REINFORCEMENT LEARNING BASED DECISION MAKING FOR SELF DRIVING & SHARED CONTROL BETWEEN HUMAN DRIVER AND MACHINE

A Dissertation

by

SANGJIN KO

Submitted to the Graduate and Professional School of
Texas A&M University
in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Chair of Committee,	Reza Langari
Committee Members,	Prabhakar R. Pagilla
	Srikanth Saripalli
	Shankar P. Bhattacharyya
Head of Department,	Guillermo Aguilar

August 2021

Major Subject: Mechanical Engineering

ABSTRACT

This study presents solutions to decision making for autonomous driving based on reinforcement learning and shared control between human driver and machine. The main objective of this research is to propose decision making models in highway driving and to study how to share control authority between two controllers which are human driver and machine in the vehicle control loop. Therefore, the research consists of two sub-topics 1) shared control between human driver and machine, and 2) reinforcement learning based decision making model.

First of all, for shared control between human driver and machine, game theoretical model predictive control (MPC) approach is studied. Four game frameworks - non-cooperative game with the simultaneous move, non-cooperative game with leader and follower, non-cooperative game with sequential move, and cooperative game - are investigated, and then several driving situations are studied under different game frameworks. Shared control strategy for fully mixed driving authority is proposed considering collision probability based on Time-to-Collision (TTC) and the weighted square sum of tracking error. Also, game transition between different game frameworks is studied in consideration of driving situations. Simulations for cooperative driving and inter-game transition driving were conducted and the simulation results show that the control authority is shared continuously by the proposed shared control strategy based on collision probability, and realistic driving with game transition is studied and analyzed from the simulation results.

For decision making models in highway driving, we developed Deep Reinforcement Learning (DRL) based decision making models. The problem is formulated as a Reinforcement Learning (RL) problem with three different types of state definitions, and several Deep Q Network (DQN) based RL approaches are applied to design decision makers for highway driving. The three different types of state definitions are 1) relative maneuvers based state with respect to surrounding vehicles, 2) surrounding inter-vehicles gap based state, 3) occupied grid based state (image-like state definition with three channels). For the highway driving decision making problem, designed DQN based algorithms show similar performances with three different types of state definitions.

To verify the performance of RL based decision model, its performance is compared with the performances of other decision models which are the conventional human driver behavior model, rule based decision model, and data-driven decision model. The mean velocities and moving distances from highway driving simulations are compared and analyzed to compare their performances, and RL based decision model shows the best performance among the decision models.

Finally, RL based decision model is applied as machine's decision model with game theoretic MPC based shared controller in several driving situations to evaluate the performance of designed RL based decision model and shared controller under a hierarchical architecture, and it is found that the proposed RL based decision model and shared controller are capable of dealing with undesired driving situations in order to prevent a collision and to guarantee vehicle safety.

ACKNOWLEDGMENTS

First of all, I would like to express my sincere gratitude to my advisor Professor Langari for the continuous support of my Ph.D. study and research for his patience, motivation, enthusiasm, and immense knowledge. His guidance helped me in all the time of research and my research would not have been possible without his guidance.

I would also like to thank the rest of my committee: Professor Pagilla, Professor Saripalli, and Professor Bhattacharyya, for their insightful comments and encouragement. Special thank goes to Mando Corporation for financial support throughout my graduate studies in the United States.

Most importantly, I would like to acknowledge with gratitude, the support, and love of my family: my parents, sisters, brothers, and my lovely wife and daughter.

Last but not least, I sincerely appreciate the almighty God for His graces, strength, faithfulness, guidance, and love in my life.

CONTRIBUTORS AND FUNDING SOURCES

Contributors

This work was supported by a dissertation committee consisting of Professor Reza Langari, Prabhakar R. Pagilla and Srikanth Saripalli of the Department of Mechanical Engineering and Professor Shankar P. Bhattacharyya of the Department of Electrical and Computer Engineering.

All work for the dissertation was completed by the student, under the advisement of Professor Reza Langari of the Department of Mechanical Engineering.

Funding Sources

Graduate study was supported by a fellowship from Mando Corporation.

NOMENCLATURE

DQN	Deep Q Network
RL	Reinforcement Learning
LSTM	Long Short Term Memory
RNN	Recurrent Neural Network
MPC	Model Predictive Control
TTC	Time-To-Collision
ADAS	Advanced Driver Assistance Systems
LDWS	Lane Departure Warning Systems
LKAS	Lane Keeping Assist Systems
FSM	Finite State Machine
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
MOMDP	Mixed Observable Markov Decision Process
MLE	Maximum Likelihood Estimator
MSM	Method of Simulated Moments
RHC	Receding Horizon Control
IDM	Intelligent Driver Model
MOBIL	Minimizing Overall Braking decelerations Induced by Lane changes
DDPG	Deep Deterministic Policy Gradient
DDAC	Deep Deterministic Actor-Critic
PDC	Parallel Distributed Compensation

DP	Dynamic Programming
CNN	Convolutional Neural Networks
AC	Actor-Critic
GRU	Gated Recurrent Unit
CPI	Collision Probability Index

TABLE OF CONTENTS

Page

LIST OF FIGURES

FIGURE Page

LIST OF TABLES

TABLE Page

1. INTRODUCTION AND LITERATURE REVIEW

1.1 Background and Objectives

Improvement of road safety has been a long-term and significant issue while many advanced driver assistance systems (ADAS), such as lane departure warning system (LDWS) and lane keep assist system (LKAS), have been developed and applied as semi-autonomous driving technologies to address safety as well as to relieve human drivers from the driving task. Recently, fully autonomous driving has received a great deal of public attention with many researchers actively working on autonomous driving. A number of companies have invested in the development of autonomous driving technologies. For instance, Google is investing over one billion dollars in the research on autonomous driving [1]. GM and Ford have also made significant investments in autonomous driving startup companies [2][3] while other companies, not only traditional automakers but also IT companies, are pouring money into developing autonomous driving technologies. Full autonomous driving, once realized, can remove all driving tasks from drivers and contribute to the reduction of fatal car accidents due to human drivers' mistakes which lead to nearly 3,700 death on the world's roads every day [4].

Autonomous driving consists of several modules that are hierarchically arranged such as sensing, perception, mapping & localization, decision making, path planning, and vehicle control as shown in Figure 1.1. Sensing processes low-level information while perception, mapping, and localization produce high-level driving information including position, speed, and acceleration. The decision making module utilizes this information and produces behavioral level decisions. The purpose of decision making is to make effective and safe decisions according to the driving situations. The path planner module generates the desired path to execute the behavioral level decision produced by the decision making module and finally the vehicle control module controls the vehicle to follow the planned path. Behavioral level decision making is very significant in autonomous driving and constitutes the primary focus of this research.

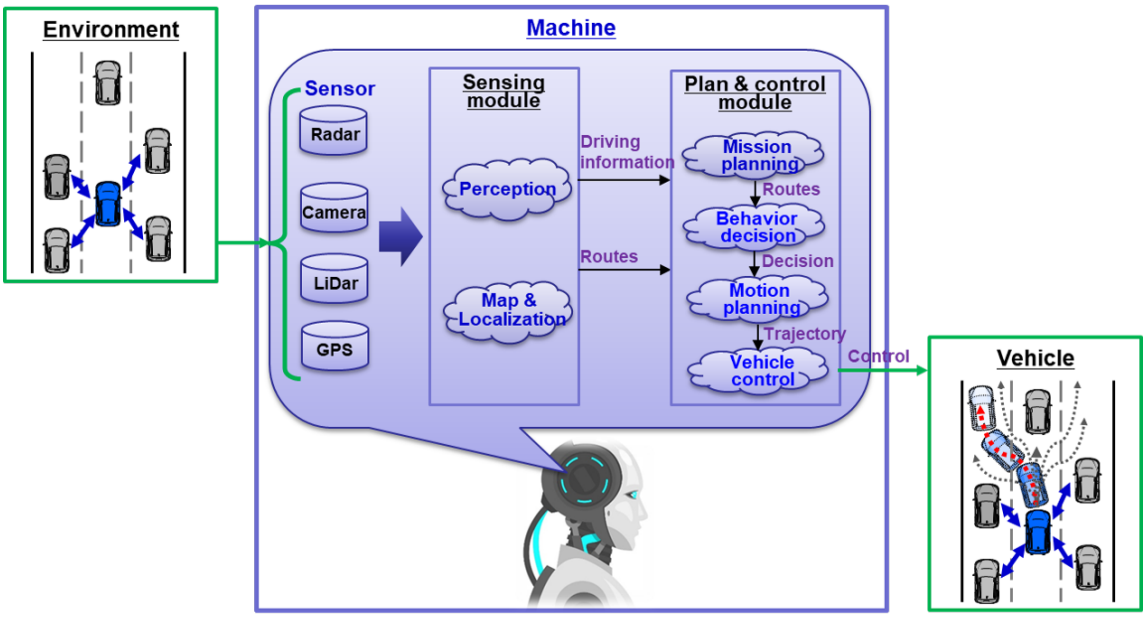


Figure 1.1: Hierarchical Autonomous Driving Tasks

Figure 1.2 shows the SAE automation levels for self-driving [5]. Human driver has the major responsibility for driving at level 3 (conditional automation) or less. But the machine (i.e. vehicle automation) has the responsibility for driving at level 4 (high automation) and level 5 (full automation), where a machine can perform all driving functions under certain conditions or all conditions. Self-driving cars have been already successfully developed and tested in some areas, but they still suffer from several limitations. The automation level for self-driving is generally between levels 3 and 4. In this context, the development of formal shared driving between vehicle control system (machine) and human driver has advantages. That is, additional information, the better quality of information, enable direct negotiations about maneuvers, and correction of driver's behavior in dangerous situations. Also, human drivers will want to drive a vehicle by their preference, even though autonomous driving may be fully available. It means that shared control between human and machine has to be considered not only at level 3 but also at levels 4 and possibly at level 5. In this research, we study shared control between human and machine to maintain control authority over the vehicle.

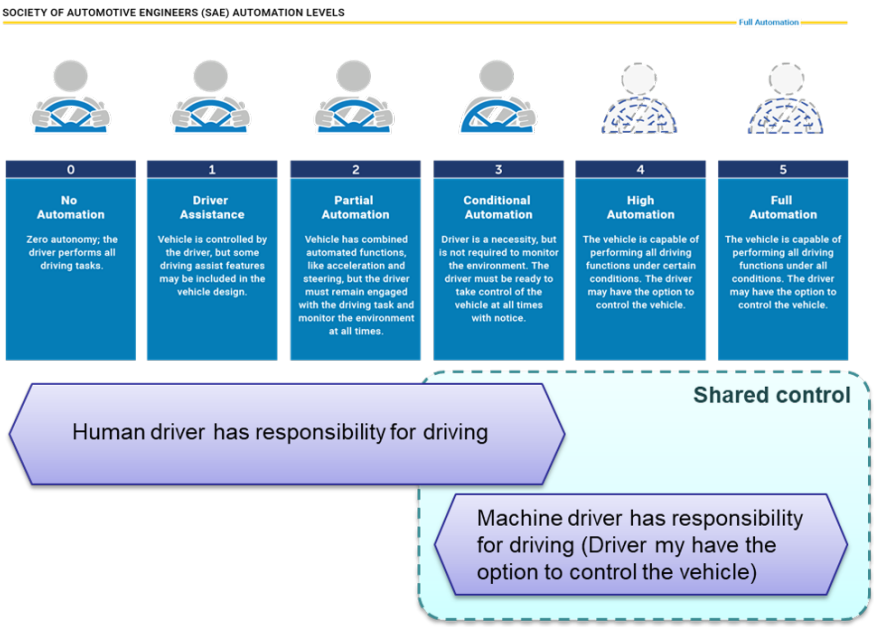


Figure 1.2: SAE automation level [1]

1.2 Literature Review

1.2.1 Decision Making for Self Driving

The decision making module for self-driving acts as a real driver in the general driving situations to make behavioral level decisions to control the vehicle. Various information including the vehicle, namely its position, speed, acceleration and information concerning the surrounding vehicles and objects is fed into the decision making module. The purpose of the decision making module is to produce a high-level decision given all the available data. Several previous directions for decision making for self-driving is summarized below.

1.2.1.1 Rule-based approach

In the early research on decision making for autonomous driving, a rule-based approach was widely used. In this approach, symbolic states of the driving situations and the corresponding decision were manually modeled. In the DARPA Urban Challenge in 2007, this was the most

[1] Modified from "Automated vehicles for safety." Available at https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

common approach and many of the participant teams used this approach in conjunction with a finite state machine (FSM) to determine the decision of the autonomous vehicle. Figure 1.3 shows an example of a FSM based behavioral decision model [6].

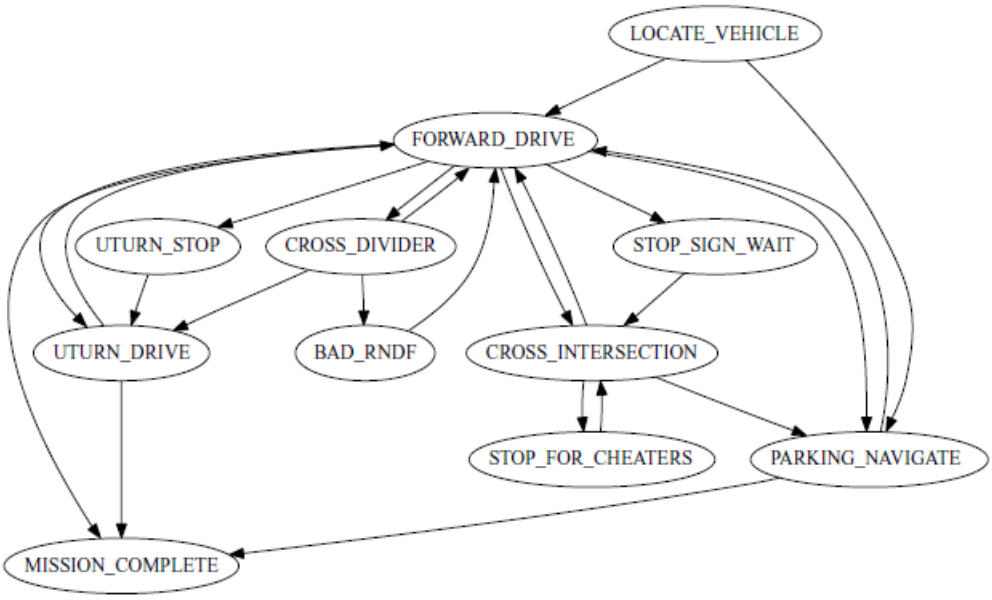


Figure 1.3: Example of Finite State Machine (FSM) based decision model [2]

The Carnegie Mellon University (CMU) team developed an autonomous vehicle, *Boss*, with a pre-encoded rules-based behavioral decision module [7]. At high level, three different contexts are defined as Road, Intersection and Zone, and the corresponding behaviors are defined as Lane Driving, Intersection Handling and Achieving a Zone Pose, respectively. Each high-level behavior has its corresponding sub-behaviors. The high-level behavior, Achieving a Zone Pose, is a behavior for unstructured or unconstrained environments such as parking lots and traffic jams in intersection. The high-level behavior, Lane Driving, consists of 5 different sub-behaviors (Vehicle Driver, Distance Keeper, Merge Planner, Current Scene Reporter, Lane Selector). The high-level behavior,

[2]Reprinted with permission from "Junior: The stanford entry in the urban challeng" by M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, et al., 2008. *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, Copyright 2008 by John Wiley and Sons

Intersection Handling, have three different sub-behaviors (Pan Head Planner, Precedence Estimator, Transition Manager). Additionally, auxiliary goal selection behavior is defined to deal with error recovery, and has two corresponding sub-behaviors, i.e. State Estimator and Goal Selector.

Similarly, the Stanford team developed an autonomous vehicle, *Junior*, with an Finite State Machin (FSM) based decision module [6]. The FSM uses cost functions which produce the behavior and vehicle trajectory deterministically and it is capable of switching among thirteen predefined different driving states. The FSM also has additional exception states in order to overcome stuck situations. At the high level, the FSM switches between the aforementioned driving states, and makes a transition to exception states.

Virginia Tech developed an autonomous vehicle named as *Odin* with a behavioral decision module involving using seven different behavior models (Route Driver, Passing Driver, Blockage Driver, Precedence Driver, Merge Driver, Left Turn Driver, Zone Driver) [8]. The decision module was developed under a hierarchical FSM and was capable of distinguishing different driving situations such as intersection, parking lot, and normal road driving. Additionally, a modified winner-takes-all approach based action selection mechanism was applied to select the most appropriate behavior in a given situation.

Team *AnnieWAY* developed an autonomous vehicle with a Concurrent Hierarchical State Machine (CHSM) based behavioral decision module [9]. In the hierarchical structure, 7 different behaviors were modeled. The high-level states were Replan, Drive, Zone, Intersection, Goal, and Global Recovery. The high level states had their sub-states for specialization and transition from a parent level states to the corresponding sub-states was modeled explicitly in order to reduce redundant states and total number of transitions. All high-level states had a sub-state for recovery to deal with a stuck situation. The Global Recovery state is defined to deal with failure of sub-states .

Finally, Team *Oshkosh* also developed an FSM based behavioral decision module for their autonomous vehicle, *TerraMax* [10]. Their decision module consisted of two parts, behavior supervisor and behavior mode & logic. The behavior supervisor module selected and supervised the appropriate behavior mode given a situation and the behavior mode & logic module had a set of

behavior modes, transition conditions between the behavior modes, and execution logic for each behavior mode. In total seventeen behavior modes were implemented to cover various driving situations. Team *Carolo* from TU Braunschweig applied a combinatorial approach of a rule-based decision and a behavioral model [11].

These rule based approaches were tested and showed successful performance in certain driving situations but very careful work was required to define the rules reflecting real driving situations. Hence, it is very hard to cover all driving situations by these rule based approaches and thus the performance is limited to those situations covered by the rule-base.

1.2.1.2 Markov Decision Process (MDP) based approach

In recent studies, several researchers applied the Markov Decision Process (MDP) approach for decision making of autonomous vehicles. Indeed, MDP is becoming increasingly popular in this field. Briefly, MDP is a mathematical tool to formulate decision and control problems under a potentially stochastic environment. MDP is defined by a tuple consisting of five elements (S, A, T, R, γ) which are the state space(S), action space(A), transition function(T), reward function(R) and discount factor(γ). With this in mind a driving environment is defined under an MDP framework and optimal action is produced to maximize the expected accumulated rewards, examples of which are given below. Figure 1.3 shows the concept of transition under the MDP framework [12].

Brechtel et al. [13] developed a probabilistic MDP based behavior planner for self-driving. They formulated a decision making problem for the traffic environment under a mathematical framework to obtain abstract symbolic states from complicated continuous temporal models defined by Dynamic Bayesian Networks (DBN). That is, they applied the combinatorial approach of DBN based continuous world prediction and discrete world MDP planning to derive symbolic states which can well present the real driving world. For the action space, they divided the spaces of longitudinal position, lateral position and velocity of the vehicle into equidistant intervals with-

[3]Reprinted with permission from *Reinforcement learning: An introduction*, by R. S. Sutton and A. G. Barto, 2018, MIT press, Massachusetts, USA. Copyright 2018 by MIT press.

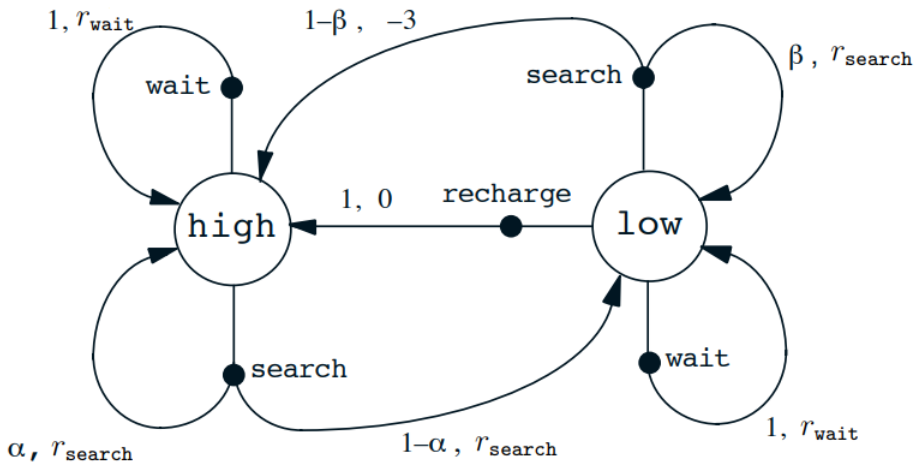


Figure 1.4: Example of transition graph under Markov Decision Process (MDP) framework [3]

out overlap, and the final state space was defined as a power set of states of all vehicles including the ego-vehicle. The action space was defined as a combination of longitudinal acceleration and lateral velocity. For approximating continuous probability density functions in discrete form, finite dirac-mixtures were used. A reward/penalty model was defined considering four different types of rewards/penalties, which were crash, make way, not rightmost lane, and comport. Their MDP based decision making model was evaluated on a two-lane highway scenario in simulation.

Ulbrich et al. [14] applied a probabilistic Partially Observable MDP (POMDP) for lane changing and suggested a two steps algorithm based on POMDP to reduce computational complexity of the POMDP for online application. In their approach, additional signal processing networks were applied to produce lane change possibility and lane change benefit considering relative distances and relative velocities. State space was defined as combinations of lane change possibility, lane change benefit and lane change progress status. Action space was defined including straight driving, initiate lane change and lane change abort. Reward function was defined according to combinations of defined states and actions. Transition probabilities were initialized by fitting with state transitions obtained from real driving situations and enhanced by expert knowledge.

Wei et al. [15] suggested a point-based MDP algorithms for single-lane autonomous driving behavior. Behavioral decision making was modeled as MDP and it was extended to a point-based

MDP to deal with POMDP in a computational efficient manner. In point-based MDP, uncertainty is considered in the first planning step and then observation as deterministic in the next step. In their approach, state space included distance to the leading vehicle, velocity of the leading vehicle, velocity of ego-vehicle, acceleration of ego-vehicle and jerk of ego-vehicle. Action is defined as a series of different possible longitudinal accelerations while a vehicle motion model was used to predict the next driving situation in a pre-defined time horizon. The cost function is calculated as the weighted sum of four different sub-costs, such as progress cost, comport cost, safety cost and fuel consumption cost.

Bandyopadhyay et al. [16] used mixed observability MDP (MOMDP) considering intentions of other road users which were assumed to be uncertain. They defined a set of intentions of others and made a motion model for each intention. These finite sets of intentions and motion models were incorporated into a MOMDP framework. The MOMDP was modeled as a tuple composed of joint state space of ego-vehicle state, state of others and intention of others, action space of ego-vehicle, observation space, transition probabilities for the ego-vehicle and others, the observation function, a reward function, and a discount factor. Their approach consisted of two steps. Firstly, other road user's intention is recognized, and then optimal action is executed in the second step. SARSP which is a leading point-based approximation was used to solve the MDMDP.

These MDP based approaches require very delicate works to model the driving environments under the MDP framework while the decision model is highly dependent on the defined MDP. Therefore, MDP should be modeled to represent real driving situations well.

1.2.1.3 Game theory based approach

Game theory is a branch of mathematics to deal with the interaction between interdependent players which can be considered as reasonable decision makers with respect to their preferences or utilities. Focusing on the capacity of game theory to treat the interaction between decision makers, some researchers applied game theory to high level decision making for self-driving to consider the interactions between the ego-vehicle and surrounding vehicles. In game theory based decision making approach, vehicles are considered as players.

