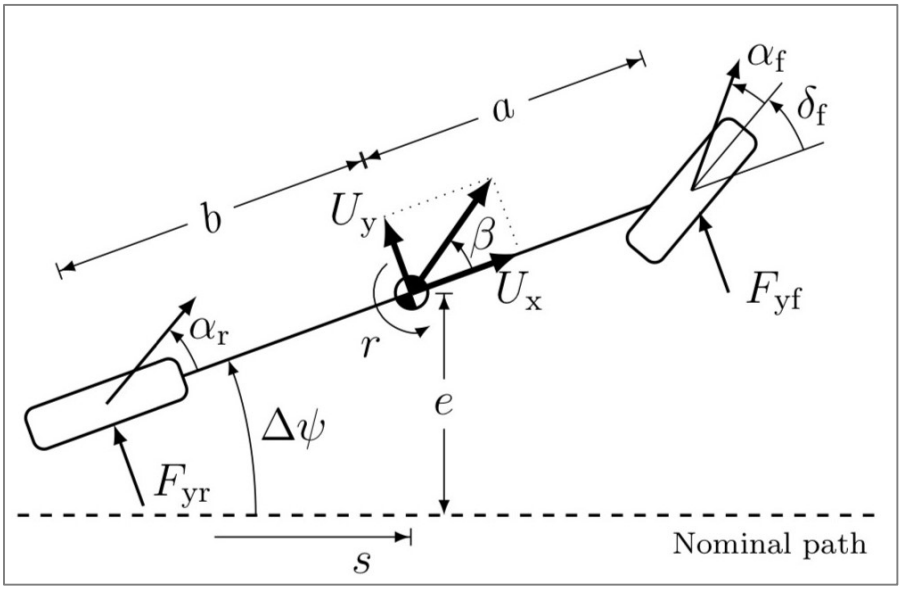

Figure 18.1: The bicycle model's schematic illustration

Velocity States

The velocity states in the vehicle model are the sideslip angle β and yaw rate r. The angle of the sideslip is given by:

$$\beta = tan(U_y/U_x) \approx U_y/U_x \quad \text{..... Equation (78.1)}$$

In the body-fixed frame, the lateral and longitudinal velocities are indicated as U_y and U_x. It was assumed for simplicity $U_x > U_y$ as a convenience and that U_x is constant. For the sideslip and yaw rates, the equations of motion are as follows.

$$\dot{\beta} = \frac{F_{yf} + F_{yr}}{mU_x} - r \quad \text{..... Equation (18.2)}$$

$$\dot{r} = \frac{aF_{yf} - bF_{yr}}{I_{zz}} \quad \text{..... Equation (18.3)}$$

The lateral tyre force on the front and rear axles is indicated by F_{yf} and F_{yr}, the vehicle mass is denoted by m, the yaw inertia is denoted by I_{zz}, and the distances between the vehicle's

centre of gravity and the front and rear axles are marked by a and b, respectively. The slip angle of the front tyre, α_f, and the rear tyre, α_r, may be stated as follows:

$$\alpha_f = tan^{-1}\left(\beta + \frac{ar}{U_x}\right) - \delta \approx \beta + \frac{ar}{U_x} - \delta \quad \text{..... Equation (18.4)}$$

$$\alpha_r = tan^{-1}\left(\beta - \frac{br}{U_x}\right) \approx \beta - \frac{br}{U_x} \quad \text{.....Equation (18.5)}$$

Small-angle approximations result in linear expressions. Fialas' brush tyre model, as described by Pacejka, provides the link between lateral tyre forces and tyre slip angles as follows:

$$F_y = \begin{cases} -C_\alpha \tan\alpha + \frac{C_\alpha^2}{3\mu F_z}|\tan\alpha|\tan\alpha \\ \quad -\frac{C_\alpha^3}{27\ {}^2F_z^2}\tan^3\alpha \qquad\qquad |\alpha| < tan^{-1}(\frac{3\mu\ z}{C_\alpha}) \\ \quad -\mu F_z \sin\alpha \end{cases}$$

Otherwise

$$F_y = f_{tire}(\alpha) \quad \text{..... Equation (18.6)}$$

The surface coefficient of friction is provided here as μ, the normal load as F_{zf} and F_{zr}, and the tyre cornering stiffness as C_α.