Kita [17] proposed a game theory based merging-giveway interaction model under a two person non-zero sum game. In this study, interaction between the merging vehicle and the closest rear vehicle approaching the merging vehicle was taken into account. It was assumed that all vehicles (or drivers) know the strategies of each other and their game was formulated as a non-cooperative game with perfect information. The pure strategy of the merging vehicle was defined as 'merge' and 'pass', and the pure strategy of the approaching vehicle was defined as 'giveway' and 'do not giveway'. Payoff matrix was determined considering driving situations. That is, elements of the payoff matrix are functions of some variables which are defined using time to collision, time headway, distance to end of the lane. The parameters of linear functions in the payoff matrix were estimated using real traffic data by a maximum likelihood estimator (MLE). Liu et al. [18] extended Kita's works under an enhanced game theoretic framework in which minimum safety gaps are considered in the payoff functions. In their study, it was assumed that the approaching vehicle prefers to keep its car-following state and minimize speed variations while the merging vehicle wants to merge within the minimum possible time with consideration of safety. Parameters of their model were estimated by a bi-level programming problem using observed data.

Talebpour et al. [19] developed a game theoretic approach to deal with incomplete information. They used a two-person nonzero-sum non-cooperative game with incomplete information for vehicular interaction during lane changing. Harsanyi transformation [20] was used to transform a game of incomplete information to a game of imperfect information. In the approach, "nature as a player" was applied as was the case in Harsanyi's work. The nature moves first to choose a mandatory or discretionary lane change. It was assumed that the lane-changing vehicle can observe the nature's move but the lag vehicle cannot observe the nature's move, and the game is formulated in normal form. The pay-off functions were determined with the assumption that the lane-changing vehicle compares vehicle accelerations to avoid collision before and after lane change in order to make a lane change and the lag vehicle compares both acceleration and speed before and after lane change to avoid collision. The parameters of the defined pay-off functions were calibrated by the method of simulated moments (MSM) in [21] using real driving data.

Yu et al. [22] proposed a decision making model for lane change based on a Stackelberg game, which is a type of game with a leader and a follower. In this game, the leader acts first and then the follower reacts. Safety payoff and space payoff were defined and total payoff were calculated as a linear combination of these payoff with consideration of aggressiveness of the driver. It was assumed that each vehicle does not know the aggressiveness of the other driver and aggressiveness estimation algorithms were suggested. Firstly, the ego-vehicle driver assumes the other driver is a normal driver without aggressiveness and will make a prediction of the other's action. Then, the ego-vehicle driver observes the real action of the other driver and updates the estimation of aggressiveness by difference between real action and predicted action. Their model were validated in a driving simulator with robot and human drivers.

Wang et al. [23] formulated car-following and lane-changing behavior under differential game frameworks with a receding horizon. Kinematic vehicle model which has three states (longitudinal position, longitudinal speed and lateral position) was used to predict vehicle states, and control inputs were defined as longitudinal acceleration, the time initiating a lane change and direction of the lane change. The lateral position was determined as a function of control inputs, longitudinal acceleration and the time initiating a lane change and a trigonometric (cosine) function represented in [24] was applied. The cost function was defined to deal with interactions between the ego-vehicle and surrounding vehicles. They formulated a problem for non-cooperative control and cooperative control, while an extended kinematic vehicle model including two vehicle states was used while a cost function was defined considering the extended model including joint state and control spaces. Pontryagin's Maximum Principle based iterative algorithms were used to solve the problem and numerical simulation was performed to verify the performance of their controller.

Additionally, other researchers have studied Stackelberg game based decision making for self-driving [25] [26]. Also, a combination approach of game theory and receding horizon control (RHC) was presented in [27] for lane change maneuvers with incomplete information. Driver behavior model was studied based on differential game theory for non-cooperative and cooperative driving and Markovian game based approach was done in [28].

1.2.1.4 Reinforcement learning based approach

Reinforcement learning (RL) is a type of machine learning. RL enables an agent to learn by interacting with its environment. In other words, RL is an iterative learning process for the agent to find suitable action in a particular condition to maximize a certain reward. Recently, RL has received a great deal of attention in behavioral decision making for self-driving due to its learning abilities and in handling complex real driving situations including uncertainties. Figure 1.5 shows the concept of RL.

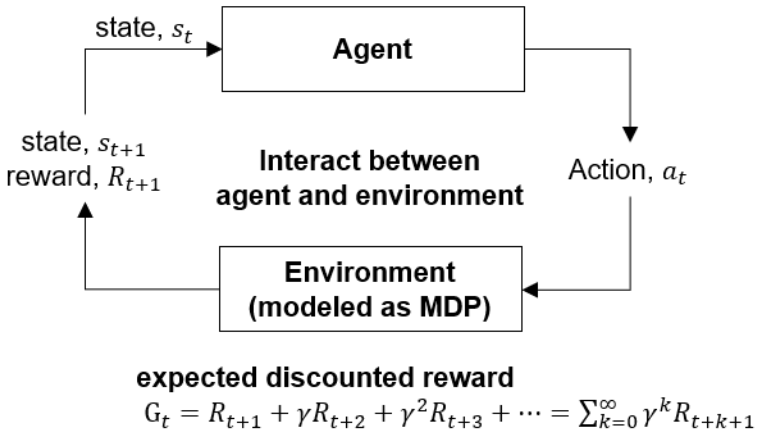


Figure 1.5: Reinforcment Learning Framework

Isele et al. [29] designed an RL based decision maker to negotiate intersections and merges onto highways. Deep Q-Network (DQN) was applied to learn the state-action value function (Q-function). The authors defined three different types of state representations, which are 1) *time-to-go* 2) *sequential actions* and 3) *creep-and-go*, and each state representation has a set of action definitions. In other words, *time-to-go* has a set of actions consisting of *wait* and *go*, and *sequential actions* has set of actions with *accelerate*, *decelerate* and *maintain constant velocity*. Similarly, *creep-and-go* consists of three actions which are *wait*, *move forward slowly* and *go*. In their re-

search, state was defined as a discretized grid in Cartesian coordinates with a binary definition of grid occupancy. The reward function consists of a reward for successful intersection navigation, penalty for collision and step penalty. Their research is limited in the specific situation of intersections and requires further research to address robustness.

Similarly, Wang et al. [30] proposed a DQN based decision making approach for the ramp merging problem. To deal with historical driving information, a Long Short-Term Memory (LSTM) based recurrent neural network was applied. To improve convergence rate and to guarantee that there always exists an optimal action, a quadratic type of Q function which has three separate neural networks was defined. State space was defined as 9-dimensional space. The state consists of speed, position, heading angle, and distances to the right and left lane of ego-vehicle, speeds of surrounding two vehicles and their positions. The action was composed of longitudinal acceleration and steering angle. Reward function has four terms for longitudinal acceleration, steering angle, speed and distance to the surrounding vehicles. Their research needs additional fine-tuning, refinement and performance evaluation.

Likewise, Wang and his colleagues [31] designed a DQN based decision maker for lane change maneuvers. Action space was defined with vehicle yaw acceleration while the state space was formulated with both vehicle driving information and road information. The state space is composed of ego-vehicle's speed, longitudinal acceleration, longitudinal position, lateral position, yaw angle, target lane, lane width, and road curvature. The reward function was formulated to include three terms which were yaw acceleration dependent reward for smoothness of maneuver, yaw rate dependent reward as additional sub-reward for smoothness of maneuver and lane change time dependent reward for functional efficiency. They used a quadratic type of Q function similar to their previous research in [30]. In their work, they only focused on lateral control based on DQN and they used an Intelligent Driver Model (IDM) for longitudinal vehicle control.

An et al. [32] suggested Deep Deterministic Policy Gradient(DDPG) algorithm based decision maker for lane change maneuvers. In other words, action space is continuous and DDPG based decision maker directly produces throttle and steering angle outputs. In their work, state space

was defined as including position, speed and heading angle information of ego-vehicle and the lag vehicle which is located in the target lane behind the ego-vehicle. Reward function was constructed with three terms which were collision, lane change completeness, and others related to driving lane on the road.

Li et al. [33] designed a Q learning based overtaking decision making model. They used a Q matrix to store the Q value and also used a vehicle dynamics model with 14 degrees of freedom for simulation purposes. The four nearest surrounding vehicles are only considered. State variables consist of the status of the ego-vehicle and four surrounding vehicles. Action was defined with two discrete actions for lane selection among two lanes. Reward was formulated as a function of the state variables. Their method was evaluated by comparison with a rule based expert system. Similarly, Li et al. [34] proposed an RL based decision making model for overtaking maneuvers. But they used DDPG for their the overtaking decision model. In their work, state variables were defined as the relative distance, relative angle, velocity of the leader vehicle, velocity of ego-vehicle, and distance to surrounding obstacles. The action space was formulated as the forward speed and angular velocity. Reward consists of four sub-rewards which are step reward for time cost, overtaking reward depending on relative maneuver to the leading vehicle, collision reward depending on distance to obstacles and navigation reward for reaching the goal position. In their method based on DDPG, continuous action space is dealt and decision making and control layers are incorporated.

Finally, some researchers have developed lane keeping assist based on Deep Deterministic Actor Critic Algorithm (DDAC) which can deal with a continuous action space [35]. Also, some researchers have studied RL based automatic parking assist in [36] and [37] by end-to-end learning.

1.2.2 Shared Control Between Human Driver and Machine

Shared control is a control strategy to share the control authority between the human driver and machine. There are two controllers, i.e. the human and the machine in a vehicle control loop shown in Figure 1.6 and these two controllers interact with each other in the shared control framework. Because human drivers have a will to drive, and want to enjoy driving and to feel

vehicle maneuver to their action, human drivers need to be considered in the vehicle control loop, even though autonomous driving may be available. Therefore, it is beneficial to take into account shared control between the human driver and the machine (i.e. the vehicle automation system).

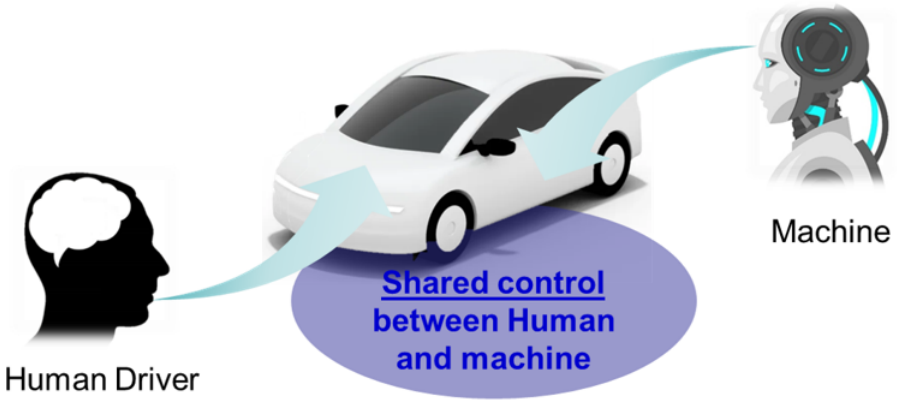


Figure 1.6: Shared control in vehicle control loop

1.2.2.1 Cooperative assist approach - direct shared control

In the direct shared control , the machine functions to assist a human driver for driving tasks. In other words, the human driver is the main controller and machine is the auxiliary controller to help the human driver perform the driving tasks such as lane keeping. Human driver holds the final control authority, and can override the machine's action. In this approach, a human driver model is used to take into account the human driver's behavior. The conceptual structure of the direct shared control is shown in Fig 1.7 [38].

Sentouh et al. [39] studied shared lateral control between the human and a steering assist controller. They used a visual angle based cybernetic human driver lane following model which includes the driver' arm neuromuscular model as well as their visual and kinesthetic perception. Steering assist controller was designed using a road-vehicle model augmented with the human

[4]Modified from "Indirect shared control of highly automated vehicles for cooperative driving between driver and automation" by R. Li, Y. Li, S. E. Li, E. Burdet, and B. Cheng, 2017. *arXiv preprint arXiv:1704.00866*

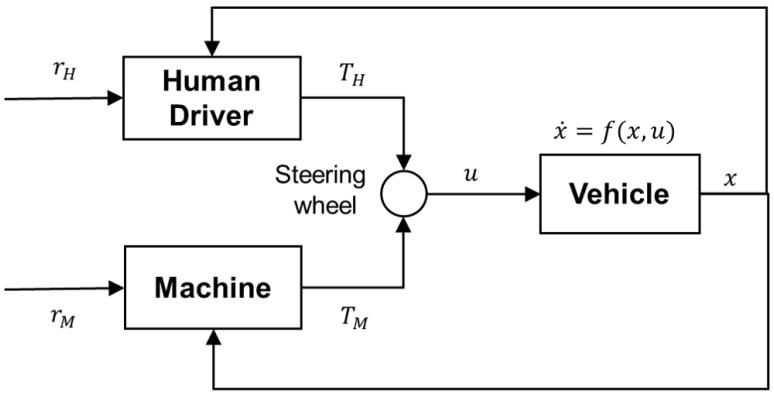


Figure 1.7: Structure of direct shared control [4]

driver lane following model. To handle conflict between the human and the machine, a control authority management method was devised via a Gaussian distribution function with lateral deviation and the difference between the human driver torque and steering assist torque. Their method basically assists the human driver to follow a lane via steering torque, but human driver takes all control authority when the difference between the driver torque and the steering assist torque is significant. This means they assume the reliability level of the machine is not sufficient and the human driver needs to take full-control when the machine action is not matched with the human driver's action. Similarly, Nguyen et al. [40] proposed a steering assist controller for lane keeping assist based on parallel distributed compensation (PDC) via a Takagi-Sugeno (T-S) fuzzy model to deal with the dependency of the vehicle dynamics on longitudinal velocity. They also used visual angles based cybernetic human driver lane following model. In their work, driver activity variable was introduced and defined as a function of the normalized driver torque and continuous driver monitoring information which has a 0 value when the driver does fails to recognize the driving situation and 1 when the driver is fully aware of the driving situation. Using the activity variable, assistance torque is adjusted. In the aforementioned approaches, the driver has the final control authority and can modulate his/her control action considering torque feedback from the machine.

Soualmi et al. [41] developed a haptic shared control strategy via steering assist torque to help human drivers to keep lane. They focused on lateral vehicle control while a simplified driver lane

keeping model was used. The driver lane keeping model is dependent on the lateral deviation error at a look ahead distance and the heading error. Lane keeping assist controller was designed using PDC via a T-S fuzzy model. In their approach, they also assumed that the human driver can recognize an obstacle which may not be undetected by the machine. Similarly, Inoue et al. [42] proposed a haptic steering guidance torque with direct yaw control to improve path tracking performance. The steering guidance torque and direct yaw control moment was obtained using the desired steering angle which is calculated from a (known) expert driver model. The haptic steering torque is simply calculated to be proportional to steering angle deviation from the desired steering angle, and direct yaw control moment is proportional to the desired steering angle. In these approaches, human driver models were considered in the controller design to deal with human driver behaviors but management of shared control authority between human and machine was not explicitly considered.

Guo et al. [43] [44] suggested model predictive control (MPC) based shared steering control for lane keeping assist. They used the lateral vehicle dynamics model in conjunction with an averaging road curvature model in [45]. The MPC based controller adjusts the weighting factor dynamically to allocate the control authority between the human drier and the controller. The dynamic weight factor has a value between zero to one and is used to scale the weight matrices in the cost function in MPC. The dynamic weight factor is set as zero when the human driver intends to control the vehicle and one otherwise. In other words, Human driver has the final authority to control the vehicle. But available lateral offset of the vehicle to the current lane center is considered as constraints in MPC to prevent a collision due to infeasible lane change by the human driver. The lateral offset constraints are determined dynamically by taking into account time-to-collision (TTC) to vehicles in adjacent lanes. If infeasible, lane change is performed by the human driver, MPC based controller increase its steering torque output to prevent infeasible lane change and the human driver can recognize potential hazards via steering torque (haptic) feedback by the controller.

Saleh et al. [46] proposed an $\mathcal{H}_2$ preview control based lateral shared control for lane keep-

ing using a vehicle–road model and cybernetic driver model for lane keeping to deal with human driver-vehicle interactions. Ercan et al. [47] designed an MPC based controller for lane keeping via steering assist torque. They considered impedance model of the human driver's arm to investigate the effect of different mechanical impedances on lane keeping performance. They took into account human driver behavior as uncertainty and analyzed the robust stability of the system with respect to uncertain driver behavior. Some researchers [48] studied MPC based shared steering control for avoiding obstacles and to ensure vehicle stability using two types of safety envelopes which are side slip and yaw rate dependent stable handling envelope and vehicle position dependent environment envelope. Some researchers have designed steering assistance torque based shared control for avoiding an emergency obstacle [49] and others have proposed shared control algorithms for lane departure prevention with consideration of the safe envelope with respect to steering angle.

1.2.2.2 Cooperative assist approach - indirect shared control

In the indirect shared control approaches, the machine transforms the human driver input and produces the final control action while the human driver indirectly controls the vehicle. In this case, the machine can arbitrarily modulate the human driver's control input. In other words, the machine has the final control authority and human driver input can in principle be preempted by the machine's input transformation, if necessary. The conceptual structure of the indirect shared control is shown in Fig 1.8 [38].

Li et al. proposed an MPC based indirect shared control which can dynamically allocate control authority considering the driver's intention [50][51][38]. They considered linear lateral vehicle dynamics to follow reference trajectories represented by lateral position and yaw angle while steering angle was dealt with as a control input. The machine's control strategy was designed by MPC while human driver's control strategy was modeled by MPC with constraints of machine's input transformation with respect to the driver's desired control authority. The human driver's desired

[5]Modified from "Indirect shared control of highly automated vehicles for cooperative driving between driver and automation" by R. Li, Y. Li, S. E. Li, E. Burdet, and B. Cheng, 2017. *arXiv preprint arXiv:1704.00866*

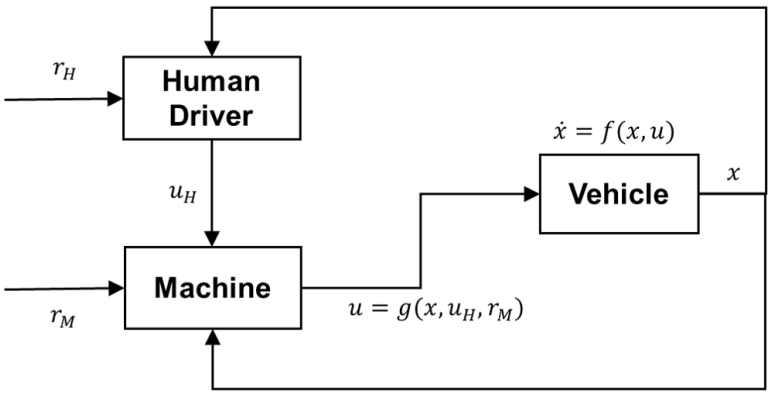


Figure 1.8: Structure of indirect shared control [5]

control authority was estimated in real time by a sliding-window least-squares estimator in their work[50]. The final control input was determined as the weighted summation of human driver's control input and machine's control input, and the weights were defined so as to sum up to one. The control authority was allocated in accordance with the estimated driver's desired authority intention. To make smooth authority allocation and to avoid system instability, a moving average filter was applied to determine the control authority allocation from the estimated human driver's authority intention[50]. The weight matrices in the cost of human driver's MPC control strategy was considered as trust level of human driver with respect to the machine[51]. In [38], automatic authority weight switching method was studied using a cumulative error of human driver input which is calculated as the difference between the estimated human driver's input and real human driver's input. In their automatic authority weight switching method, control authority weight is switched by two pre-defined values. One of pre-defined values is for relieving control effort of human driver and the other is for allocation of more control authority to the human driver. In their work, authority allocation is dependent on not machine but human driver, even though machine incorporates the final input transformation function. That means the human driver plays as the final authority holder in their approach.

Jiang et al. [52][53] proposed a shared control algorithm based on the current relative distance between the ego vehicle and an obstacle, and applied the shared control law on a kinematic vehicle

model. They designed a reference path tracking controller based using a Lyapunov based method. Human driver was also considered as a controller which can be designed based on the same Lyapunov based method. They defined shared control input as the weighted summation of human driver's input and machine's input. The weight referred to as shared function represents allocation of control authority to human driver and machine. The shared function was defined according to three kinds of regions which are safe, dangerous and hysteresis regions. The shared function is set to one in a safe region where there is no collision risk but as zero in dangerous region where there exists a collision risk. In the hysteresis region, the shared function is determined as one or zero considering the previous region. In their approach, machine is dealt with the final control authority holder and that means it was assumed that the machine can detect obstacles which may not be detected by a human driver. They extended their research to vehicle dynamic model in [54]. In their works, once again, machine was taken as the final authority holder. That means they assumed machine can detect a obstacle which may not be observed by a human driver.

1.2.2.3 Differential game theory based approach

Differential game is a mathematical tool to formulate a control problem with several controllers with its own control targets based on system dynamics represented as differential equations. Interaction between human driver and machine and their interactive control strategies can be analyzed in the differential game theory framework. Several previous researches for shared control between machine and human driver have been studied under differential game theory.

Na et al. [55][56] showed in his researches how the vehicle's moving trajectory converge to human driver's reference or machine's reference according to different control parameters in a single lane change situation based on LQ optimization and differential game theory which can be formulated as model predictive control with receding horizon . But in their works the issue of how to share the control authority according to a driving condition was not addressed.

Ji et al. [57][58] extended Na's research in [55] and [56] to add a dynamics of steering system to consider steering torque interaction between a human driver and machine. In their works the issue of how to dynamically allocates control authority according to a driving condition was not

addressed.

1.3 Organization of the Dissertation

The dissertation will consist of three parts.

Chapter 2 is about shared control based on game theoretic model predictive control (MPC) for human driver and machine. In Section 2.1, theoretical background and mathematical formulation of the problem are described. In Section 2.2, shared control strategy is introduced, and in Section 2.3 simulation studies are explained. In Section 2.4, conclusions are drawn.

Chapter 3 is about reinforcement learning (RL) based decision making for autonomous driving. In Section 3.1, theoretical background of reinforcement learning is described. In Section 3.2, decision making model based on RL for highway driving is introduce, and in Section 3.3 performance comparison study is specified. In Section 3.4, conclusions are stated.

Chapter 4 is about study on application reinforcement learning (RL) based decision model with game theoretic MPC based shared controller. In Section 4.1, simulation studies are described, and in Section 4.2 conclusions are drawn.

In Chapter 5, final summary and conclusions are described and future works are discussed.

2. SHARED CONTROL BASED ON GAME THEORETIC MODEL PREDICTIVE CONTROL FOR HUMAN DRIVER AND MACHINE

This chapter addresses the shared control strategy between human driver and machine. In this research, it is understood that human driver and machine are game players to control a vehicle according to their purposes. Human driver and machine are well-functioning vehicle controllers and can be considered as model predictive control (MPC) based vehicle controller with certain periods of receding horizons [55]. Therefore, shared control between them is studied based on game theoretical model predictive control (MPC) in this research. Four different types of game are investigated - non-cooperative game with simultaneous move, non-cooperative game with leader-and-follower, non-cooperative game with sequential move and cooperative game. Shared control strategy for fully mixed control authority is proposed to be capable of driving safely to deal with high collision risk. Also, it can be analyzed that human driver and machine can establish and change a game according to driving situations. Thus, game transition model is studied based on finite state machine (FSM) to represent dynamic game change between the two players according to driving situations. This section consists of following subsections. Theoretical backgrounds and mathematical formulations are stated in Section 2.1, and Section 2.2 provides details of the proposed shared control stategy and game transition model. Simulation studies are given in Section 2.3 Finally, conclusions are specified in Section 2.4.