The MPC controller's vehicle model uses the front tyre force to maintain the problem linear in terms of input. Equations (18.4) and (18.6) provide the following steering angle:

$$\delta = \beta + \frac{ar}{U\ x} - f_{tire}^{-1}(F_{yf}) \quad \text{..... Equation (18.7)}$$

To handle the nonlinearity of the rear tyres, the brush tyre model is linearized at a given rear tyre slip angle ($\bar{\propto}_r$), and the rear tyre force (F_{yr}) is therefore treated as an affine function of $\propto_r$.

$$F_{yr} = \bar{F}_{yr} - \bar{C}_{\bar{\propto}_r}(\alpha_r - \bar{\alpha}_r) \quad \text{..... Equation (18.8)}$$

The comparable cornering stiffness at $\bar{\alpha}_r$ is $\bar{C}_{\bar{\alpha}_r}$, and $\bar{F}_{yr} = f_{tire}(\bar{\alpha}_r)$. The current rear slip angle, α_r, is selected to be $\bar{\alpha}_r$ in the prediction horizon's first-time steps. This enables the MPC controller to take rear tyre saturation into account in the short-term prediction. This enables the MPC controller to explicitly account for rear tyre saturation in short-term prediction.

The velocity state equations of motion may now be expressed as affine functions of the states and inputs, F_{yr}:

$$\dot{\beta} = \frac{F_{yf} + \bar{F}_{yr} - \bar{C}_{\bar{\alpha}_r}(\beta - \frac{br}{U_x} - \bar{\alpha}_r)}{mU_x} - r \quad \text{Equation (18.9)}$$

$$\dot{r} = \frac{aF_{yf} - b[\bar{F}_{yr} - \bar{C}_{\bar{\alpha}_r}(\beta - \frac{br}{U_x} - \bar{\alpha}_r)]}{I_{zz}} \quad \text{Equation (18.10)}$$

Position States

The vehicle's position states, the heading deviation ($\Delta\psi$) and lateral deviation (e), are expressed in terms of a nominal route that does not have to be obstacle-free. The heading deviation and lateral deviation have the following equations of motion:

$$\dot{\Delta}\psi = r \quad \text{Equation (18.11)}$$

$$\dot{e} = U_x \sin(\Delta\psi) + U_y \cos(\Delta\psi) \quad \text{Equation (18.12)}$$

Small angle assumptions for (($\Delta\psi$) and (β) are used to approximate the following nonlinear equations as linear functions of the vehicle states, yielding:

$$\dot{e} \approx U_x\Delta\psi + U_x\beta \quad \text{Equation (18.13)}$$

As a result of combining equations (18.9), (18.10), (18.11), and (18.13), a continuous state-space representation of the vehicle model is obtained as:

$$\dot{x} \approx A_c(\bar{\alpha}_r) + B_c F_{yf} + d_c(\bar{\alpha}_r) \quad \text{Equation (18.14)}$$

and,

$$x = [\beta \quad r \quad \Delta\psi \quad e]^T$$

$$A_c(\bar{\alpha}_r) = \begin{bmatrix} -\frac{\bar{C}_{\bar{\alpha}_r}}{mU_x} & \frac{b\bar{C}_{\bar{\alpha}_r}}{mU_x^2} - 1 & 0 & 0 \\ \frac{b\bar{C}_{\bar{\alpha}_r}}{I_{zz}} & \frac{b^2\bar{C}_{\bar{\alpha}_r}}{I_{zz}U_x} & 0 & 0 \\ 0 & 1 & 0 & 0 \\ U_x & 0 & U_x & 0 \end{bmatrix}$$

$$B_c = \begin{bmatrix} \frac{1}{mU_x} & \frac{a}{I_{zz}} & 0 & 0 \end{bmatrix}^T$$

$$d_c(\bar{\alpha}_r) = \begin{bmatrix} \frac{\bar{F}_{yf} - \bar{\alpha}_r\bar{C}_{\bar{\alpha}_r}}{mU_x} & -\frac{b(\bar{F}_{yf} - \bar{\alpha}_r\bar{C}_{\bar{\alpha}_r})}{I_{zz}} & 0 & 0 \end{bmatrix}^T$$

A continuous-time model is denoted by the letter c. $A_c(\bar{\alpha}_r)$ denotes A_c's linearization around $\bar{\alpha}_r$.