2.1 Theoretical backgrounds and Mathematical Formulation of a Problem

2.1.1 Vehicle dynamic model

In this research, 3 degrees of freedom (DOF) nonlinear vehicle dynamics are considered with nonlinear tire force. Also, 1 DOF steering model is taken into account to deal with steering torque input. Even though steering system can be modeled by 2 or 3 DOF with consideration of stiffnesses at steering column or at motor driving shaft, 1 DOF model can be enough for study focusing on vehicle-level behavior with steering torque input. Therefore, 1 DOF steering model is used in this

research. The whole vehicle system including steering system and tire force model is depicted in Figure 2.1.

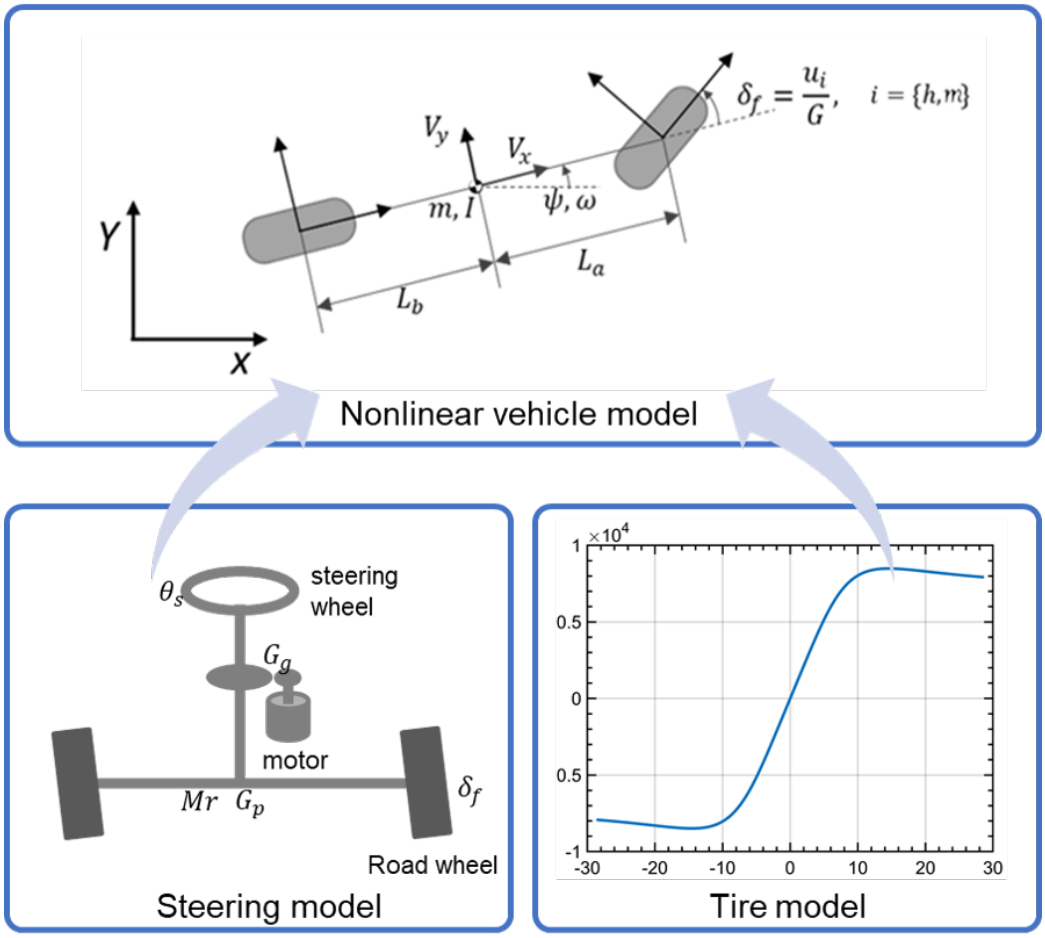


Figure 2.1: Vehicle dynamic model

The vehicle model has two control inputs in the perspective of longitudinal and lateral behaviors. The longitudinal acceleration can be dealt as an input for longitudinal behavior and steering torque can be considered as an input for lateral behavior. It can be understood that human driver and machine are able to control longitudinal acceleration of vehicle fast enough with respect to their demanded longitudinal acceleration, so that longitudinal acceleration is directly dealt as longitudinal control input rather than throttle and brake input in order to have a physical meaning in

terms of vehicle-level behavior. Therefore, longitudinal acceleration and steering torque inputs are defined as each player's control inputs. In regard of tire model, reduced Pacejka tire model is used. The Pacejka tire model is an empirical model which is formulated as a mathematical curve to match the measured data. Because the Pacejka tire model well represents tire's nonlinear characteristics, it is commonly used for vehicle dynamic studies. The reduced Pacejka tire model is a simplified model of Pacejka tire model and the reduced Pacejka model well represent tire's characteristics on static normal force and road surface with fewer parameters. The normal highway driving situation can be handled with static normal force and road surface, the reduced Pacejka model is used in this study. It is more beneficial to reduce complexity of the whole vehicle system without reduction of the model's ability of representation under intended conditions. The vehicle system dynamics can be expressed as below.

$$
\begin{aligned}
\dot{X} &= v_x \cos\psi - v_y \sin\psi \\
\dot{Y} &= v_x \sin\psi + v_y \cos\psi \\
\dot{\psi} &= \omega \\
\dot{v_x} &= \omega v_y + a_{x,m} + a_{x,d} \\
\dot{v_y} &= -\omega v_y + \frac{1}{m}\left(F_{y,f}\cos\delta_f + F_{y,r}\right) \\
\dot{\omega} &= \frac{1}{I_z}\left(L_f F_{y,f}\cos\delta_f - L_r F_{y,r}\right) \\
\ddot{\theta_s} &= \frac{1}{J_s + G_p^2 M_r}\left(\tau_d + G_g\tau_m - K_r G_p^2\theta_s - (B_c + B_r G_p^2)\dot{\theta_s} - \frac{\eta_t F_{y,f}}{G}\right)
\end{aligned}
\tag{2.1}
$$

$$
\begin{aligned}
\text{where, } F_{y,f} &= D_f \sin(C_f \arctan B_f \alpha_f) \\
F_{y,r} &= D_r \sin(C_r \arctan B_r \alpha_r) \\
\alpha_f &= \delta_f - \arctan\left(\frac{L_f\omega + v_y}{v_x}\right) \\
\alpha_r &= -\arctan\left(\frac{L_r\omega - v_y}{v_x}\right) \\
\delta_f &= \frac{\theta_s}{G}
\end{aligned}
$$

The states of system and control inputs are defined as X(longitudinal position), Y(lateral position), ψ(yaw angle), v_x(longitudinal velocity), v_y(lateral velocity), ω(yaw rate), θ_s(steering angle) and $\dot{\theta_s}$(steering velocity). The control inputs are defined as τ_d(steering torque input) and $acc_{x,d}$(acceleration input) for a human driver, and τ_m (motor torque input) and $acc_{x,m}$ (acceleration input) for a machine. The subscript d and m means "driver" and "machine" respectively. Equation (2.2) shows the defined states and control inputs.

$$
\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \psi \\ v_x \\ v_y \\ \omega \\ \theta_s \\ \dot{\theta_s} \end{bmatrix}, \quad \mathbf{u_m} = \begin{bmatrix} u_{m,1} \\ u_{m,2} \end{bmatrix} = \begin{bmatrix} a_{x,m} \\ \tau_m \end{bmatrix}, \quad \mathbf{u_d} = \begin{bmatrix} u_{d,1} \\ u_{d,2} \end{bmatrix} = \begin{bmatrix} a_{x,d} \\ \tau_d \end{bmatrix} \tag{2.2}
$$

The system outputs for human driver and machine are determined as X(longitudinal position), Y(lateral position), ψ(yaw angle) and v_x(longitudinal velocity) among vehicle states. The overall system can be expressed in Equation (2.3) with the defined states and control inputs and its parameters are summarized in Table 2.1. Details of system dynamics are given in Appendix A.1

Table 2.1: Parameters of vehicle model

Parameter	Description
m	the mass of vehicle
I_z	the yaw moment of inertia of vehicle
L_f	the distance from the center of gravity of vehicle to the front axle
L_r	the distance from the center of gravity of vehicle to the rear axle
G	the overall steering gear ratio (steering angle/road wheel angle)
G_g	gear ratio between steering column and motor driving shaft
G_p	gear ratio of steering rack and column
M_r	mass of steering rack
K_r	stiffness of steering rack
B_r	viscous damping of steering rack
J_c	equivalent inertia of steering column and steering wheel
B_c	viscous damping of steering column
η_t	pneumatic trail of front tire
D_f, C_f, B_f	Pacejka model parameters of front tire
D_r, C_r, B_r	Pacejka model parameters of rear tire

$$\begin{aligned} \dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) \\ \mathbf{y_m} &= \mathbf{g_m}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) \\ \mathbf{y_d} &= \mathbf{g_d}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) \end{aligned} \tag{2.3}$$

$$\begin{aligned} \text{where, } \mathbf{f} &= [f_1, f_2, f_3, f_4, f_5, f_6, f_7, f_8]^T \\ \mathbf{g_m} &= [g_{m,1}, g_{m,2}, g_{m,3}, g_{m,4}]^T \\ \mathbf{g_d} &= [g_{d,1}, g_{d,2}, g_{d,3}, g_{d,4}]^T \end{aligned}$$

2.1.2 Formulation of Game Theoretic Model Predictive Control

The conventional Model Predictive Control (MPC) is based on discrete time linear models. The MPC problem is formulated as Quadratic Programming (QP) problem with consideration for prediction and control horizons. In this research, nonlinear vehicle system described in Equation (2.3) is taken into account and linearization based approach is used to deal with nonlinearity of the vehicle system [59]. Successively, a continuous linear system is approximated at an operating point and is discretized, and then linear model based game theoretic MPC is formulated. The algorithm 1 and Figure 2.2 shows the procedure of linear system approximation based MPC.

Algorithm 1 Linear approximation based game theoretic MPC

Input

Initial state $\mathbf{x}(0)$, Prediction horizon N_p, Control horizon N_c, Sampling step T_s

Repeat

1. Linearize the system at $\mathbf{x}(t)$
   $\rightarrow$ $A_t, B_{t,m}, B_{t,d}, C_{t,m}, C_{t,d}, D_{t,m}, D_{t,d}$
2. Dicretize the linearized system by T_s
   $\rightarrow$ $A_k, B_{k,m}, B_{k,d}, C_{k,m}, C_{k,d}, D_{k,m}, D_{k,d}$
3. Formulate game theoretic MPC with N_p and N_c
   $\rightarrow$ $F_k, \Phi_{k,m}\Phi_{k,d}, E_k, K_k$
4. Formulate QP problem ($i = d, m$)
   $\rightarrow$ $J_i = \frac{1}{2}\overrightarrow{\mathbf{u}}_i^T G_i \overrightarrow{\mathbf{u}}_i + \overrightarrow{\mathbf{u}}_i^T W_i$
   s.t. $\overrightarrow{\mathbf{u}}_{i,l} \leq \overrightarrow{\mathbf{u}}_i \leq \overrightarrow{\mathbf{u}}_{i,u}$
   $\triangle\overrightarrow{\mathbf{u}}_{i,l} \leq \triangle\overrightarrow{\mathbf{u}}_i \leq \triangle\overrightarrow{\mathbf{u}}_{i,u}$
5. Solve QP problem
   $\rightarrow$ $\overrightarrow{\mathbf{u}}_m, \overrightarrow{\mathbf{u}}_d$
6. Take and apply the first elements of the QP solution and go back to 1.

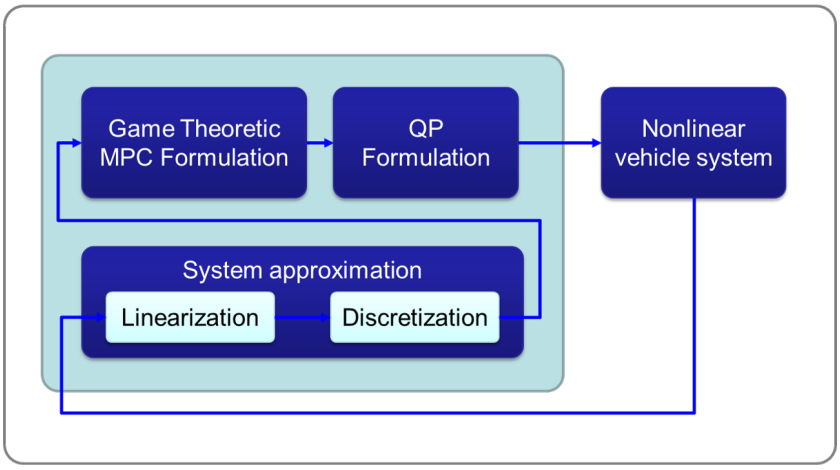


Figure 2.2: Game Theoretic conceptual diagram

2.1.2.1 Linear approximation of nonlinear system

The overall system dynamics with two players, human driver and machine, are derived in (2.4). Consider the nonlinear vehicle system with states $\mathbf{x} \in R^{n_x}$, inputs, $\mathbf{u_m} \in R^{n_u}$, and $\mathbf{u_d} \in R^{n_u}$ and outputs, $\mathbf{y_m} \in R^{n_o}$ and $\mathbf{y_d} \in R^{n_o}$.

$$\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d})$$

$$\mathbf{y_m} = \mathbf{g_m}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d})$$

$$\mathbf{y_d} = \mathbf{g_d}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d})$$

where $\mathbf{f}$ is a function mapping $R^{n_x} \times R^{n_u} \times R^{n_u} \rightarrow R^{n_x}$ and $\mathbf{g_m}$ and $\mathbf{g_d}$ are a function mapping $R^{n_x} \times R^{n_u} \times R^{n_u} \rightarrow R^{n_o}$.

The nonlinear vehicle system can be linearized at an operating point with Jacobian matrix. Let the operating point as $\overline{\mathbf{x}}$, $\overline{\mathbf{u}}_\mathbf{d}$ and $\overline{\mathbf{u}}_\mathbf{m}$ respectively, and the Jacobian matrices are obtained as

$$
A_t = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \frac{\partial f_1}{\partial x_2} & \dots & \frac{\partial f_1}{\partial x_{n_x}} \\ \frac{\partial f_2}{\partial x_1} & \frac{\partial f_2}{\partial x_2} & \dots & \frac{\partial f_2}{\partial x_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial f_{n_x}}{\partial x_1} & \frac{\partial f_{n_x}}{\partial x_2} & \dots & \frac{\partial f_{n_x}}{\partial x_{n_x}} \end{bmatrix}
$$

$$
B_{t,m} = \begin{bmatrix} \frac{\partial f_1}{\partial u_{m,1}} & \frac{\partial f_1}{\partial u_{m,2}} \\ \frac{\partial f_2}{\partial u_{m,1}} & \frac{\partial f_2}{\partial u_{m,2}} \\ \vdots & \vdots \\ \frac{\partial f_{n_x}}{\partial u_{m,1}} & \frac{\partial f_{n_x}}{\partial u_{m,2}} \end{bmatrix}, B_{t,d} = \begin{bmatrix} \frac{\partial f_1}{\partial u_{d,1}} & \frac{\partial f_1}{\partial u_{d,2}} \\ \frac{\partial f_2}{\partial u_{d,1}} & \frac{\partial f_2}{\partial u_{d,2}} \\ \vdots & \vdots \\ \frac{\partial f_{n_x}}{\partial u_{d,1}} & \frac{\partial f_{n_x}}{\partial u_{d,2}} \end{bmatrix}
$$

$$
C_{t,m} = \begin{bmatrix} \frac{\partial g_{m,1}}{\partial x_1} & \frac{\partial g_{m,1}}{\partial x_2} & \dots & \frac{\partial g_{m,1}}{\partial x_{n_x}} \\ \frac{\partial g_{m,2}}{\partial x_1} & \frac{\partial g_{m,2}}{\partial x_2} & \dots & \frac{\partial g_{m,2}}{\partial x_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{m,n_o}}{\partial x_1} & \frac{\partial g_{m,n_o}}{\partial x_2} & \dots & \frac{\partial g_{m,n_o}}{\partial x_{n_x}} \end{bmatrix}, C_{t,d} = \begin{bmatrix} \frac{\partial g_{d,1}}{\partial x_1} & \frac{\partial g_{d,1}}{\partial x_2} & \dots & \frac{\partial g_{d,1}}{\partial x_{n_x}} \\ \frac{\partial g_{d,2}}{\partial x_1} & \frac{\partial g_{d,2}}{\partial x_2} & \dots & \frac{\partial g_{d,2}}{\partial x_{n_x}} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial g_{d,n_o}}{\partial x_1} & \frac{\partial g_{d,n_o}}{\partial x_2} & \dots & \frac{\partial g_{d,n_o}}{\partial x_{n_x}} \end{bmatrix}
$$

$$
D_{t,mm} = \begin{bmatrix} \frac{\partial g_{m,1}}{\partial u_{m,1}} & \frac{\partial g_{m,1}}{\partial u_{m,2}} \\ \frac{\partial g_{m,2}}{\partial u_{m,1}} & \frac{\partial g_{m,2}}{\partial u_{m,2}} \\ \vdots & \vdots \\ \frac{\partial g_{m,n_x}}{\partial u_{m,1}} & \frac{\partial g_{m,n_x}}{\partial u_{m,2}} \end{bmatrix}, D_{t,md} = \begin{bmatrix} \frac{\partial g_{m,1}}{\partial u_{d,1}} & \frac{\partial g_{m,1}}{\partial u_{d,2}} \\ \frac{\partial g_{m,2}}{\partial u_{d,1}} & \frac{\partial g_{m,2}}{\partial u_{d,2}} \\ \vdots & \vdots \\ \frac{\partial g_{m,n_x}}{\partial u_{d,1}} & \frac{\partial g_{m,n_x}}{\partial u_{d,2}} \end{bmatrix}
$$

$$
D_{t,dd} = \begin{bmatrix} \frac{\partial g_{d,1}}{\partial u_{d,1}} & \frac{\partial g_{d,1}}{\partial u_{d,2}} \\ \frac{\partial g_{d,2}}{\partial u_{d,1}} & \frac{\partial g_{d,2}}{\partial u_{d,2}} \\ \vdots & \vdots \\ \frac{\partial g_{d,n_x}}{\partial u_{d,1}} & \frac{\partial g_{d,n_x}}{\partial u_{d,2}} \end{bmatrix}, D_{t,dm} = \begin{bmatrix} \frac{\partial g_{d,1}}{\partial u_{m,1}} & \frac{\partial g_{d,1}}{\partial u_{m,2}} \\ \frac{\partial g_{d,2}}{\partial u_{m,1}} & \frac{\partial g_{d,2}}{\partial u_{m,2}} \\ \vdots & \vdots \\ \frac{\partial g_{d,n_x}}{\partial u_{m,1}} & \frac{\partial g_{d,n_x}}{\partial u_{m,2}} \end{bmatrix} \tag{2.4}
$$

The linearziaed system can be written as

$$
\dot{\mathbf{x}} - \dot{\overline{\mathbf{x}}} = A_t(\mathbf{x} - \overline{\mathbf{x}}) + B_{t,m}(\mathbf{u_m} - \overline{\mathbf{u}}_\mathbf{m}) + B_{t,d}(\mathbf{u_d} - \overline{\mathbf{u}}_\mathbf{d}) \tag{2.5}
$$

and the linearized system in (2.5) can be re-written as

$$\dot{\mathbf{x}} = A_t\mathbf{x} + B_{t,m}\mathbf{u_m} + B_{t,d}\mathbf{u_d} + \dot{\overline{\mathbf{x}}} - A_t\overline{\mathbf{x}} - B_{t,m}\overline{\mathbf{u}}_\mathbf{m} - B_{t,d}\overline{\mathbf{u}}_\mathbf{d} \tag{2.6}$$

Finally, the linearized system can be specified with introduction of new term K_t as below.

$$\dot{\mathbf{x}} = A_t\mathbf{x} + B_{t,m}\mathbf{u_m} + B_{t,d}\mathbf{u_d} + K_t \tag{2.7}$$

$$\mathbf{y_m} = C_{t,m}\mathbf{x} + D_{t,mm}\mathbf{u_m} + D_{t,md}\mathbf{u_d} \tag{2.8}$$

$$\mathbf{y_d} = C_{t,d}\mathbf{x} + D_{t,dd}\mathbf{u_d} + D_{t,dm}\mathbf{u_{dm}} \tag{2.9}$$

where, $K_t = \dot{\overline{\mathbf{x}}} - A_t\overline{\mathbf{x}} - B_{t,m}\overline{\mathbf{u}}_\mathbf{m} - B_{t,d}\overline{\mathbf{u}}_\mathbf{d}$

2.1.2.2 Discretization of linearized system

The linearized system in Equation (2.7)-(2.9) is a continuous time system and it needs to be discretized to design MPC based controller. The solutions of Equation (2.7) for state values $\mathbf{x}$ at time kT_s and $(k+1)T_s$ is given as below.

$$\mathbf{x}((k+1)T_s) = e^{A_t(k+1)T_s}\mathbf{x}(0) + e^{A_t(k+1)T_s}\int_0^{(k+1)T_s} e^{-A_t\tau}(B_{t,m}\mathbf{u_m}(\tau) + B_{t,d}\mathbf{u_d}(\tau) + K_t)\,d\tau \tag{2.10}$$

$$\mathbf{x}(kT_s) = e^{A_tkT_s}\mathbf{x}(0) + e^{A_tkT_s}\int_0^{kT_s} e^{-A_t\tau}(B_{t,m}\mathbf{u_m}(\tau) + B_{t,d}\mathbf{u_d}(\tau) + K_t)\,d\tau \tag{2.11}$$

By muliplying Equation (2.11) by $e^{A_tT_s}$, the equation below is obtained.

$$e^{A_t(k+1)T_s}\mathbf{x}(0) = e^{A_tT_s}\mathbf{x}(kT_s) - e^{A_t(k+1)T_s}\int_0^{kT_s} e^{-A_t\tau}(B_{t,m}\mathbf{u_m}(\tau) + B_{t,d}\mathbf{u_d}(\tau) + K_t)\,d\tau \tag{2.12}$$

By substituting Equation (2.12), Equation (2.10) can be rewritten as

$$\mathbf{x}((k+1)T_s) = e^{A_t T_s}\mathbf{x}(kT_s) + e^{A_t(k+1)T_s}\int_{kT_s}^{(k+1)T_s} e^{-A_t\tau}(B_{t,m}\mathbf{u_m}(\tau) + B_{t,d}\mathbf{u_d}(\tau) + K_t)\,d\tau \tag{2.13}$$

It is taken into account that $B_{t,m}$ and $B_{t,d}$ are constant and $\mathbf{u_m}$ and $\mathbf{u_d}$ are constant within the interval from kT_s to $(k+1)T_s$, and then Equation (2.13) can be reorganized as below.

$$\begin{aligned}\mathbf{x}((k+1)T_s) = e^{A_t T_s}\mathbf{x}(kT_s) &+ \int_{kT_s}^{(k+1)T_s} e^{-A_t[(k+1)T_s-\tau]}\,d\tau B_{t,m}\mathbf{u_m}(\tau) \\ &+ \int_{kT_s}^{(k+1)T_s} e^{-A_t[(k+1)T_s-\tau]}\,d\tau B_{t,d}\mathbf{u_d}(\tau) \\ &+ \int_{kT_s}^{(k+1)T_s} e^{-A_t[(k+1)T_s-\tau]}\,d\tau K_t, \qquad \tau \in [kT_s, (k+1)T_s]\end{aligned} \tag{2.14}$$

Introducing new variable $\lambda = (k+1)T_s - \tau$, λ has a range from T_s to 0 and a relationship of $d\lambda = -d\tau$. Then, Equation (2.14) is expressed as below with respect to the variable λ.

$$\begin{aligned}\mathbf{x}((k+1)T_s) = e^{A_t T_s}\mathbf{x}(kT_s) &+ \int_0^{T_s} e^{A_t\lambda}\,d\lambda B_{t,m}\mathbf{u_m}(kT_s) \\ &+ \int_0^{T_s} e^{A_t\lambda}\,d\lambda B_{t,d}\mathbf{u_d}(kT_s) \\ &+ \int_0^{T_s} e^{A_t\lambda}\,d\lambda K_t, \qquad \lambda \in [0, T_s]\end{aligned} \tag{2.15}$$

Using $\frac{d}{dt}e^{A_t T_s} = A_t e^{A_t T_t} = e^{A_t T_t}A_t$, the discretized system can be otained as below from Equation

(2.15) in consideration of independence of $\mathbf{g_m}$ and $\mathbf{g_d}$ on control inputs, $\mathbf{u_m}$ and $\mathbf{u_m}$.

$$
\begin{aligned}
&A_k = e^{A_t T_s} \\
&B_{k,m} = \int_0^{T_s} e^{A_t \lambda}\, d\lambda B_{t,m} = A_t^{-1} \int_0^{T_s} A^t e^{A_t \lambda}\, d\lambda B_{t,m} = A_t^{-1}(e^{A_t T_s} - I) B_{t,m} \\
&B_{k,d} = \int_0^{T_s} e^{A_t \lambda}\, d\lambda B_{t,d} = A_t^{-1} \int_0^{T_s} A^t e^{A_t \lambda}\, d\lambda B_{t,d} = A_t^{-1}(e^{A_t T_s} - I) B_{t,d} \\
&K_k = \int_0^{T_s} e^{A_t \lambda}\, d\lambda K_t = A_t^{-1} \int_0^{T_s} A^t e^{A_t \lambda}\, d\lambda K_t = A_t^{-1}(e^{A_t T_s} - I) K_t \\
&C_{k,m} = C_{t,m} \\
&C_{k,d} = C_{t,d} \\
&D_{k,mm} = D_{t,mm} = 0 \\
&D_{k,md} = D_{t,md} = 0 \\
&D_{k,dd} = D_{t,dd} = 0 \\
&D_{k,dm} = D_{t,dm} = 0
\end{aligned}
\tag{2.16}
$$

2.1.3 Formulation of Model Predictive Control

Model Predictive Control (MPC) is a kind of optimal control strategy based on mathematical optimization to predict future system response and future control inputs for finite prediction/control horizons in consideration of performance index also known as cost function. The conventional MPC deals with single player control problem in which there exists one controller which have its own control strategy. But game theory is a mathematical model which is capable of dealing with interaction among rational players. Human driver and machine recognize the existence of each other in the vehicle control loop and they can be considered as rational players who want to control a vehicle based on their own control strategies. In other words, it is a reasonable assumption that they act as game players to control a vehicle under game frameworks. In this research, interaction and control behavior between human driver and machine are studied under game theoretic frameworks, and the conventional MPC based approach is extended to game theoretic problems under different types of game which are non-cooperative game with simultaneous move, non-cooperative game with leader-and-follower, non-cooperative game with sequential move and cooperative game.

2.1.3.1 Derivation of matrix form for Model Predictive Control

Based on derived linear discrete system model, future system state variables are predicted with the future control inputs which are optimized variables for prediction horizon. In regards to future control inputs, control horizon is also used to consider trajectory of future control inputs. The length of prediction horizon is denoted as N_p. Similarly, the length of control horizon is denoted as N_c and it is defined to be less than or equal to the prediction horizon. With prediction horizon, N_p, and control horizon, N_c, the future system states are predicted and below.

$$
\begin{aligned}
\mathbf{x}(k+1) &= A_k\mathbf{x}(k) + B_{k,d}\mathbf{u_d}(k) + B_{k,m}\mathbf{u_m}(k) + K_k \\
\mathbf{x}(k+2) &= A_k^2\mathbf{x}(k) + A_kB_{k,d}\mathbf{u_d}(k) + A_kB_{k,m}\mathbf{u_m}(k) + A_kK_k \\
&\quad + B_{k,d}\mathbf{u_d}(k+1) + B_{k,m}\mathbf{u_m}(k+1) + K_k \\
&\vdots \\
\mathbf{x}(k+N_p) &= A_k^{N_p}\mathbf{x}(k+1) + A_k^{N_p-1}B_{k,d}\mathbf{u_d}(k) + A_k^{N_p-1}B_{k,m}\mathbf{u_m}(k) + A_k^{N_p-1}K_k + \cdots \\
&\quad + A_k^{N_p-N_c}B_{k,d}\mathbf{u_d}(k+N_c-1) + A_k^{N_p-N_c}B_{k,m}\mathbf{u_m}(k+N_c-1) + A_k^{N_p-N_c}K_k
\end{aligned} \tag{2.17}
$$

Then, the future system outputs can be obtained using the predicted future system states as below with notation, $i = \{d, m\}$.

$$
\begin{aligned}
\mathbf{y_i}(k+1) &= C_{k,i}A_k\mathbf{x}(k) + C_{k,i}B_{k,d}\mathbf{u_d}(k) + C_{k,i}B_{k,m}\mathbf{u_m}(k) + C_{k,i}K_k \\
\mathbf{y_i}(k+2) &= C_{k,i}A_k^2\mathbf{x}(k+1) + C_{k,i}A_kB_{k,d}\mathbf{u_d}(k) + C_{k,i}A_kB_{k,m}\mathbf{u_m}(k) + C_{k,i}A_kK_k \\
&\quad + C_{k,i}B_{k,d}\mathbf{u_d}(k+1) + C_{k,i}B_{k,m}\mathbf{u_m}(k+1) + C_{k,i}K_k \\
&\vdots \\
\mathbf{y_i}(k+N_p) &= C_{k,i}A_k^{N_p}\mathbf{x}(k+1) + C_{k,i}A_k^{N_p-1}B_{k,d}\mathbf{u_d}(k) + C_{k,i}A_k^{N_p-1}B_{k,m}\mathbf{u_m}(k) + C_{k,i}A_k^{N_p-1}K_k \\
&\quad + \cdots + C_{k,i}A_k^{N_p-N_c}B_{k,d}\mathbf{u_d}(k+N_c-1) + C_{k,i}A_k^{N_p-N_c}B_{k,m}\mathbf{u_m}(k+N_c-1) \\
&\quad + C_{k,i}A_k^{N_p-N_c}K_k
\end{aligned} \tag{2.18}
$$

Equation (2.17) and (2.18) can be reorganized as matrix form below.

$$\overrightarrow{\mathbf{y_i}} = F_i\mathbf{x}(k) + \Phi_{i,d}\overrightarrow{\mathbf{u_d}} + \Phi_{i,m}\overrightarrow{\mathbf{u_m}} + E_iK_k \tag{2.19}$$

where, $i = \{d, m\}$, $j = \{d, m\}$

$$\overrightarrow{\mathbf{y_i}} = [\mathbf{y_i}(k+1) \quad \mathbf{y_i}(k+2)\cdots\mathbf{y_i}(k+N_p)]^T$$

$$\overrightarrow{\mathbf{u_i}} = [\mathbf{u_i}(k) \quad \mathbf{u_i}(k+1)\cdots\mathbf{u_i}(k+N_c-1)]^T$$

$$\Phi_{i,j} = \begin{bmatrix} C_{k,i}B_{k,j} & 0 & \cdots & 0 \\ C_{k,i}A_kB_{k,j} & C_{k,i}B_{k,j} & \cdots & 0 \\ C_{k,i}A_k^2B_{k,j} & C_{k,i}A_kB_{k,j} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ C_{k,i}A_k^{N_p-1}B_{k,j} & C_{k,i}A_k^{N_p-2}B_{k,j} & \cdots & C_{k,i}A_k^{N_p-N_c}B_{k,j} \end{bmatrix}, F_i = \begin{bmatrix} C_{k,i}A_k \\ C_{k,i}A_k^2 \\ C_{k,i}A_k^3 \\ \vdots \\ C_{k,i}A_k^{N_p} \end{bmatrix}$$

$$E_i = \begin{bmatrix} C_{k,i} \\ C_{k,i}(I + A_k) \\ C_{k,i}(I + A_k + A_k^2) \\ \vdots \\ C_{k,i}(I + \sum_{j=1}^{N_p-N_c} A_k^j) \end{bmatrix}$$

2.1.3.2 Definition of Cost Function

Let tracking error of each player as $\mathbf{e_i}(k) = \mathbf{y_i}(k) - \mathbf{r_i}(k)$ with respect to its control reference $\mathbf{r_i}$, where, $i = d, m$. Then, cost function for each player can be determined in terms of the tracking error $\mathbf{e_i}$ and control inputs $\mathbf{u_i}$ as below.

$$J_i = \sum_{j=1}^{N_p} (\mathbf{e_i}(k+j)) Q_i (\mathbf{e_i}(k+j)) + \sum_{j=0}^{N_c-1} \mathbf{u_i}(k+j) R_i \mathbf{u_i}(k+j)$$
$$= \overrightarrow{\mathbf{e_i}}^T \overline{Q}_i \overrightarrow{\mathbf{e_i}} + \overrightarrow{\mathbf{u_i}}^T \overline{R}_i \overrightarrow{\mathbf{u_i}} \tag{2.20}$$

where,

$$\overrightarrow{\mathbf{e_i}} = \overrightarrow{\mathbf{y_i}} - \overrightarrow{\mathbf{r_i}} = \begin{bmatrix} \mathbf{y_i}(k+1) \\ \mathbf{y_i}(k+2) \\ \vdots \\ \mathbf{y_i}(k+N_p) \end{bmatrix} - \begin{bmatrix} \mathbf{r_i}(k+1) \\ \mathbf{r_i}(k+2) \\ \vdots \\ \mathbf{r_i}(k+N_p) \end{bmatrix} = \begin{bmatrix} \mathbf{e_i}(k+1) \\ \mathbf{e_i}(k+2) \\ \vdots \\ \mathbf{e_i}(k+N_p) \end{bmatrix}$$

$$\overline{Q}_i = \begin{bmatrix} Q_i & 0 & \cdots & 0 \\ 0 & Q_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_i \end{bmatrix}, \ \overline{R}_i = \begin{bmatrix} R_i & 0 & \cdots & 0 \\ 0 & R_i & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & R_i \end{bmatrix}$$

The cost function in Equation (2.20) can be rewritten in terms of control inputs $\overrightarrow{\mathbf{u_i}}$ which is variable to be optimized by substituting Equation (2.19).

$$\begin{aligned} J_i =& \frac{1}{2}(F_i \mathbf{x}(k) + \Phi_{i,i} \overrightarrow{\mathbf{u_i}} + \Phi_{i,-i} \overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}})^T \overline{Q}_i (F_i \mathbf{x}(k) + \Phi_{i,i} \overrightarrow{\mathbf{u_i}} + \Phi_{i,-i} \overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}}) \\ &+ \overrightarrow{\mathbf{u_i}}^T \overline{R}_i \overrightarrow{\mathbf{u_i}} \\ =& \frac{1}{2} \overrightarrow{\mathbf{u_i}}^T (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i) \overrightarrow{\mathbf{u_i}} + \overrightarrow{\mathbf{u_i}}^T \Phi_{i,i}^T \overline{Q}_i (\Phi_{i,-i} \overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}}) \\ &+ (\Phi_{i,-i} \overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}})^T \overline{Q}_i (\Phi_{i,-i} \overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}}) \end{aligned} \tag{2.21}$$

The notation $-i$ in Equation (2.21) means the other player. Because the third term in Equation

(2.21) does not depend on the optimized variable $\overrightarrow{\mathbf{u_i}}$, the cost function can be simplified as below in order to express a standard from of QP problem as $J = \frac{1}{2}\overrightarrow{\mathbf{u}}^T G \overrightarrow{\mathbf{u}} + \overrightarrow{\mathbf{u}}^T W$.

$$J_i = \frac{1}{2}\overrightarrow{\mathbf{u_i}}^T(\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)\overrightarrow{\mathbf{u_i}} + \overrightarrow{\mathbf{u_i}}^T \Phi_{i,i}^T \overline{Q}_i(F_i\mathbf{x}(k) + \Phi_{i,-i}\overrightarrow{\mathbf{u_{-i}}} + E_i K_k - \overrightarrow{\mathbf{r_i}}) \tag{2.22}$$

$$\begin{aligned}\triangle\overrightarrow{\mathbf{u_i}} &= \begin{bmatrix} \triangle\mathbf{u_i}(k) \\ \triangle\mathbf{u_i}(k+1) \\ \triangle\mathbf{u_i}(k+2) \\ \vdots \\ \triangle\mathbf{u_i}(k+N_c-1) \end{bmatrix} = \begin{bmatrix} I & 0 & 0 & \cdots & 0 & 0 \\ -I & I & 0 & \cdots & 0 & 0 \\ 0 & -I & I & \cdots & 0 & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & 0 & \cdots & -I & I \end{bmatrix} \begin{bmatrix} \mathbf{u_i}(k) \\ \mathbf{u_i}(k+1) \\ \mathbf{u_i}(k+2) \\ \vdots \\ \mathbf{u_i}(k+N_c-1) \end{bmatrix} - \begin{bmatrix} \mathbf{u_i}(k-1) \\ 0 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \\ &= \boldsymbol{\Theta}\overrightarrow{\mathbf{u_i}} - \overrightarrow{\mathbf{u_i}}(k-1)\end{aligned}$$

The constraints for $\overrightarrow{\mathbf{u}_i}$ and $\triangle\overrightarrow{\mathbf{u}_i}$ can be expressed in terms of $\overrightarrow{\mathbf{u}_i}$ as below.

$$\begin{aligned} -I\overrightarrow{\mathbf{u_i}} &\leq -U_{i,min} \\ I\overrightarrow{\mathbf{u_i}} &\leq U_{i,max} \\ -\triangle\overrightarrow{\mathbf{u_i}} = -\boldsymbol{\Theta}\overrightarrow{\mathbf{u_i}} + \overrightarrow{\mathbf{u_i}}(k-1) &\leq -\triangle U_{i,min} \\ \triangle\overrightarrow{\mathbf{u_i}} = \boldsymbol{\Theta}\overrightarrow{\mathbf{u_i}} - \overrightarrow{\mathbf{u_i}}(k-1) &\leq \triangle U_{i,max} \end{aligned} \tag{2.23}$$

where,

$$\begin{aligned} U_{i,max} &= [\mathbf{u_i}_{max} \quad \mathbf{u_i}_{max} \quad \cdots \quad \mathbf{u_i}_{max}]^T \\ U_{i,min} &= [\mathbf{u_i}_{min} \quad \mathbf{u_i}_{min} \quad \cdots \quad \mathbf{u_i}_{min}]^T \\ \triangle U_{i,max} &= [\triangle\mathbf{u_i}_{max} \quad \triangle\mathbf{u_i}_{max} \quad \cdots \quad \triangle\mathbf{u_i}_{max}]^T \\ \triangle U_{i,min} &= [\triangle\mathbf{u_i}_{min} \quad \triangle\mathbf{u_i}_{min} \quad \cdots \quad \triangle\mathbf{u_i}_{min}]^T \end{aligned}$$

The constraints in Equation (2.23) can be rewritten in terms of $\overrightarrow{\mathbf{u}_i}$ as Equation (2.24) and it can be reorganized as a matrix form in (2.25).

$$\begin{aligned} -I\overrightarrow{\mathbf{u_i}} &\leq -U_{i,min} \\ I\overrightarrow{\mathbf{u_i}} &\leq U_{i,max} \\ -\mathbf{\Theta}\overrightarrow{\mathbf{u_i}} &\leq -\triangle U_{i,min} - \overrightarrow{\mathbf{u_i}}(k-1) \\ \mathbf{\Theta}\overrightarrow{\mathbf{u_i}} &\leq \triangle U_{i,max} + \overrightarrow{\mathbf{u_i}}(k-1) \end{aligned} \tag{2.24}$$

$$A_{cons,i}\overrightarrow{\mathbf{u_i}} \leq B_{cons,i} \tag{2.25}$$

$$\text{where, } A_{cons,i} = \begin{bmatrix} -I \\ I \\ -\mathbf{\Theta} \\ \mathbf{\Theta} \end{bmatrix}, B_{cons,i} = \begin{bmatrix} -U_{i,min} \\ U_{i,max} \\ -\triangle U_{i,min} - \overrightarrow{\mathbf{u_i}}(k-1) \\ \triangle U_{i,max} + \overrightarrow{\mathbf{u_i}}(k-1) \end{bmatrix}$$

2.1.3.3 Formulation of Quadratic Programming

In this research, four different types of game are studied - non-cooperative game with simultaneous move, non-cooperative game with leader-and-follower, non-cooperative game with sequential move and cooperative game [60][61]. In Table 2.2, four different types of games are summarized. For each game type, quadratic programming problems are formulated using previously obtained matrix forms of MPC and cost functions.

Non-cooperative game with simultaneous move The non-cooperative game with simultaneous move is simply called as non-cooperative game. Each player focuses on his own performance index in order to maximize it under this game, and make a decision simultaneously. Each player

Table 2.2: Game Definitions

Non-cooperative game with simultaneous move	Each player knows the effect of the actions of the other and compensates the effect of the other's action by his optimal action. The players communicate with each other.
Non-cooperative game with leader-and-follower (Stackelberg game)	Similar to non-cooperative game with simultaneous move, but there exists a leader and a follower. The leader make a decision at first and then the follower make a decision with information about the leader's decision.
Non-cooperative game with sequential move	Similar to non-cooperative game with simultaneous move, but there exists sequence of players. Each player makes a decision at his turn and keep previous decision otherwise.
Cooperative game	The players share a common objective. The players communicate with each other.

recognizes the other player as a rational player. The quadratic programming problem of non-cooperative game with simultaneous move can be formulated as below.

$$\begin{aligned}
\overrightarrow{\mathbf{u_i}}^* &= \min_{\overrightarrow{\mathbf{u_i}}} J_i = \min_{\overrightarrow{\mathbf{u_i}}} \frac{1}{2}\overrightarrow{\mathbf{u_i}}^T(\Phi_{i,i}^T\overline{Q}_i\Phi_{i,i} + \overline{R}_i)\overrightarrow{\mathbf{u_i}} + \overrightarrow{\mathbf{u_i}}^T\Phi_{i,i}^T\overline{Q}_i(F_i\mathbf{x}(k) + \Phi_{i,-i}\overrightarrow{\mathbf{u_{-i}}}^* + E_iK_k - \overrightarrow{\mathbf{r_i}}) \qquad (2.26)\\
&\text{s.t. } A_{cons,i}\overrightarrow{\mathbf{u_i}} \leq B_{cons,i}\\
&\quad \overrightarrow{\mathbf{u_{-i}}}^* = \min_{\overrightarrow{\mathbf{u_{-i}}}} \frac{1}{2}\overrightarrow{\mathbf{u_{-i}}}^T(\Phi_{-i,-i}^T\overline{Q}_{-i}\Phi_{-i,-i} + \overline{R}_{-i})\overrightarrow{\mathbf{u_{-i}}} + \overrightarrow{\mathbf{u_{-i}}}^T\Phi_{-i,-i}^T\overline{Q}_{-i}(F_i\mathbf{x}(k) + \Phi_{-i,i}\overrightarrow{\mathbf{u_i}}^* + E_{-i}K_k - \overrightarrow{\mathbf{r_{-i}}})\\
&\quad\quad \text{s.t. } A_{cons,-i}\overrightarrow{\mathbf{u_{-i}}} \leq B_{cons,-i}
\end{aligned}$$

The notation $*$ of in $\overrightarrow{\mathbf{u_i}}^*$ and $\overrightarrow{\mathbf{u_{-i}}}^*$ in (2.26) means the optimal actions of players. In other words, the quadratic programming problem for each player is dependent on the other player's optimal action which is a solution of the other player's quadratic programming problem.