Appendix -19: Alternative MPC Formulation for Lane Dividers

An affine vehicle model is used to determine the vehicle input for lateral tyre force using a model predictive control (MPC) technique. A 4-state bicycle dynamic model with a constant speed assumption is employed in the MPC formulation. The vehicle sideslip angle, yaw rate r, heading deviation, and lateral deviation e make up the state vector:

$$x = [U_y \;\; r \;\; \Delta\psi \;\; e]^T$$

The lateral tyre force, which is nonlinearly related to the steering angle, is the vehicle model u's control input. When the vehicle input is represented as a lateral tyre force, however, the vehicle states have a linear relationship:

$$x^{(k+1)} = A^{(k)}x^{(k)} + B^{(k)}u^{(k)} + c^{(k)} \quad k = 1 \ldots . T \quad \ldots\ldots\ldots \text{Equation (19.1)}$$

At each time step in the forecast horizon k, up to a fixed period T.

A nonzero diagonal element in the weighting matrix Q is associated with lateral deviation and heading deviation as these states are defined relative to a nominal route. The following limitation represents the safe driving space in which a vehicle may travel in the environment:

$$H_{env}^{k}x^{(k)} = G_{env}^{(k)} \quad k = 1 \ldots . . T \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (19.8)}$$

Vehicle states must operate in an obstacle-free environment. The limits of a roadway's traffic lanes are encoded using:

$$H_{tra}^{k}x^{(k)} = G_{tra}^{(k)} \quad k = 1 \ldots . . T \quad \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{Equation (19.3)}$$

where the subscript "tra" denotes traffic-related regulation. The following summarises the optimization problem:

$$\textit{Minimize (u):}\ \sum_{k=1}^{T} x^{(k)T} Q^{(k)} x^{(k)} + v^{(k)T} R^{(k)} v^{(k)} + W_{env}^{k}\sigma_{env}^{k} + W_{tra}^{k}(\sigma_{tra}^{k})^{2} \quad \textit{Equation (19.4a)}$$

$$\textit{Subject to:}\ x^{(k+1)} = A^{(k)} x^{(k)} + B^{(k)} u^{(k)} + C^{(k)} \quad \textit{Equation (19.4b)}$$

$$H_{env}^{k} x^{(k)} = G_{env}^{(k)} \quad \textit{Equation (19.4c)}$$

$$H_{tra}^{k} x^{(k)} = G_{tra}^{(k)} \quad \textit{Equation (19.4d)}$$

$$\left|u^{(k)}\right| \leq u_{max}^{(k)} \quad \textit{Equation (19.4e)}$$

$$\left|v^{(k)}\right| \leq v_{max}^{(k)} \quad \textit{Equation (19.4f)}$$

where $v^{(k)} = u^{(k)} - u^{(k-1)}$ *represents the change in lateral tyre force,* σ_{env} *and* σ_{tra} *are slack variables on the restrictions enforcing the safe environment and traffic lanes, respectively, and* $v_{max}^{(k)}$ *and* $u_{max}^{(k)}$ *are physical limitations in the steering system and tyre forces.*

The weight given to changes in steering inputs is determined by the cost (R). Cost W_{env} decides how much emphasis should be paid on staying inside the safe environmental envelope, i.e., avoiding collisions with objects. Cost of the weight given to following traffic regulations, such as not crossing a double yellow line, is determined by W_{tra}. The higher the priority of environmental envelope and/or traffic law violation in the list of restrictions, the more weight is given to cost terms with a slack variable. The environmental slack variable σ_{env} is linear, but the traffic law slack variable σ_{tra} is quadratic in the cost function to avoid penalising relatively minor lane boundary violations.

Appendix – 20: Formulation of the POMDP Speed Scale

The vehicle's acceleration values are directly controlled by the Markov decision process (POMDP) formulation presented in Chapter 4. Since the acceleration orders vary with each time step, the steering MPC is unable to forecast how the speed will change along the prediction horizon during the prediction horizon. As a result, Chapter 4 assumes that the speed remains constant over the prediction horizon. A speed profile, such as that proposed by Funke et al., might be used as an alternate method of speed prediction for steering MPC. Instead of commanding acceleration directly, an alternative POMDP formulation would scale the desired speed profile rather than commanding acceleration directly to reconcile POMDP speed control with MPC speed profile predictions. This appendix illustrates the speed scaling POMDP formulation by using the scenario of a pedestrian crosswalk on a two-lane roadway with a huge vehicle occluding the event of a pedestrian crossing, as illustrated in figure 4.1.