Non-cooperative game with leader-and-follower The non-cooperative game with leader-and-follower is also known as Stackelberg game. Leader plays first with knowledge about the follower will player as a rational player using observed leader's action. Firstly, the quadratic programming problem for a leader can be formulated as below.

$$\overrightarrow{\mathbf{u_l}}^* = \min_{\overrightarrow{\mathbf{u_l}}} J_l = \min_{\overrightarrow{\mathbf{u_l}}} \frac{1}{2}\overrightarrow{\mathbf{u_l}}^T(\Phi_{l,l}^T\overline{Q}_l\Phi_{l,l} + \overline{R}_l)\overrightarrow{\mathbf{u_l}} + \overrightarrow{\mathbf{u_l}}^T\Phi_{l,l}^T\overline{Q}_l(F_l\mathbf{x}(k) + \Phi_{l,f}\overrightarrow{\mathbf{u_f}}^* + E_lK_k - \overrightarrow{\mathbf{r_l}}) \tag{2.27}$$

s.t. $A_{cons,l}\overrightarrow{\mathbf{u_l}} \leq B_{cons,l}$

$$\overrightarrow{\mathbf{u_f}}^* = \min_{\overrightarrow{\mathbf{u_f}}} \frac{1}{2}\overrightarrow{\mathbf{u_f}}^T(\Phi_{f,f}^T\overline{Q}_f\Phi_{f,f} + \overline{R}_f)\overrightarrow{\mathbf{u_f}} + \overrightarrow{\mathbf{u_f}}^T\Phi_{f,f}^T\overline{Q}_f(F_f\mathbf{x}(k) + \Phi_{f,l}\overrightarrow{\mathbf{u_l}}^* + E_fK_k - \overrightarrow{\mathbf{r_f}})$$

s.t. $A_{cons,f}\overrightarrow{\mathbf{u_f}} \leq B_{cons,f}$

The notation l and f in (2.27) means "leader" and "follower" respectively. As shown in (2.27), the leader's quadratic programming problem is dependent on the follower's optimal action which is a solution of the other player's quadratic programming problem. On the one hand, follower produces his action with information about observed leader's action. That is, the follower's optimization problem is a standard quadratic programming problem. The quadratic programming problem for the follower can be formulated with given $\overrightarrow{\mathbf{u_l}}^*$ as below.

$$\overrightarrow{\mathbf{u_f}}^* = \min_{\overrightarrow{\mathbf{u_f}}} J_f = \min_{\overrightarrow{\mathbf{u_f}}} \frac{1}{2}\overrightarrow{\mathbf{u_f}}^T(F_f\mathbf{x}(k) + \Phi_{f,f}^T\overline{Q}_f\Phi_{f,f} + \overline{R}_f)\overrightarrow{\mathbf{u_f}} + \overrightarrow{\mathbf{u_f}}^T\Phi_{f,f}^T\overline{Q}_f(\Phi_{f,l}\overrightarrow{\mathbf{u_l}}^* + E_fK_k - \overrightarrow{\mathbf{r_f}}) \tag{2.28}$$

s.t. $A_{cons,f}\overrightarrow{\mathbf{u_f}} \leq B_{cons,f}$

Non-cooperative game with sequential move In non-cooperative game with sequential move, one player make a decision at his turn with information about the other players' previous decision and keep his decision otherwise as shown in (2.29).

$$\overrightarrow{\mathbf{u_i}}(k) = \begin{cases} \min_{\overrightarrow{\mathbf{u_i}}} J_i, & \text{if } k \text{ is } i\text{'s turn} \\ \overrightarrow{\mathbf{u_i}}(k-1), & \text{otherwise} \end{cases} \tag{2.29}$$

Therefore, players' optimization problem is a standard quadratic programming problem . The

quadratic programming problem can be formulated with given $\overrightarrow{\mathbf{u}_{-\mathbf{i}}}^*$ as below.

$$\overrightarrow{\mathbf{u_i}}^* = \min_{\overrightarrow{\mathbf{u_i}}} J_i = \min_{\overrightarrow{\mathbf{u_i}}} \frac{1}{2}\overrightarrow{\mathbf{u_i}}^T(\Phi_{i,i}^T\overline{Q}_i\Phi_{i,i} + \overline{R}_i)\overrightarrow{\mathbf{u_i}} + \overrightarrow{\mathbf{u_i}}^T\Phi_{i,i}^T\overline{Q}_i(F_i\mathbf{x}(k) + \Phi_{i,-i}\overrightarrow{\mathbf{u_{-i}}}^* + E_iK_k - \overrightarrow{\mathbf{r_i}}) \tag{2.30}$$
$$\text{s.t. } A_{cons,i}\overrightarrow{\mathbf{u_i}} \leq B_{cons,i}$$

Cooperative game In this game, players share a common objective which is defined as weighted summation of each cost function as follows.

$$J = \sum_{j=d,m} \rho_j J_j \tag{2.31}$$

The other player's control action is neglected in the cost function of one player because each player cannot affect and adjust the other player's control action directly. Therefore, the cost function for each player can be expressed as

$$\begin{aligned}
\overrightarrow{\mathbf{u_i}}^* &= \min_{\overrightarrow{\mathbf{u_i}}} \sum_{j=d,m} \rho_j J_j \\
&= \sum_{j=1}^{N_p} \begin{bmatrix} \mathbf{e_d}(k+j) \\ \mathbf{e_m}(k+j) \end{bmatrix}^T \begin{bmatrix} \rho_d Q_d & 0 \\ 0 & \rho_m Q_m \end{bmatrix} \begin{bmatrix} \mathbf{e_d}(k+j) \\ \mathbf{e_m}(k+j) \end{bmatrix} \\
&\quad + \sum_{j=1}^{N_c-1} \overrightarrow{\mathbf{u_i}}(k+j)(\rho_i R_i)\overrightarrow{\mathbf{u_i}}(k+j) \\
&= \sum_{j=1}^{N_p} \mathbf{e_{co}}(k+j)^T Q_{co}\mathbf{e_{co}}(k+j) + \sum_{j=1}^{N_c-1} \overrightarrow{\mathbf{u_i}}(k+j)^T R_{i,co}\overrightarrow{\mathbf{u_i}}(k+j) \\
&= \overrightarrow{\mathbf{e_{co}}}^T\overline{Q}_{co}\overrightarrow{\mathbf{e_{co}}} + \overrightarrow{\mathbf{u_i}}^T\overline{R}_{i,co}\overrightarrow{\mathbf{u_i}}
\end{aligned} \tag{2.32}$$

where, $i = \{d, m\}$

$$\mathbf{e_{co}}(k) = \begin{bmatrix} \mathbf{e_d}(k) \\ \mathbf{e_m}(k) \end{bmatrix},\ Q_{co} = \begin{bmatrix} \rho_d Q_h & 0 \\ 0 & \rho_a Q_a \end{bmatrix},\ R_{i,co} = \rho_i R_i$$

$$\overrightarrow{\mathbf{e_{co}}} = \begin{bmatrix} \mathbf{e_{co}}(k+1) \\ \mathbf{e_{co}}(k+2) \\ \vdots \\ \mathbf{e_{co}}(k+N_p) \end{bmatrix},\ \overline{Q}_{co} = \begin{bmatrix} Q_{co} & 0 & \cdots & 0 \\ 0 & Q_{co} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & Q_{co} \end{bmatrix}$$

The cost function for cooperative game in Equation (2.32) can be converted to standard non-cooperative game with simultaneous move in regards to the augmented system described below.

$$\begin{bmatrix} \mathbf{x}(k+1) \\ \mathbf{x}(k+1) \end{bmatrix} = \begin{bmatrix} A_k & 0 \\ 0 & A_k \end{bmatrix} \begin{bmatrix} \mathbf{x}(k) \\ \mathbf{x}(k) \end{bmatrix} + \begin{bmatrix} B_{k,d} \\ B_{k,d} \end{bmatrix} \mathbf{u_d}(k) + \begin{bmatrix} B_{k,m} \\ B_{k,m} \end{bmatrix} \mathbf{u_m}(k) + \begin{bmatrix} K_k \\ K_k \end{bmatrix} \tag{2.33}$$

2.1.4 Solution of Game Theoretic Model Predictive Control

2.1.4.1 Non-cooperative game with simultaneous move

Game theoretic MPC for non-cooperative game with simultaneous move is formulated as (2.26). Each player's cost function includes the other player's action. Without the constraints, the solution can be obtained as

$$\overrightarrow{\mathbf{u_i}}^* = K_{r,i}\overrightarrow{\mathbf{r_i}} - K_{x,i}\mathbf{x}(k) - K_{u,i}\overrightarrow{\mathbf{u_{-i}}}^* - K_{k,i} \tag{2.34}$$

where

$$K_{r,i} = (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i$$
$$K_{x,i} = (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i F_i$$
$$K_{u,i} = (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i \Phi_{i,-i}$$
$$K_{k,i} = (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i E_i K_k$$

From (2.34), the unconstrained solution can be gotten as

$$\begin{bmatrix} \overrightarrow{\mathbf{u_d}}^* \\ \overrightarrow{\mathbf{u_m}}^* \end{bmatrix} = \begin{bmatrix} I & K_{u,d} \\ K_{u,m} & I \end{bmatrix}^{-1} \begin{bmatrix} T_d & 0 \\ 0 & T_m \end{bmatrix} \begin{bmatrix} \mathbf{z_d} \\ \mathbf{z_m} \end{bmatrix} + \begin{bmatrix} I & K_{u,d} \\ K_{u,m} & I \end{bmatrix}^{-1} \begin{bmatrix} -K_{k,d} \\ -K_{k,m} \end{bmatrix} \tag{2.35}$$

where,

$$T_d = \begin{bmatrix} -K_{x,d} & K_{r,d} \end{bmatrix},\ T_m = \begin{bmatrix} -K_{x,m} & K_{r,m} \end{bmatrix}$$
$$\mathbf{z_d} = \begin{bmatrix} \mathbf{x}(k) & \overrightarrow{\mathbf{r_d}} \end{bmatrix}^T,\ \mathbf{z_m} = \begin{bmatrix} \mathbf{x}(k) & \overrightarrow{\mathbf{r_m}} \end{bmatrix}^T$$

The first element of the optimized $\overrightarrow{\mathbf{u_d}}^*$ and $\overrightarrow{\mathbf{u_m}}^*$ is chosen as control input by the receding horizon concept.

$$\mathbf{u_d}(k) = [I \quad 0 \quad 0 \quad \cdots \quad 0]\overrightarrow{\mathbf{u_d}}^* \tag{2.36}$$
$$\mathbf{u_m}(k) = [I \quad 0 \quad 0 \quad \cdots \quad 0]\overrightarrow{\mathbf{u_m}}^* \tag{2.37}$$

To consider the constraints in the quadratic programming problem, the iterative approach in

[60] based on convexity of the cost functions can be used. The iterative algorithms is described in Algorithm 2.

Algorithm 2 Iterative Nash solution - convex step [60]

Iteration step i = 0
Initialize $\overrightarrow{\mathbf{u_d}}^i$ and $\overrightarrow{\mathbf{u_m}}^i$
Define ω_1 and ω_2 satisfying $\omega_1 + \omega_2 = 1$.
while $\mid \triangle \mid \leq \epsilon$ **do**
  Solve the quadratic programming problems, (21) and (22) for each player using $\overrightarrow{\mathbf{u_d}}^i$ and $\overrightarrow{\mathbf{u_m}}^i$ and get the solutions as $\overrightarrow{\mathbf{u_d}}^0$ and $\overrightarrow{\mathbf{u_m}}^0$

  Calculate $\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}$ as
    $(\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}) = \omega_1(\overrightarrow{\mathbf{u_d}}^0, \overrightarrow{\mathbf{u_m}}^i) + \omega_2(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^0)$

  Calculate $\triangle_d = J_d(\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}) - J_d(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)$
    $\triangle_m = J_m(\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}) - J_m(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)$
    $\triangle = \max(\triangle_d, \triangle_m)$
  Update $i \leftarrow i + 1$
end while

Because the cost decreases on iteration, $J_d(\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}) \leq J_d(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)$, the iterative solution converges [60]. The convergence of the iterative solution can be proved in (2.38).

With $\omega_1 + \omega_2 = 1$ and $p = \{d, m\}$,

$$\begin{aligned} J_p(\overrightarrow{\mathbf{u_d}}^{i+1}, \overrightarrow{\mathbf{u_m}}^{i+1}) &= J_p(\omega_1(\overrightarrow{\mathbf{u_d}}^0, \overrightarrow{\mathbf{u_m}}^i) + \omega_2(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^0)) \\ &\leq J_p(\omega_1(\overrightarrow{\mathbf{u_d}}^0, \overrightarrow{\mathbf{u_m}}^i)) + J_p(\omega_2(\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^0)) \\ &\leq \omega_1 J_p((\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)) + J_p(\omega_2 \overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)) \\ &\leq \omega_1 J_p((\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)) + \omega_2 J_p((\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)) \\ &= J_p((\overrightarrow{\mathbf{u_d}}^i, \overrightarrow{\mathbf{u_m}}^i)) \end{aligned} \tag{2.38}$$

2.1.4.2 Non-cooperative game with leader-and-follower

In this game, the leader knows how the follower reacts to his action. Therefore, the leader uses the expected reaction of the follower to make his decision. On the other hand, the follower makes a decision with given leader's decision, and quadratic programming problem of the follower is a standard form of conventional MPC problem. Without the constraints, leader's solution can be obtained as

$$\overrightarrow{\mathbf{u_l}}^* = K_{r,l}^{'}\overrightarrow{\mathbf{r_l}} - K_{x,l}^{'}\mathbf{x}(k) - K_{r,f}^{'}\overrightarrow{\mathbf{r_f}} - K_{k,l}^{'} \tag{2.39}$$

where,

$$\begin{aligned}
K_{r,l}^{'} &= ({\Phi_{l,l}^{'}}^T\overline{Q}_l\Phi_{l,l}^{'} + \overline{R}_l)^{-1}{\Phi_{l,l}^{'}}^T\overline{Q}_l \\
K_{x,l}^{'} &= ({\Phi_{l,l}^{'}}^T\overline{Q}_l\Phi_{l,l}^{'} + \overline{R}_l)^{-1}{\Phi_{l,l}^{'}}^T\overline{Q}_l F_l^{'} \\
K_{r,f}^{'} &= ({\Phi_{l,l}^{'}}^T\overline{Q}_l\Phi_{l,l}^{'} + \overline{R}_l)^{-1}{\Phi_{l,l}^{'}}^T\overline{Q}_l\Phi_{l,f}^{'} \\
K_{k,l}^{'} &= K_{k,l} - \Phi_{l,f}K_{k,f} \\
F_l^{'} &= F_l - \Phi_{l,f}F_f \\
\Phi_{l,l}^{'} &= \Phi_{l,l} - \Phi_{l,f}\Phi_{l,f} \\
\Phi_{l,f}^{'} &= \Phi_{l,f}K_{r,f}
\end{aligned}$$

Follower's solution for non-constraint problem can be derived as below.

$$\overrightarrow{\mathbf{u_f}}^* = K_{r,f}\overrightarrow{\mathbf{r_f}} - K_{x,f}\mathbf{x}(k) - K_{u,f}\overrightarrow{\mathbf{u_l}}^* - K_{k,f} \tag{2.40}$$

where,

$$K_{r,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f$$
$$K_{x,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f F_f$$
$$K_{u,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f \Phi_{f,l}$$
$$K_{k,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f E_f K_k$$

Because constraints of leader are dependent on the follower's control action, iterative approach is used for leader's solution to consider the constraints in the quadratic programming problem similar to non-cooperative simultaneous move. Meanwhile, the follower's solution with constraints can be obtained by existing algorithms such as active set methods or Hildreth's quadratic programming procedures [62] for standard constraint MPC problem. The solution for non-constraint problem can be derived as below.

$$\overrightarrow{\mathbf{u_f}}^* = K_{r,f} \overrightarrow{\mathbf{r_f}} - K_{x,f} \mathbf{x}(k) - K_{u,f} \overrightarrow{\mathbf{u_l}}^* - K_{k,f} \tag{2.41}$$

where,

$$K_{r,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f$$
$$K_{x,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f F_f$$
$$K_{u,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f \Phi_{f,l}$$
$$K_{k,f} = (\Phi_{f,f}^T \overline{Q}_f \Phi_{f,f} + \overline{R}_f)^{-1} \Phi_{f,f}^T \overline{Q}_f E_f K_k$$

2.1.4.3 Non-cooperative game with sequential move

In this game, players make their decision sequentially. In other words, they make new decision at their turn and keep previous decision otherwise. Therefore, their quadratic programming problems are same as a standard form of conventional MPC problem with given the other's decision

similar to follower's MPC problem in Stackelberg game. The solution for non-constraint problem can be derived as below.

$$\overrightarrow{\mathbf{u_i}}^* = K_{r,i}\overrightarrow{\mathbf{r_i}} - K_{x,i}\mathbf{x}(k) - K_{u,i}\overrightarrow{\mathbf{u_i}}^* - K_{k,i} \tag{2.42}$$

where, $i = \{d, m\}$

$$\begin{aligned}
K_{r,i} &= (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i \\
K_{x,i} &= (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i F_i \\
K_{u,i} &= (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i \Phi_{i,-i} \\
K_{k,i} &= (\Phi_{i,i}^T \overline{Q}_i \Phi_{i,i} + \overline{R}_i)^{-1} \Phi_{i,i}^T \overline{Q}_i E_i K_k
\end{aligned}$$

The solution with constraints can be obtained as standard constraint MPC solution similar to aforementioned follower's solution of Stackelberg game.

2.1.4.4 Cooperative game

In Section 2.1.3.3, it is shown that cooperative game is converted to non-cooperative game with simultaneous move with respect to an augmented system described in (2.33). Therefore solution for this game can be obtained in the same way of non-cooperative game with simultaneous move for the augmented system.

2.2 Shared Control Strategy

Human driver and machine use information about collision safety and tracking error to make a decision. The collision with surrounding vehicles or objects should be avoided, and the two vehicle controllers, human driver and machine, figure out how the vehicle maneuvers is dangerous according to relative maneuver with respect to surrounding vehicles. Zhang et al.[63] suggested a three warning levels based on time-to-collision(TTC) for forward collision warning system as follows.

- 1.5 sec $\leq$ TTC $<$ 2.5 sec : Cautionary warning (visual signal)
- 0.5 sec $\leq$ TTC $<$ 1.5 sec : Imminent warning (visual signal+auditory signals)
- TTC $<$ 0.5 sec : Overriding (automatic braking)

TTC can be obtained using relative speed and distance from the front vehicle as follows.

$$TTC(t) = \frac{D(t)}{V_D(t)} \tag{2.43}$$

where $D(t)$ is the relative distance and V_D is the relative speed from the front vehicle. Chen et al.[64] defined the collision probability index(CPI) curve based on the three warning levels using a Z-shaped membership function as follows.

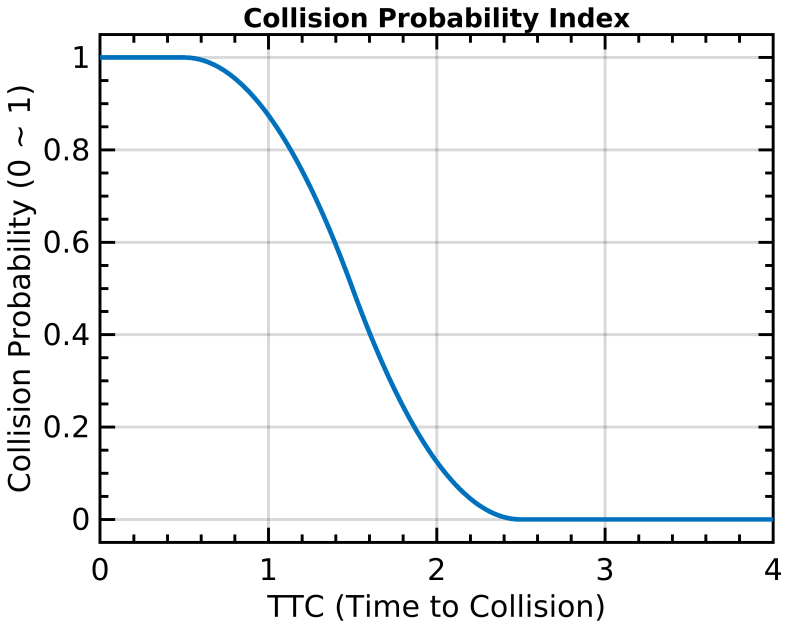


Figure 2.3: Collision probability index(CPI)

Also, tracking errors is the other important information for human driver and machine to represent how the vehicle follows their designated references. The weight sum of tracking error for each player can be defined as follows, where $i = \{d, m\}$

$$\mathbf{e_{trck,i}}(k) = \sqrt{\sum_j w_j(\mathbf{y_i} - \mathbf{r_i})} \tag{2.44}$$

The information of collision probability in Figure 2.3 and tracking error in (2.44) are used in our shared control strategy.

2.2.1 Shared control strategy for machine

It can be understood that tracking error represents how a vehicle follows control reference. Accordingly, machine will basically increase its control action as its tracking error increases. But machine also needs to consider human driver's intention and to react appropriately to human driver's intention. In this research, the human driver's intention is not directly estimated but the driver's intention is indirectly taken into account. Because the conflict by human driver with a different intention leads to the machine's high tracking error, the machine can figure out whether the human driver has different intention or not based on machine's tracking error. Therefore, the machine's tracking error is used as information to judge the human driver's intention in this research. It is desirable that machine needs to reduce its control action in a non-dangerous situation even though its tracking error is large, because the human driver has a different intention. In other words, the machine is able to follow its own decision or to follow human driver's decision considering its tracking error and collision probability. Based on this idea, shared control strategy for machine can be defined as below.

$$\frac{d}{dt}\rho_m = E \cdot (1 - \text{CPI}) + E \cdot \text{CPI} \tag{2.45}$$

where,

E: tracking error gain from weighted sum of tracking errors

CPI: collision probability index

The meaning of (2.45) is that machine increases its control action to take more control authority according to its tracking error in safety critical conditions while machine decreases its control action to reduce more control authority in non-safety critical conditions where there is not any collision risk. In addition, machine needs to be capable of dealing with situations where a human driver does not have an intention to control a vehicle. In these situations, machine has to increase its control authority to control a vehicle by itself, regardless of the risk of collision. To handle these situations, control activity function is introduced as shown in Equation (2.46) and Figure 2.4.

$$A = 1 - \exp^{-\left(k11\left|\frac{u_{d,1}}{u_{d,1max}}\right|\right)^{k12}\left(k21\left|\frac{u_{d,2}}{u_{d,2max}}\right|\right)^{k22}} \tag{2.46}$$

Using control activity in (2.46), finally shared control strategy for machine is proposed as below.

$$\frac{d}{dt}\rho_m = S \cdot (1 - \text{CPI}) + E \cdot \text{CPI} \tag{2.47}$$

where,

$$S = \begin{cases} -E, & \text{if } A > 0 \\ E, & \text{otherwise} \end{cases}$$

Figure 2.5 illustrates the proposed shared control strategy for machine.

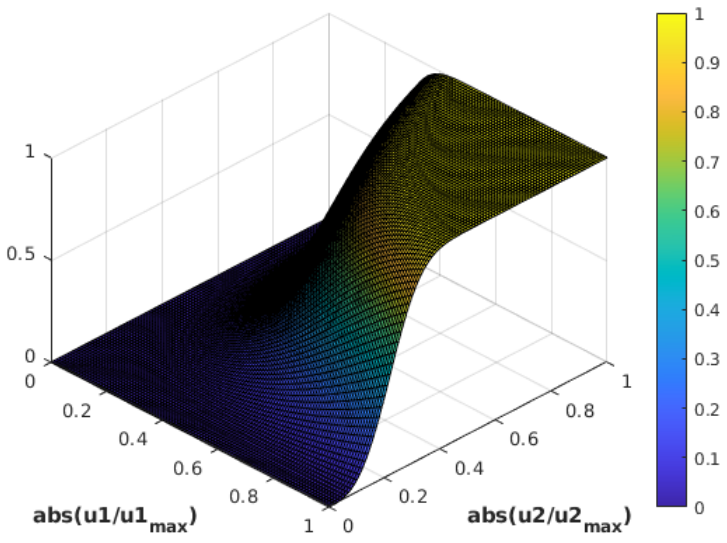


Figure 2.4: Control activity

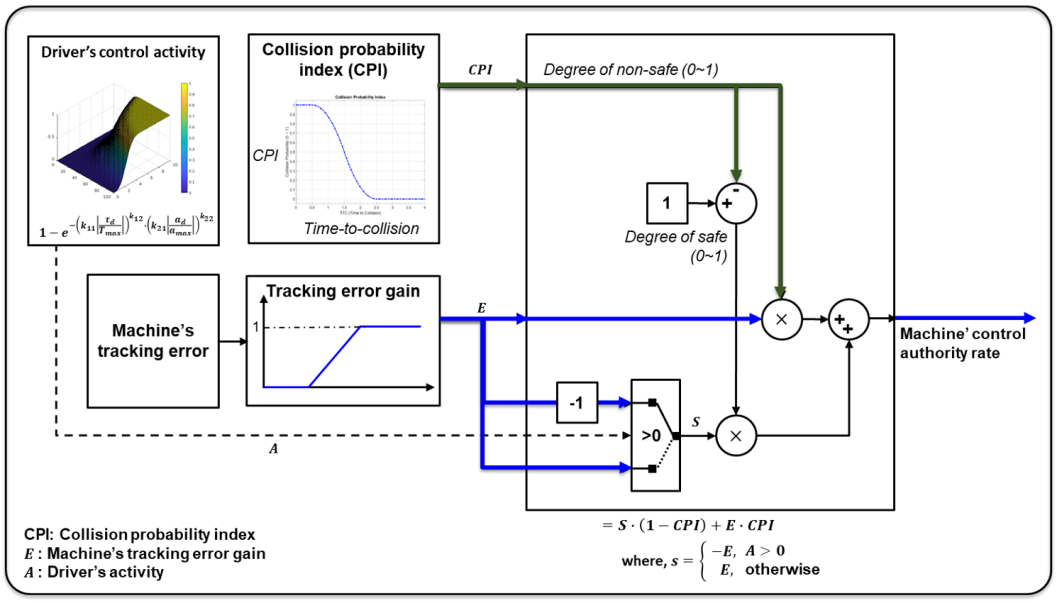


Figure 2.5: Shared control strategy of machine

In regards to non-safety critical condition where collision risk is low, Figure 2.6 shows the path for reducing machine's control authority, when a human driver has an intention to control a vehicle. On the other hand, Figure 2.7 illustrates the path for increasing machine's control authority in non-safety critical condition, when a human driver has an intention to control a vehicle.

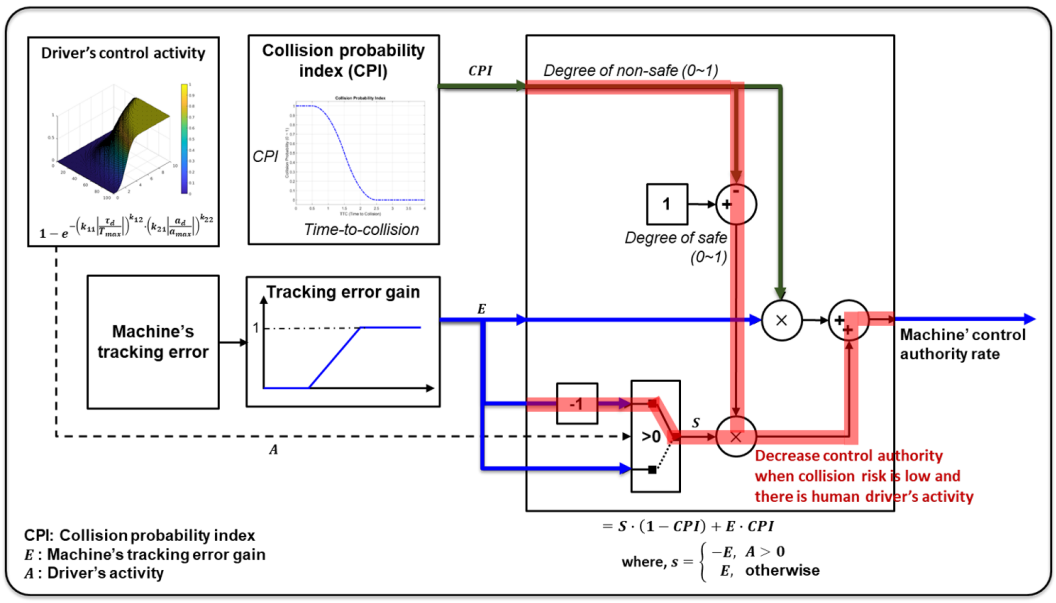


Figure 2.6: Shared control strategy of machine with human driver's intention - non-safety critical

Similarly, Figure 2.8 illustrates the path for increasing machine's control authority in regards to safety critical condition where collision risk is high.

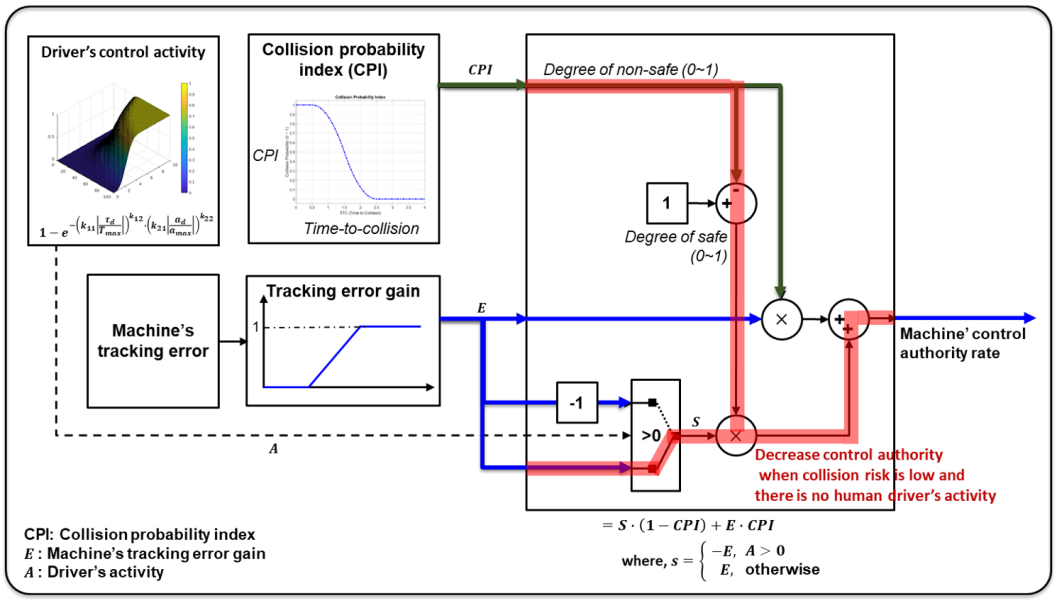


Figure 2.7: Shared control strategy of machine without human driver's intention - non-safety critical

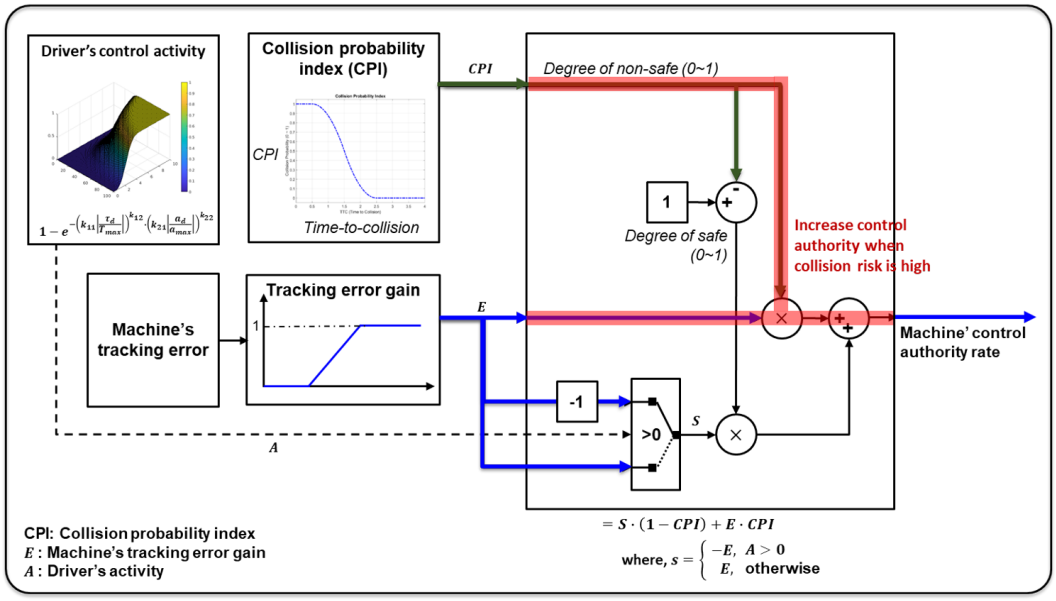


Figure 2.8: Shared control strategy of machine - safety critical

2.2.2 Shared control strategy for human

Similar to machine's shared control strategy, it can be understood that human driver uses information about his tracking error as a measure to figure out how a vehicle maneuver corresponds to his demand. Hence, human driver will basically increase his control action as his tracking error increases. But human drier's reaction to machine's control action is dependent on human driver's characteristics. In other words, some human drivers tend to follow his own decision and some human drivers tend to follow machine's decision, when human driver's tracking error is high. Their reaction can be varying according to their intention to follow his own or machine's decision. With consideration of this idea, human driver's shared strategy is formulated as below.

$$\frac{d}{dt}\rho_d = E \cdot (1 - \mathrm{I}) + E \cdot \mathrm{I} \tag{2.48}$$

where,

E: tracking error gain from weighted sum of tracking errors

I: human driver's intention to follow his own or machine's decision

It can be understood that human driver is cooperative with machine if value of intention term, I, in (2.48) is 0. In other words, human driver acts as a cooperative driver to follow machine's decision when human driver's tracking error is high. If value of intention term, I, is 1, human driver will be non-cooperative and follow to his own decision by increasing his control authority when his tracking error is high. Figure 2.9 shows the suggested shared control strategy for human driver. To be specific, Figure 2.10 shows the path for reducing human driver's control authority when human driver does not have an intention to control a vehicle. Similarly, Figure 2.11 illustrates the path for increasing human driver's control authority in the condition that human driver has an intention to control a vehicle by himself.

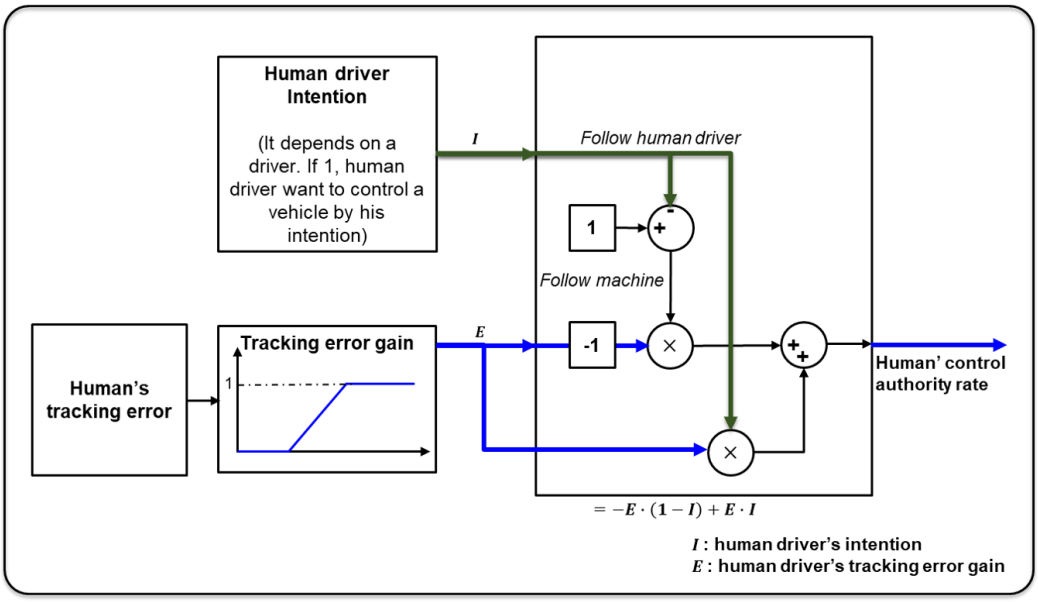


Figure 2.9: Shared control strategy of human driver

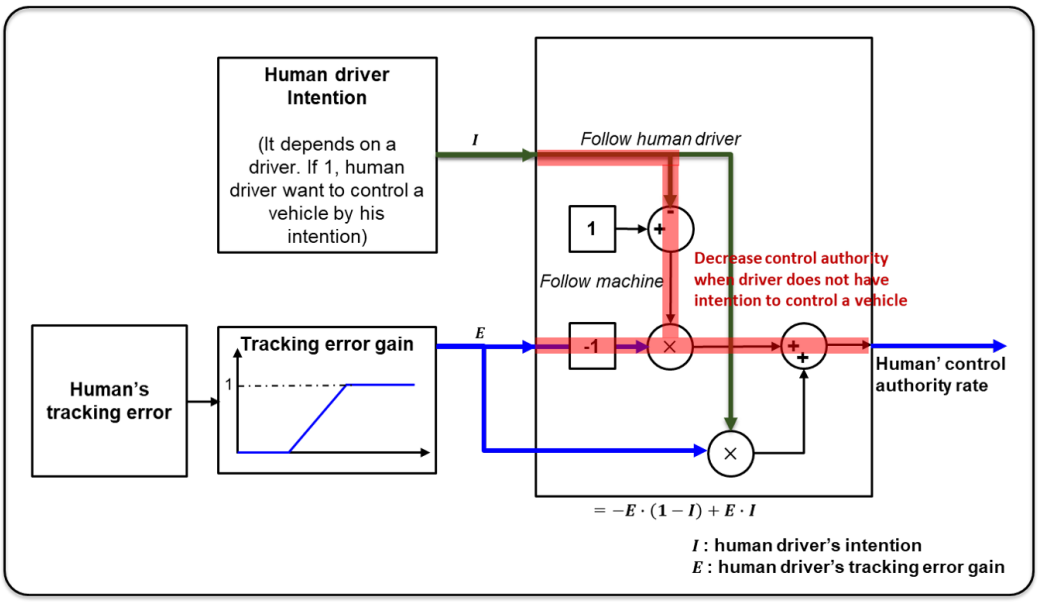


Figure 2.10: Shared control strategy of human driver - non-safety critical

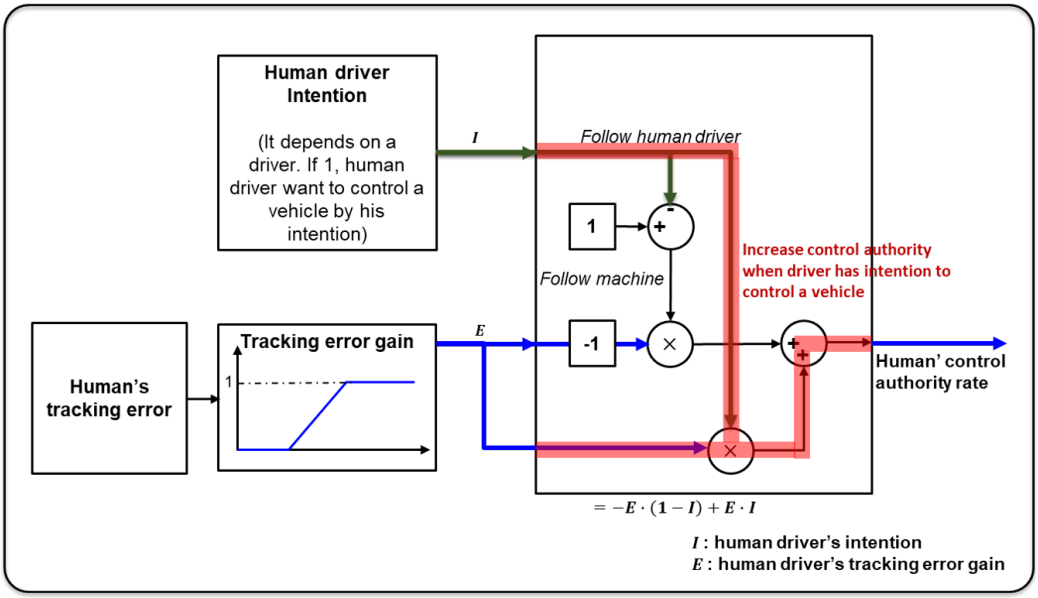


Figure 2.11: Shared control strategy of human driver - non-safety critical

2.2.3 Game transition

In real driving situations, human driver and the machine may establish and change their game according to driving situations. When a human driver starts driving, the human driver and the machine recognize the existence of the other player and know both players are well-performing vehicle controllers. Therefore, it is reasonable that the two players, human driver and machine, start their game under cooperative game framework. Unless the human driver cooperates with the other, the game is maintained as cooperative game. But in some cases, some drivers might intend to compete with machine, and then the game is converted from cooperative to noncooperative if machine also want to control a vehicle by itself. If collision probability is high in noncooperative game, machine needs to increase its control authority and then eventually has to take all control authority to avoid a collision. In this situation, the vehicle is operated in fully autonomous driving mode. The driving mode goes back to cooperative driving from fully autonomous driving mode, after the dangerous driving situation is gone. Based on this principle, finite state machine (FSM) based game transition model is developed as illustrated in Figure 2.12. In game transition conditions, new term 'degree of conflict' is used to represent how machine's action conflicts with human

driver's action. The 'degree of conflict' is defined by multiplication of machine's tracking error and machine's control authority. That means the variable is high when machine increases more control authority but vehicle maneuver still does not corresponds to its action, and it is low otherwise. To remove human driver's control action, specific mechanism such as electric clutch or steer-by-wire needs to be equipped in real applications.

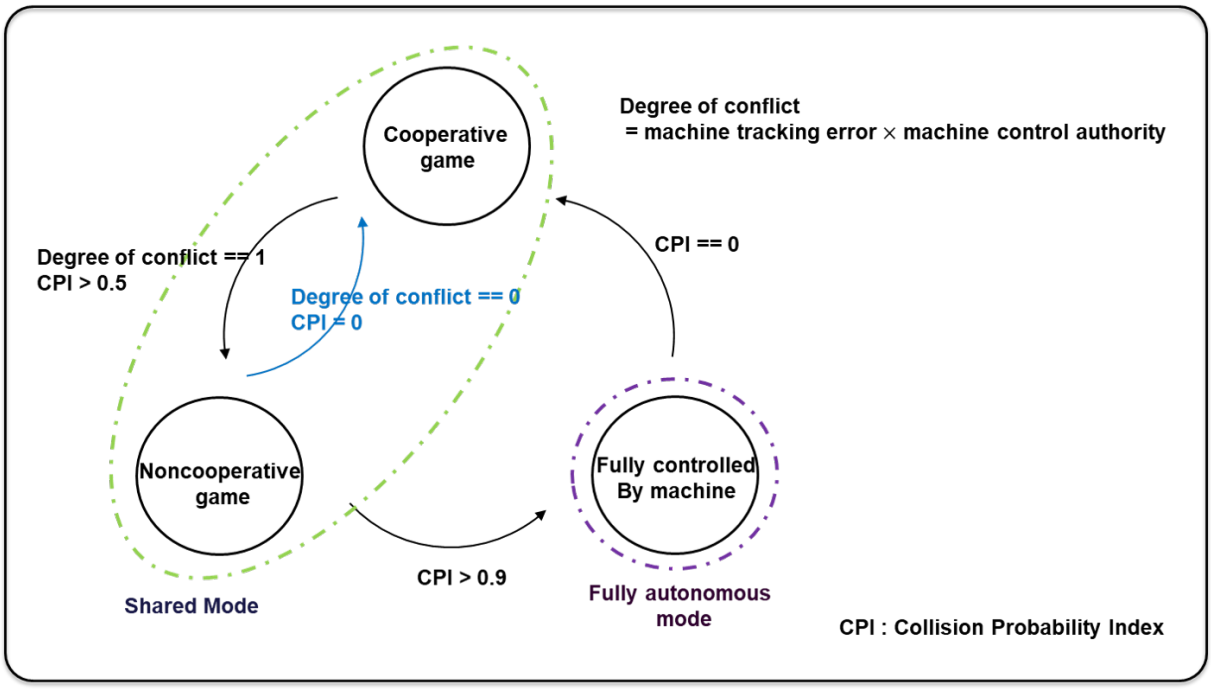


Figure 2.12: Game Transition

2.3 Simulation studies

In this section, simulation results are investigated. First of all, lane change scenario is studied under fixed game framework. Simulation results are compared according to aforementioned four different types of game. Also, lane change scenario with game transition is analyzed to understand how human driver and machine change their control intention (authority) in accordance with driving situations. In addition, more complicated driving scenario is studied. In the scenario, lane change is required to avoid a collision from a slower front vehicle but the other vehicle also makes

a lane change to the same target lane of ego-vehicle. In this situation, game transition and shared control authority are investigated.

2.3.1 Lane change scenario under fixed game

In this scenario, lane change is required to avoid a collision from a slower front vehicle, but it is assumed that human driver does not recognize the front vehicle and keeps driving on current lane. Also, it is supposed that human driver and machine do not change their game. Actually, human driver and machine may change their game in real driving situations but simulations were performed under fixed game framework to investigate simulation results according to specific game type. Figure 2.13 briefly shows the simulation scenario.

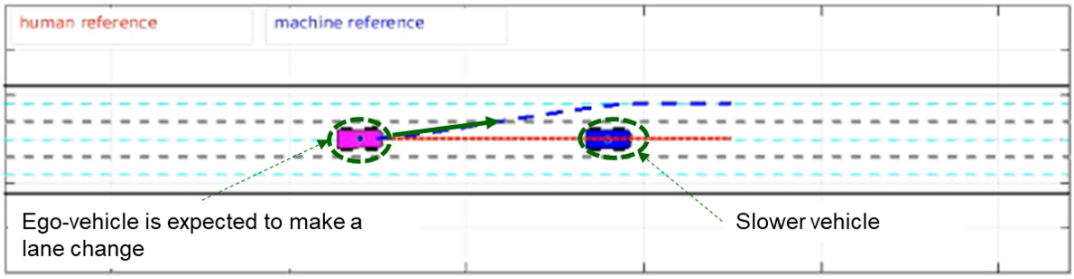
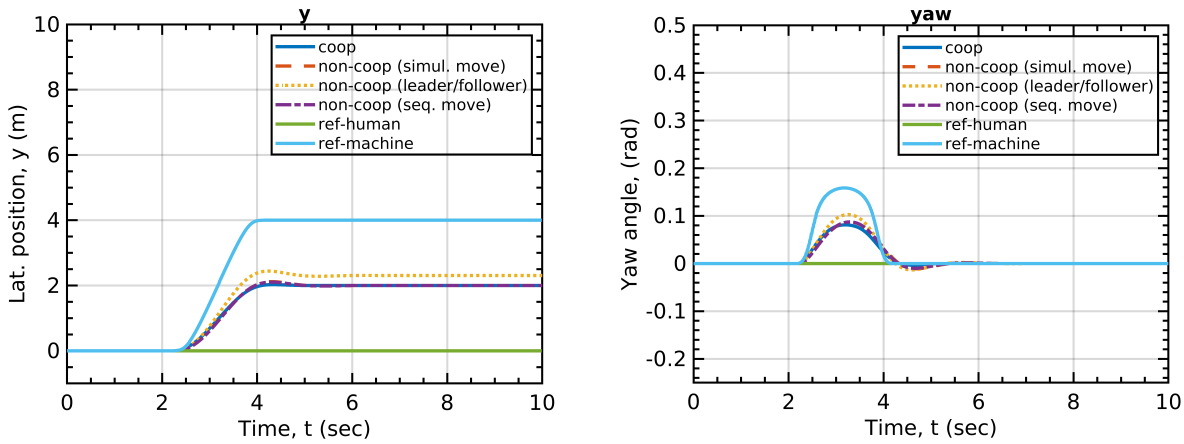


Figure 2.13: Simulation scenario - Lane change under fixed game

In regards to non-cooperative game with leader-and-follower, it is assumed that human driver plays as a leader and machine is a follower. This simulation scenario focuses more on lateral maneuver than longitudinal maneuver. The simulation results are illustrated in Figure 2.14, 2.15 and 2.16. Under fixed non-cooperative game with simultaneous move and fixed non-cooperative game with sequential move, moving trajectory of ego-vehicle converges to the middle between two players' references. Also, ego-vehicle goes to the middle of two player's references under fixed cooperative game framework. One the other hands, moving trajectory of ego-vehicle leans toward follower's reference. It can be interpreted that a leader make decision to maximize his payoff with knowledge that a follower also make a decision to maximize his payoff using given leader's

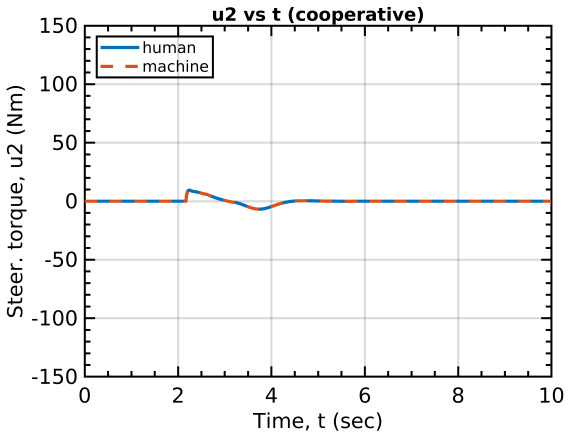
action. Therefore, a leader takes an action which give the best payoff among the best reactions of a follower with respect to leader's action, and it makes the trajectory of ego-vehicle is closer to follower's reference. In regards to control actions, human driver and machine exert control actions in the opposite direction in non-cooperative games but the two players produce similar control actions in cooperative game due to shared objective. In all game frameworks, ego-vehicle collides with the front vehicle because ego-vehicle drives along the middle of two players' references or slightly lean to reference of the follower(machine).