The state space is represented as a low-dimensional subspace that contains information about the vehicle's behaviour and velocity, as well as information about its perception. The following components of the state are taken into consideration in this work: the vehicle's velocity (v_t), the vehicle's distance along the path (d_t), and the occurrence of a pedestrian crossing (c_t). Continuous states exist for speed and distance along the course. States vt and dt are discretized to minimise the size of the problem even further. The maximum speed taken into consideration for the scenario is $10m/s$, with discretization intervals of $1m/s$ between each frame. For a path that is $60m$ long, the distance along the path is discretized into intervals of $0.5m$. State (c_t) has previously been discretized as a binary occurrence in the previous state.

The longitudinal acceleration of the vehicle is the type of actuation considered in this case. Proportional speed control is used to determine the commanded longitudinal acceleration of the vehicle. Consequently, the POMDP action space may be thought of as a speed scaling factor that can be applied to the desired speed in longitudinal control. Following the discretization of the action space, the actions are denoted by $A = 0, 10\%, 20\%, 30\%, 50\%, 60\%, 70\%, 80\%, 90\%, 100\%$, respectively.

The observation space stores data that the agent observes because of performing an action. The number of unobservable tiles (n_t) and the detection of a pedestrian crossing are the two sorts of observations that are taken into consideration (c_t). Since the number of unobservable tiles has been reduced to ten discretized bins that are linearly spaced between 0 and 1800 unobservable tiles, the problem size has been simplified. The identification of a pedestrian

crossing is done by a separate algorithm that has been particularly created to identify pedestrians on the road.

Specifically, the reward function in this POMDP formulation is meant to achieve the following goals:

- Encourage the vehicle to continue driving until it reaches the end of the route.
- If the vehicle detects a pedestrian, it must surrender to the pedestrian. Thus, when a pedestrian crossing event occurs, non-zero scale factors are penalised.
- Furthermore, when the vehicle cannot see the pedestrian, it should not travel at a high rate of speed.

To accomplish the described objectives, the reward function is provided with the following two costs:

- *Complete path reward:* The vehicle that drives all the way to the end of the path will receive a +100 bonus.
- *Not yielding cost:* The cost of not yielding is -50 if the vehicle fails to yield to a pedestrian crossing.
- *Too fast cost:* The cost of deterring the car from rushing around the occlusion because to its proximity to a pedestrian crossing is set at -5, which is orders of magnitude lower than the cost of a collision. The penalty for exceeding 6 m/s when a pedestrian crossing is not visible is simply applied in this case by penalising the vehicle for exceeding the speed limit.

For all other states, it is presumed that the reward is equal to zero. The system's dynamics are not genuinely stochastic; rather, uncertainty is injected into the system's dynamics because of the rough discretization of the state space. A pedestrian crossing is also represented as a random process in the case of a collision. The state transition model is defined by the following parameters:

- Because the vehicle simulation is closer to continuous time than state space discretization, speed scaling and changes in speed are not instantly realised in the state space discretization.
- It is presumed that the vehicle is either halted or going ahead (does not reverse direction).
- It is presumed that if a pedestrian is present within the crosswalk, that person is standing immobile.

Although the state in a POMDP is a belief state, the state transition function for a POMDP is the identical as the state transition function for an MDP (assuming no state uncertainty). While the state contains the facts about a pedestrian crossing event, the problem merely keeps a belief about whether a pedestrian crossing event occurred based on observations. The observation model in a typical POMDP problem is described as the conditional probability of witnessing each observation given the current state (s) and the action (a) done to get there: $Pr(o|s,a)$. It is assumed for the purposes of this research that the activity does not add to the observation (o) made. As a result, the observation model is no longer dependent on a and merely needs to define $Pr(o|s)$. $Pr(on, oc|s)$, which is the probability of having a few unobservable tiles and detecting a pedestrian given the present state, is the observation model using the observation space mentioned above. If the situation of a pedestrian not crossing is noticed, a basic observation model with uniform distribution is implemented. Given a state in which a pedestrian is crossing, the observation distribution shifts 30% in favour of identifying a pedestrian crossing.