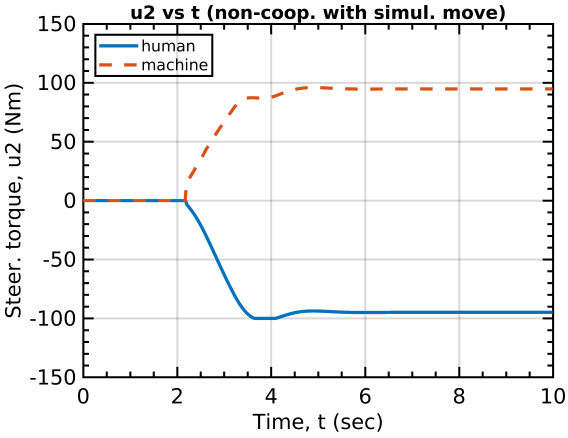
(a) lateral position

(b) yaw angle

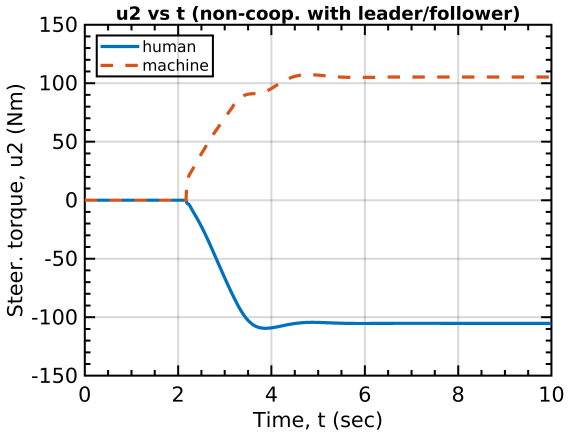
Figure 2.14: Vehicle trajectory - lane change under fixed game framework



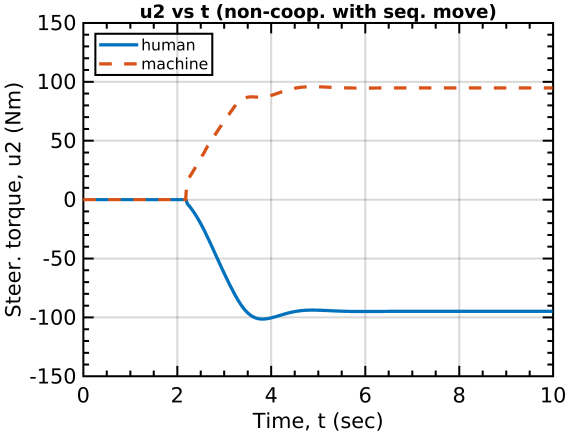
(a) Cooperative game



(b) Non-cooperative game with simultaneous move

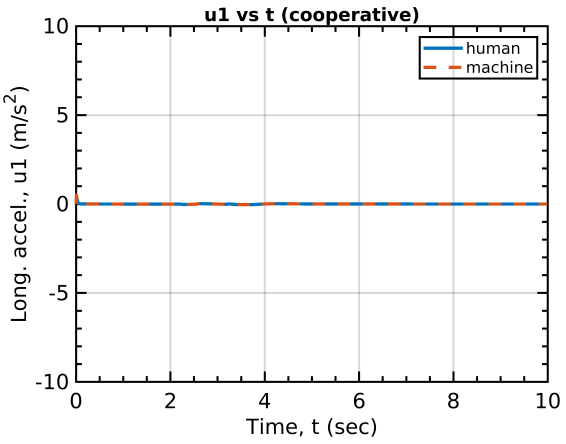


(c) Non-cooperative game with leader/follower

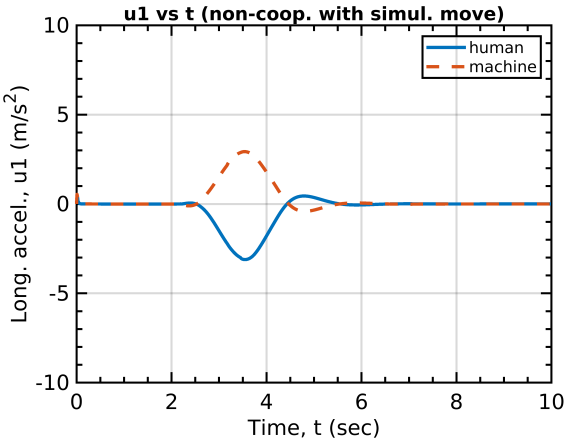


(d) Non-cooperative game with sequential move

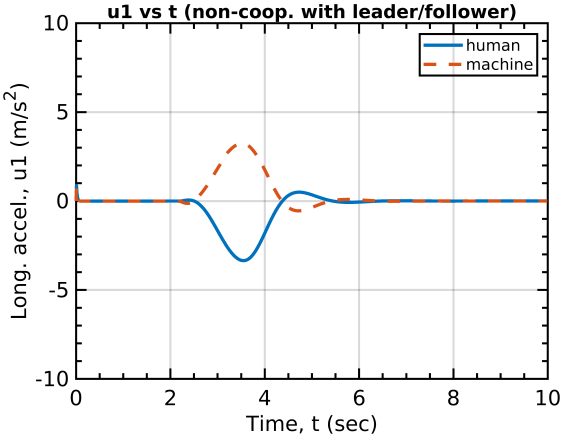
Figure 2.15: Steering torque input - lane change under fixed game framework



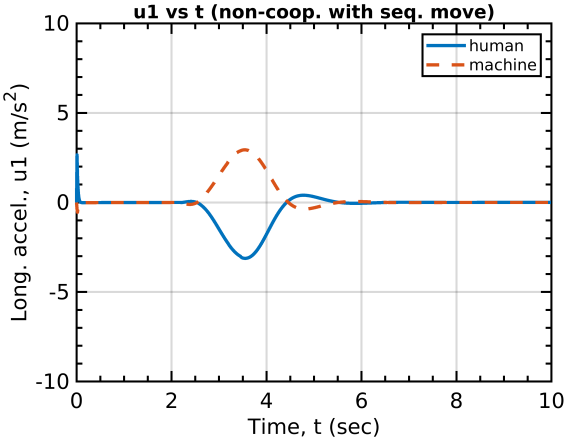
(a) Cooperative game



(b) Non-cooperative game with simultaneous move



(c) Non-cooperative game with leader/follower



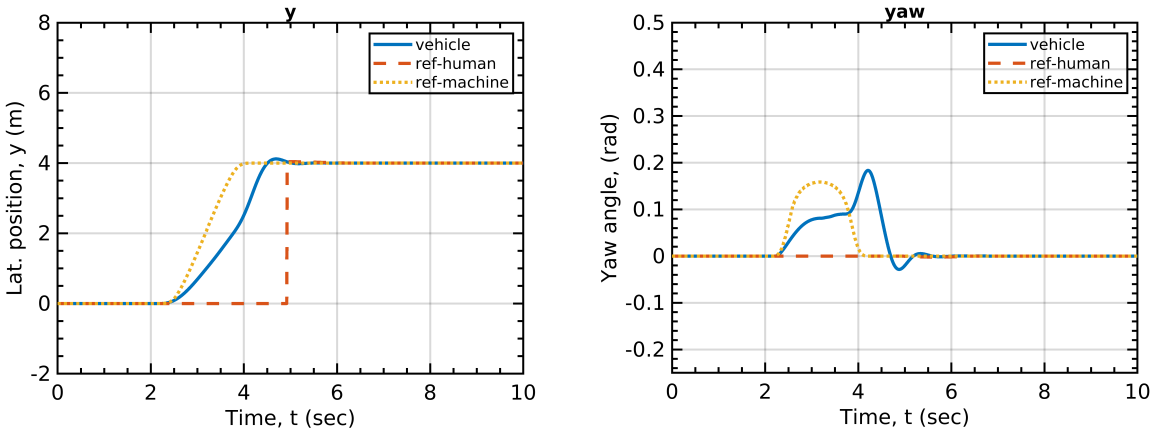
(d) Non-cooperative game with sequential move

Figure 2.16: Long. acceleration input - lane change under fixed game framework

2.3.2 Lane change scenario with game transition

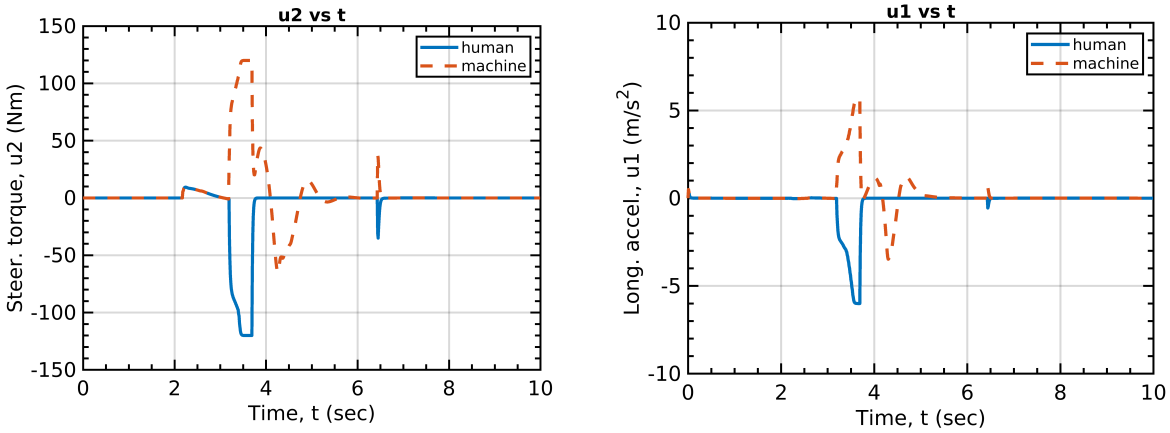
This scenario is similar to previous scenario under fixed game framework illustrated in Figure 2.13 but a game type can be varying according to driving situations. It can be more reasonable in real driving situations to change a game type during driving. Similarly, it is assumed that human driver does not recognize the front vehicle and keeps driving on current lane, and additionally it is supposed that human driver and machine player play non-cooperative game with sequential move among aforementioned three different types of non-cooperative games, when they play non-cooperatively. Meanwhile, human driver can figure out correspondence between his decision and machine's decision through his tracking error, because tracking error can be considered as a degree of representing how vehicle maneuver follows to his decision. In the condition that human driver's tracking error is high, human driver can judge that machine wants to control a vehicle with different control reference from human driver's. In this condition, human diver may follow to machine's decision or control a vehicle by his own decision. Therefore, the simulation results are dependent on human driver's intention to follow his own decision or to follow machine's decision when there exists conflict between human driver's and machine's demands. Figure 2.17, 2.18 and 2.19 show the simulation results with a human driver who has an intention to follow his own demand in the conflict condition. In contrast to results under fixed game framework, ego-vehicle avoids a collision from the front vehicle because machine can take all control authority from human driver when collision risk is high. Firstly, the game starts as cooperative game and then game is changed to non-cooperative game. Eventually, their game is transitioned to fully autonomous driving when collision risk is high enough. It is shown that each player applies the maximum input value in the opposite direction with the other player during noncooperative game. This means that the two players only focus on own decision and then exert their control action to compensate for the effect of the other player's control action. Eventually, control action of each player converges to the maximum possible value. After successful lane change, game type is returned to cooperative game because there is no collision risk. The game type 3, 5 and 4 in Figure 2.19a means cooperative game, non-cooperative game and fully autonomous mode, respectively. The control authority in-

formation in Figure 2.19b indicates how much players want to control a vehicle by their intention. In other words, control authority represents 1 when a player wants to fully control a vehicle by his own intention but 0 if a player is willing to follow the other's action.



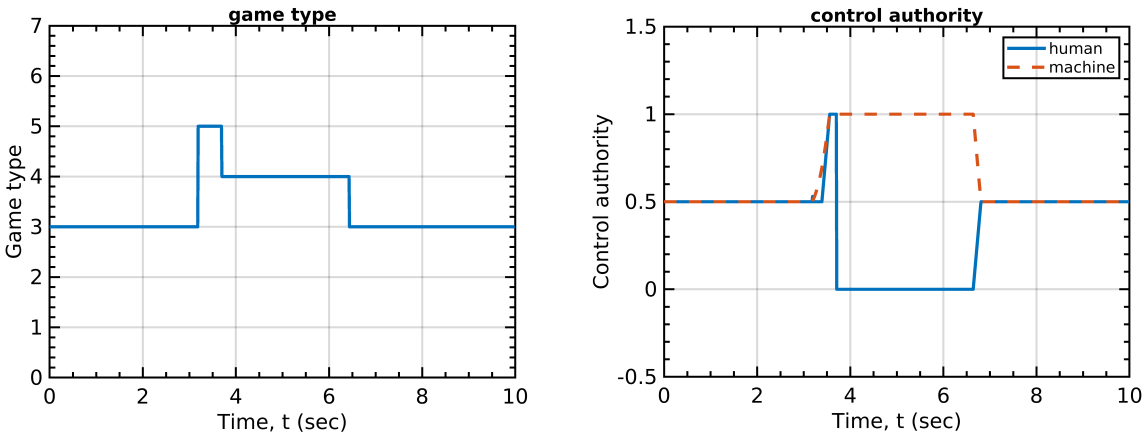
(a) lateral position (b) yaw angle

Figure 2.17: Vehicle trajectory - lane change with game transition



(a) Steering torque input (b) Long. accel. input

Figure 2.18: Control inputs - lane change with game transition

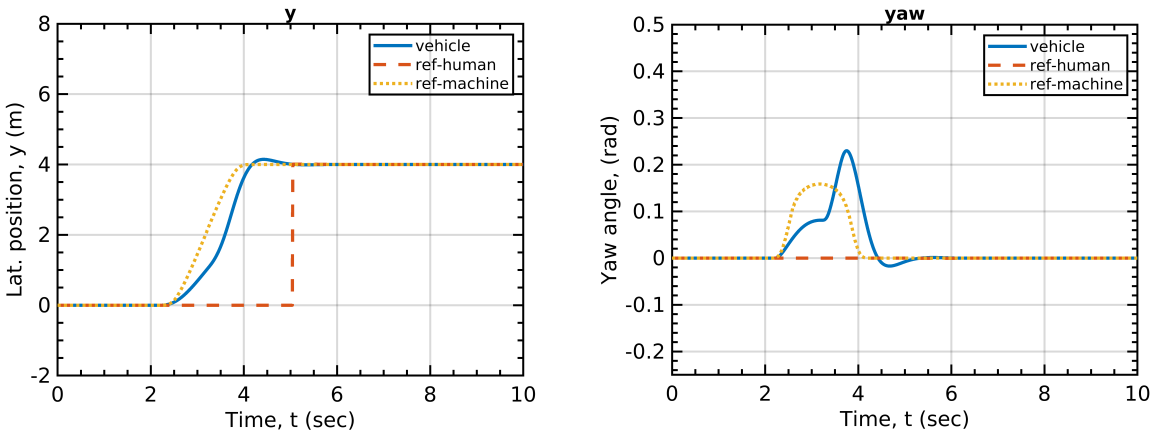


(a) Game type

(b) Control authority

Figure 2.19: Control inputs - lane change with game transition

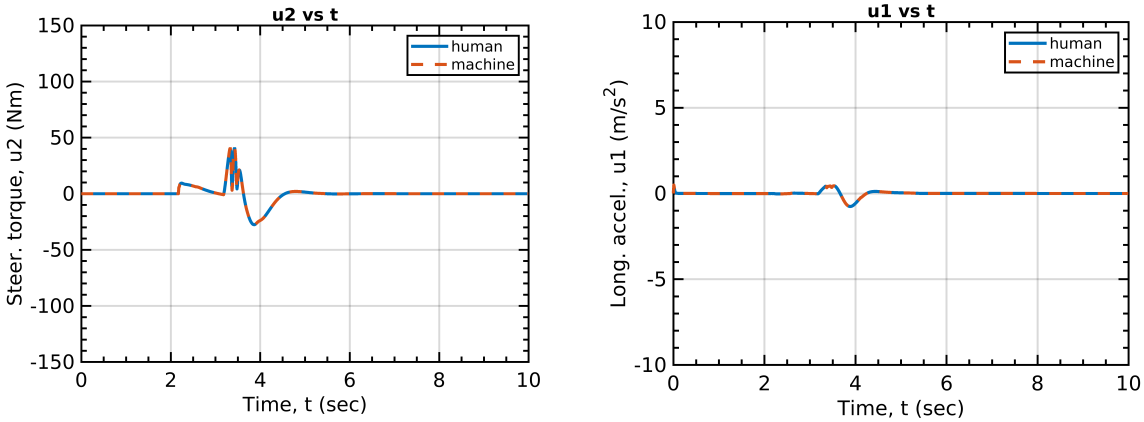
Figure 2.20, 2.21 and 2.22 illustrate the other simulation results with a human driver who follows to machine's decision in the conflict condition. It is shown that ego-vehicle is able to avoid a collision from the front vehicle. Unlike previous results shown in Figure 2.17, 2.18 and 2.19, game is not changed and stay the same in cooperative game because human driver follows machine decision in the conflict condition. To avoid a collision from the front vehicle, human driver reduces his control authority to follow machine's decision and machine increase its control authority. After avoiding a collision from the front vehicle, their control authorities go back to the default value by an assumption that two players intend to control a vehicle together with half of their control authority in normal driving situations in which they have a similar decision.



(a) lateral position

(b) yaw angle

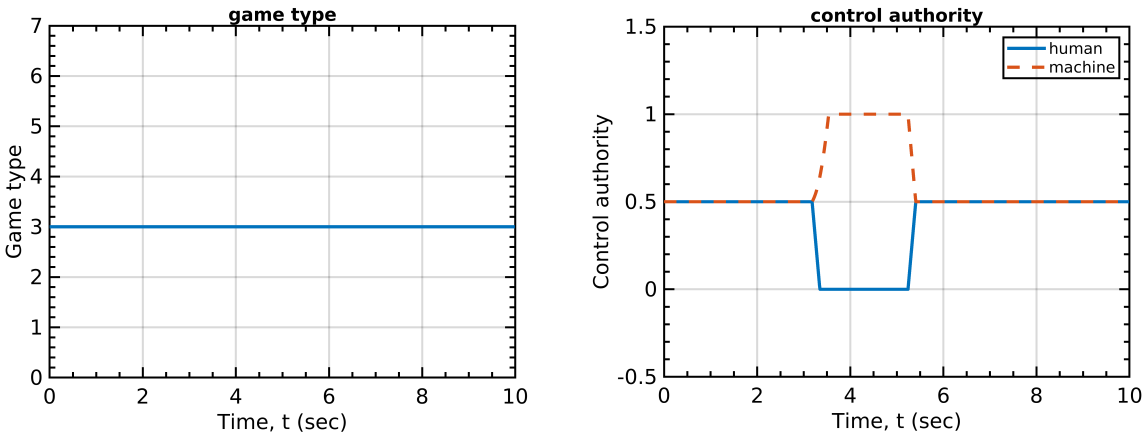
Figure 2.20: Vehicle trajectory - lane change with game transition & cooperative driver



(a) Steering torque input

(b) Long. accel. input

Figure 2.21: Control inputs - lane change with game transition & cooperative driver



(a) Game type (b) Control autority

Figure 2.22: Control inputs - lane change with game transition & cooperative driver

Also, additional simulation was conducted without any front vehicle. In other words, in this scenario it is assumed that a human driver makes a lane change by his preference, even though lane change is not necessary. Figure 2.23 illutrates this simulation scenario.

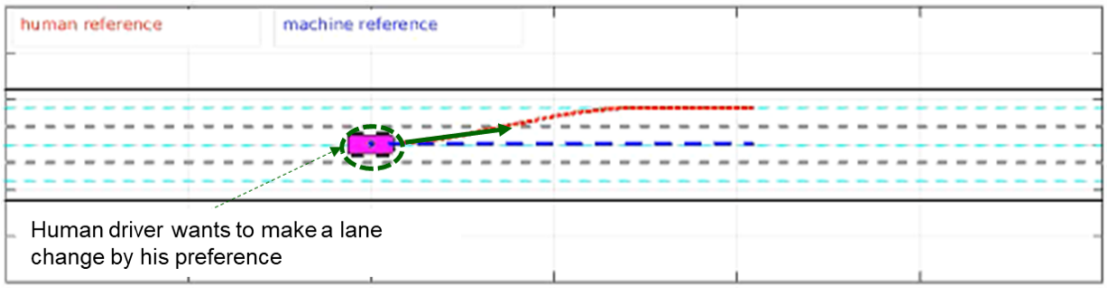
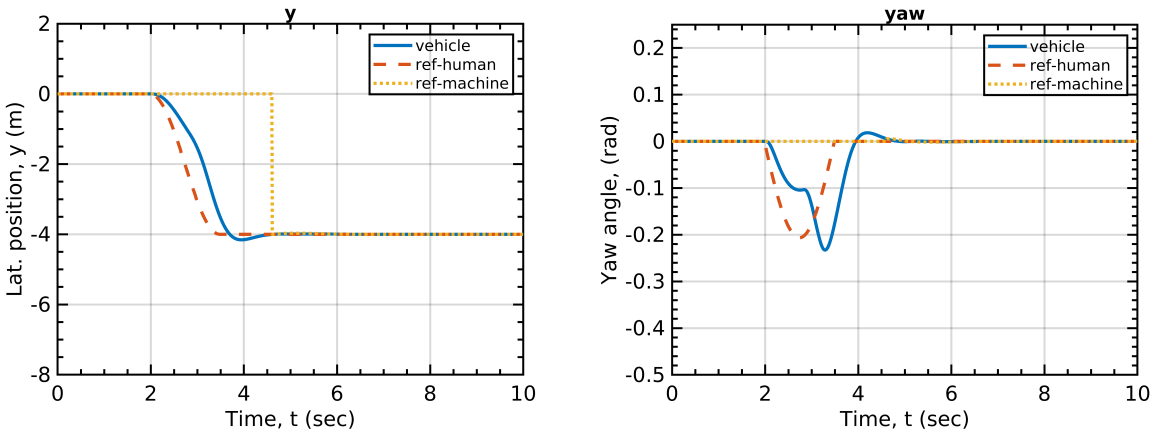


Figure 2.23: Simulation scenario - Lane change by human driver's preference

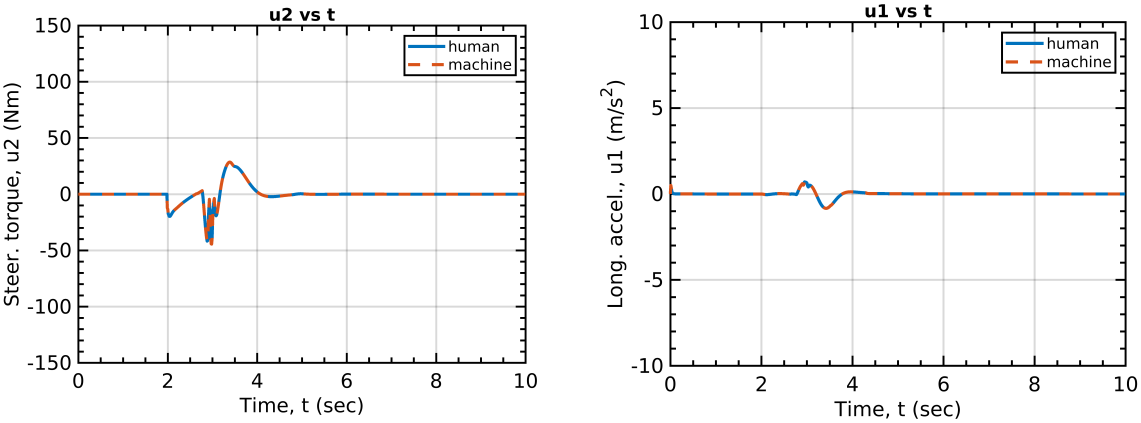
In this scenario, it is expected that machine has to reduce its control authority to follow human driver's decision in this scenario. Simulation results are shown in Figure 2.24, 2.25 and 2.26. From the simulation results, it can be analyzed that machine reduces its control authority and their game is kept as cooperative game. That is, machine cooperates with human driver in a non-safety critical condition, even though machine has a different decision from human driver.



(a) lateral position

(b) yaw angle

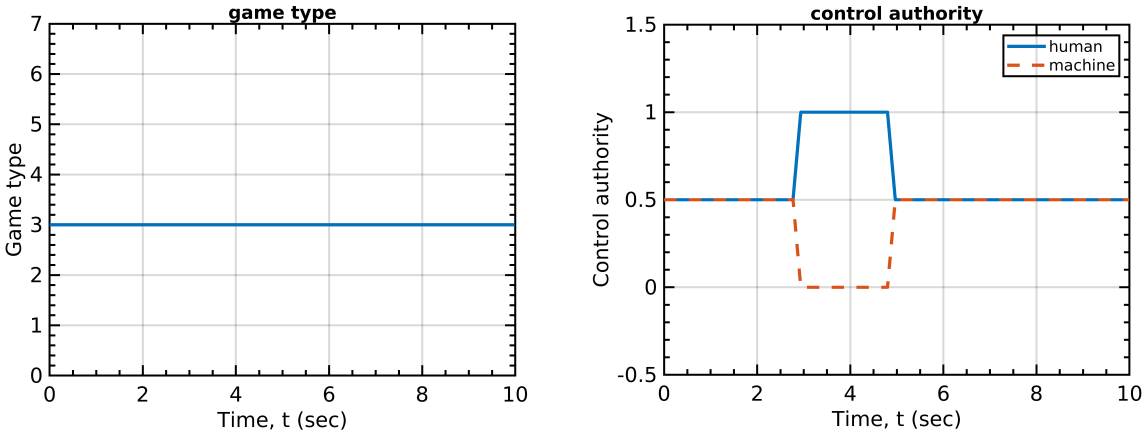
Figure 2.24: Vehicle trajectory - lane change by driver's preference



(a) Steering torque input

(b) Long. accel. input

Figure 2.25: Control inputs - lane change by driver's preference



(a) Game type

(b) Control autority

Figure 2.26: Control inputs - lane change by driver's preference

From these simulation results with game transition, it is investigated that machine is capable of respecting human driver's decision in driving conditions without collision risk and is able to control a vehicle based on its own decision in driving conditions with high collision risk by taking all control authority.

2.3.3 Lane change scenario with an interrupting vehicle

In this scenario, more complicated driving situation is investigated. Lane change is desired to avoid a slower front vehicle in this scenario but there exist another vehicle interrupting a lane change of ego-vehicle by making a lane change to the same lane. Figure 2.27 gives brief explanation about this simulation scenario.

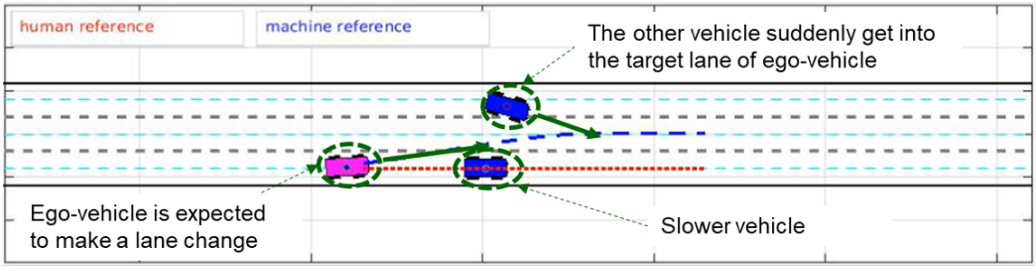
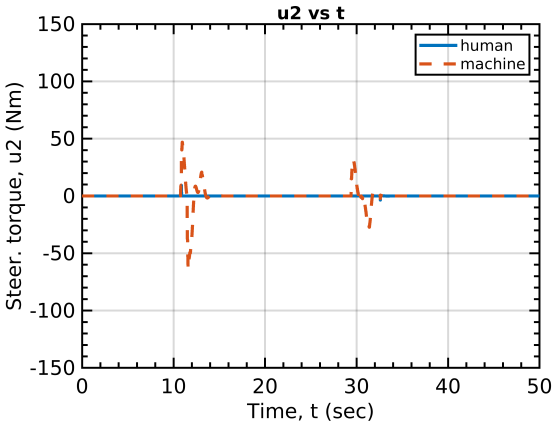
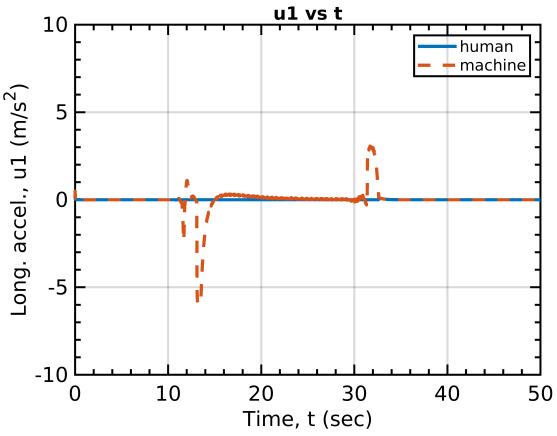


Figure 2.27: Simulation scenario - Lane change with an interrupting vehicle

In this situation, ego-vehicle needs to go back to the original lane and then slow down to follow the front vehicle. That is, this simulation scenario is intended for lateral and longitudinal maneuvers. In this simulation, it is presumed that human driver cannot recognize the interrupting vehicle. The simulation results are shown in Figure 2.29, 2.28 and 2.30. Firstly, ego-vehicle tries to make a lane change to avoid the front vehicle, but the other vehicle makes a lane change to the same lane. Therefore, ego-vehicle goes back to the original lane and then reduce vehicle speed to keep safe distance from the front vehicle and then make a lane change when lane change is available. The game type is changed during driving. Firstly, it starts as cooperative game and then is changed to non-cooperative game. Finally, ego-vehicle is operated in fully autonomous driving mode by the machine taking all control authority to handle dangerous situation, so that ego-vehicle successfully avoids a collision from surrounding vehicles. After successfully lane change is done, two players go back to a normal driving condition.

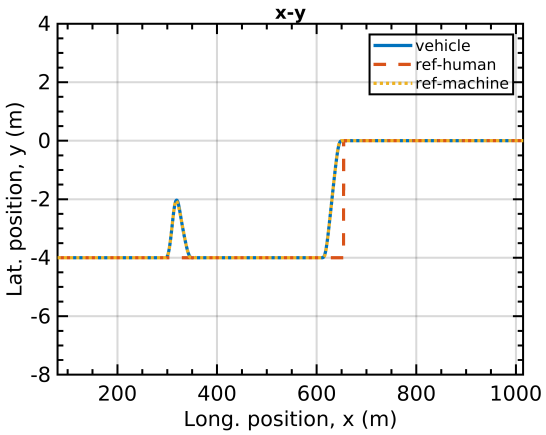


(a) Steering torque input

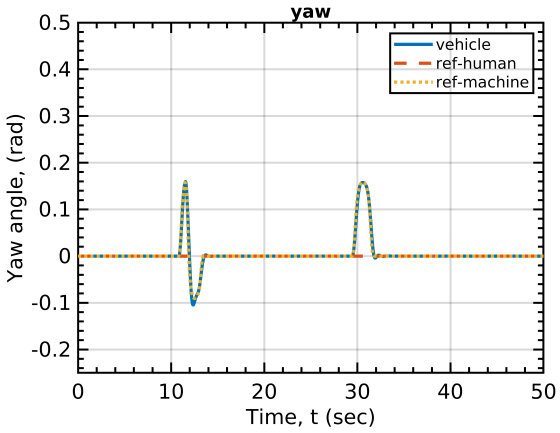


(b) Long. accel. input

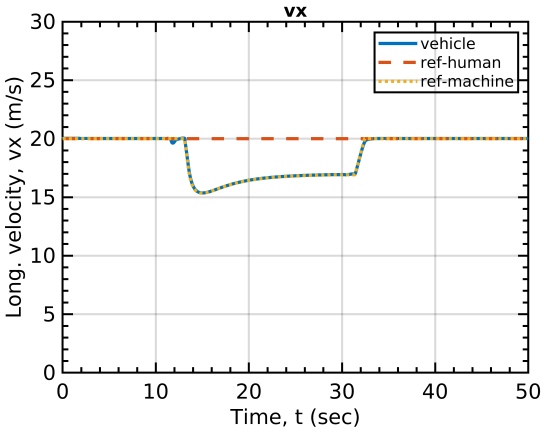
Figure 2.28: Control inputs - lane change with an interrupting vehicle



(a) vehicle position (x-y)

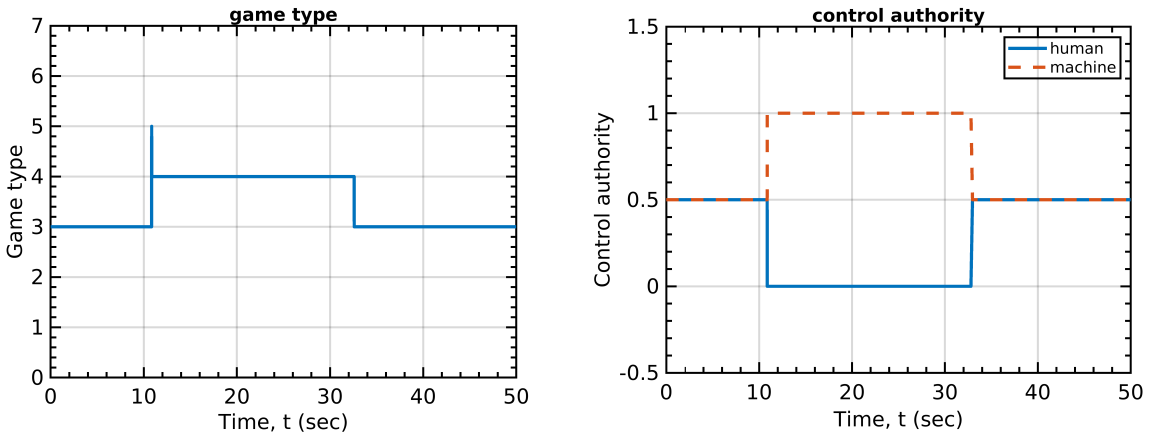


(b) yaw angle



(c) long. velocity

Figure 2.29: Vehicle trajectory - lane change with an interrupting vehicle



(a) Game type

(b) Control autority

Figure 2.30: Control inputs - lane change with an interrupting vehicle

2.4 Conclusions

In this work, shared control between human driver and machine is studied based on game theoretical model predictive control (MPC) framework. Four types of game are examined for interaction between human and machine - non-cooperative game with simultaneous move, non-cooperative game with leader-and-follower, non-cooperative game with sequential move and co-operative game. Game transition is analyzed based on finite state machine (FSM), and shared control strategy is suggested using information about collision probability and tracking error. Simulation results show shared driving successfully deals with safety-critical conditions in which machine demands high control authority and non-safety critical conditions in which machine needs to cooperate with human driving by adjusting its control authority.

The proposed game theoretic MPC based controller requires lots of computational efforts, due to inherent computational complexity associated with the property of receding horizon optimization based control methodologies. It is an open research area in MPC based control field to reduce online computational burden, and many researchers suggested computationally efficient approaches for real-time implementation [65][66][67]. In this research, it is focused that mathematical formulation of interaction between human driver and machine to understand their relationship

under game theoretic framework, rather than computational efficiency. It can be expected that the computational burden of the proposed game theoretic MPC controller can be reduced using existing computational efficient solutions for MPC based controllers.

3. REINFORCEMENT LEARNING BASED DECISION MAKING FOR SELF-DRIVING

In this section, reinforcement learning based decision making model is specified for self-driving. The decision making model generates behavioral level decisions and is a very significant part in self driving system. Recently, reinforcement learning based approaches for decision making model have received much attention due to its ability of learning from experiences in an environment, even though knowledge about an environment is not known [29]-[37]. Many RL algorithms have been proposed lately and several Deep Q Networks (DQN) algorithms based decision models are suggested in this research using three different state definitions which are relative maneuver based state definition, surrounding gap based state definition and occupied grid based state definition. Designed RL based decision models are compared to other types of decision models such as conventional human driver model for traffic simulation purpose, rule based decision model and data-driven decision model. In regards to conventional human driver model, Intelligent Driver Model (IDM) and Minimizing Overall Braking Induced by Lane Changes (MOBIL) model are used, and finite state machine (FSM) based decision model is designed as a rule based model. Also, Recurrent Neural Networks (RNN) based decision model is designed as data-driven decision model using real world highway driving data, Next Generation Simulation (NGSIM) data from U.S. Department of Transportation [68]. Comparison results show RL based decision model has the best performance with respect to vehicle speed and moving distance in the simulation environment. This section consists of following subsections. Section 3.1 describes theoretical backgrounds of RL, and Section 3.2 provides details of designed RL based decision model including algorithms details, state definitions reward functions and comparison studies. Finally, conclusions are given in Section 3.4.

3.1 Theoretical backgrounds - Reinforcement learning

Reinforcement learning (RL) is sequential decision making problem under uncertainty. There exists a learner called as agent, and that which is outside of the agent is called the environment.

In the RL framework, the agent interacts with the environment and learns from its own experience to maximize the expected temporally discounted reward. The agent acts in specific state and then observes the next state and receives a reward from the environment. The reward is considered as a signal for positive or negative action taken by the agent, and the goal of RL problem is to find the optimal action policy which is a map from a given state to a probability distribution for each possible action. The environment is generally modeled as a Markov process but the agent does not know about the environment and should learn about the environment via its own experience. The conceptual structure of RL is shown in Fig 3.1. In this section, the theoretical background of reinforcement learning is described.

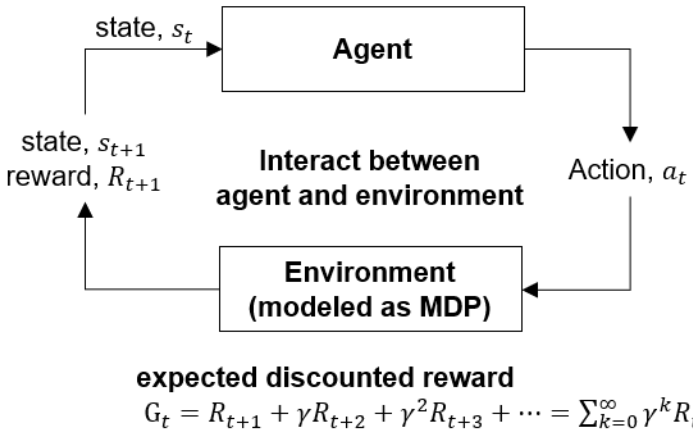


Figure 3.1: Concept of interaction between agent and environment under RL framework

If the environment is exactly known, the optimal policy can be obtained by dynamic programming via value iteration or policy iteration. But since the agent does not have prior knowledge about the environment, it has to reinforce its policy using its own experience.

3.1.1 Markov Decision Process (MDP)

The decision making process can modeled as an MDP, and in the RL framework, MDP is formed by combination of the agent's action and the environment. Mathematically, MDP can be expressed by a tuple composed of five elements (S, A, T, R, γ). The S, A, T, R and γ are the state space, action space, transition function, reward function and discount factor, respectively. The goal of the learning problem is to maximize the future rewards for an episodic task where there exists the final time, T:

$$R_t = r_{t+1} + r_{t+2} + \cdots + r_T. \tag{3.1}$$

However, for a task without a final time, the future rewards specified above became infinite. Therefore, discounted future rewards are used and formulated as:

$$R_t = r_{t+1} + \gamma r_{t+2} + \cdots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}. \tag{3.2}$$

The discount factor $0 \leq \gamma \leq 0$ is defined to be close to one if future rewards are to be considered relatively important and close to zero if the current reward is to be more important. The Markov property holds for the underlying state model if the conditional probability of future states depends only on the current state, not on all past states. In an MDP the agent's next state and reward only depend on the current state and the agent's current action.

The transition function, T, is defined as a probability distribution over next possible states $(s^{'})$ given the current state (s) and action (a) as follows.

$$\begin{aligned} &T : S \times A \longrightarrow \Pi(S) \\ &\sum_{s' \in S} T(s, a, s^{'}) = 1, \;\; \forall s \in S, \;\; \forall a \in A \end{aligned} \tag{3.3}$$

The reward function, R, is defined as a mapping from a given current state, current action and next

state to a reward as follows:

$$R : S \times A \times S \longrightarrow \mathcal{R}. \tag{3.4}$$

The agent's policy, $\pi(s, a)$, can be defined as a probability distribution over the agent's possible action space and satisfies the following condition.:

$$\sum_{a \in A} \pi(s, a) = 1, \ \ \forall s \in S. \tag{3.5}$$

The *state value* function provides the value of given state and action under the corresponding policy, and it is used to evaluate the agent's policy choices:

$$V^{\pi}(s) = E_{\pi}\{\sum_{k=0}^{T} \gamma^{k} r_{k+t+1} | s_k = s\}. \tag{3.6}$$

In the above formula T can be the final time step for an episodic task and also can be ∞ for a non-finite task. The state value function can be expressed as a recursive equation called the *Bellman expectation equation*:

$$V^{\pi}(s) = \sum_{a \in A} \pi(s, a) \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V^{\pi}(s')). \tag{3.7}$$

The optimal policy, π^*, which maximizes the discounted future rewards, satisfies the following condition:

$$V^*(S) \geq V^{\pi}(s), \ \ \forall \pi, \ \forall s \in S. \tag{3.8}$$

The optimal state value function with respect to the optimal policy, π^* is called a *Bellman optimality equation* and it can be defined by a recursive equation as follows.

$$V^*(s) = \max_{a \in A} \sum_{s' \in S} T(s, a, s')(R(s, a, s') + \gamma V^*(s')) \tag{3.9}$$

Similarly, one needs to define a state-action value function for a specific state-action pair with respect to a given policy and it is described as follows.

$$Q^{\pi}(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V^{\pi}(s')) \tag{3.10}$$

The optimal state-action value function with respect to the optimal policy, π^*, is specified as follows.

$$Q^{*}(s,a) = \sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V^{*}(s')) \tag{3.11}$$

3.1.2 Dynamic Programming

Dynamic programming (DP) is used to compute the optimal policy using an exactly known environment model based on MDP. Even though DP can be used for continuous state and action spaces problem, it is only possible to get exact solutions by DP in special cases [12]. The method based on quantization of state and action space is commonly applied for a continuous state space problem. Therefore, in this section, DP is explained under finite state and action framework to illustrate the theoretical background of DP. The key idea behind DP is to use the aforementioned value functions to find good policies. Value functions represent how valuable a specific state or specific state-action pair is, and it can be used to search for better policies.

The transition probability which depends on the previously mentioned transition function, T in (3.3), is rewritten as the probability over each possible next states with respect to a given current state and it is defined as

$$\mathcal{P}^{a}_{ss'} = Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{3.12}$$

The expected value of the next reward is rewritten as follows.

$$\mathcal{R}^{a}_{ss'} = E\{s_{t+1} = s' | s_t = s, a_t = a\} \tag{3.13}$$

An arbitrary policy is evaluated in terms of value functions based on *Bellman expectation equation*, but state value should be repeatedly computed with zero initial value until it converge. This algorithmic process can be expressed as

$$\begin{aligned} V_{k+1}(s) &= E_{\pi}\{r_{t+1} + \gamma V_k(s_{t+1}) | s_t = s\} \\ &= \sum_{a \in A} \pi(s,a) \sum_{s^{'} \in S} \mathcal{P}^{a}_{ss^{'}} (\mathcal{R}^{a}_{ss^{'}} + \gamma V_k(s^{'})) \end{aligned} \tag{3.14}$$

The subscript, k in the above refers to the iteration step and the iterations stop when the value reaches a stationary point, such that $|V_{k+1} - V_k| \longrightarrow 0$. This iterative algorithmic procedure is called as *policy evaluation*. To apply *policy evaluation*, $\mathcal{R}^{a}_{ss^{'}}$ and $\mathcal{P}^{a}_{ss^{'}}$ should be known in advance.

The *policy evaluation* is to evaluate a given current policy, but a procedure is needed to obtain a better policy which produces higher rewards. The improved policy can be obtained as the policy which maximizes state action value as follows, and this process is called as *policy improvement*.

$$\begin{aligned} \pi^{'}(s) &= \arg\max_{a} Q^{\pi}(s,a) \\ &= \arg\max_{a} \sum_{s^{'} \in S} \mathcal{P}^{a}_{ss^{'}} (\mathcal{R}^{a}_{ss^{'}} + \gamma V_k(s^{'})) \end{aligned} \tag{3.15}$$

These two phases, *policy evaluation* and *policy improvement*, constitute the policy iteration algorithm as shown below.

One drawback of policy iteration is separation of policy evaluation and policy improvement, which leads to a computational burden that requires computations of state values for the entire state space. Value iteration is an iterative algorithm which truncates policy evaluation without loss of convergence guarantees of policy iteration. The value iteration algorithm is described as follows.

Algorithm 3 Policy iteration

1. Initialization
$V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

2. Iterative policy evaluation
repeat
  Set $\triangle = 0$
  For each $s \in S$
    $v \longleftarrow V(s)$
    $V(s) \longleftarrow \sum_{a \in A} \pi(s,a) \sum_{s' \in S} \mathcal{P}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma V(s'))$
    $\triangle \longleftarrow \max(\triangle, |v - V(s)|)$
until $\triangle < \theta$ (θ is a small positive number)

3. Policy improvement
repeat
  For each $s \in S$
  $b \longleftarrow \pi(s)$
  $\pi(s) \longleftarrow \arg\max_a \sum_{s' \in S} \mathcal{P}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma V(s'))$
  $b \neq \pi(s)$, then go to policy evaluation
until $b = \pi(s)$

3.1.3 Reinforcement Learning

In the DP framework, prior knowledge about the state transition probability, $\mathcal{P}^a_{ss'}$, and rewards, $\mathcal{R}^a_{ss'}$, is required. If this prior information is not given, the optimal policy cannot be found using DP, in which case RL needs to be used to search for the optimal policy. The basic idea of RL is to learn the optimal policy from experience without prior knowledge. In other words, the agent will play an actual game and learns the state value based on observed rewards from the environment. The state value in (3.14) can be written as follows by *bootstrapping* in which updates are performed using an existing estimate.

Algorithm 4 Value iteration

Initialization
$V(s) \in R$ and $\pi(s) \in A(s)$ arbitrarily for all $s \in S$

repeat
Set $\triangle = 0$
For each $s \in S$
$v \longleftarrow V(s)$
$V(s) \longleftarrow \max_a \sum_{a \in A} \pi(s,a) \sum_{s' \in S} \mathcal{P}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma V(s'))$
$\triangle \longleftarrow \max(\triangle, |v - V(s)|)$
until $\triangle < \theta$ (θ is a small positive number)

Output a deterministic policy, π, such that
$\pi(s) \longleftarrow \arg\max_a \sum_{s' \in S} \mathcal{P}^a_{ss'}(\mathcal{R}^a_{ss'} + \gamma V(s'))$

$$V^\pi(s) = r_{t+1} + \gamma V^\pi(s') \tag{3.16}$$

Therefore, update of the state value is expressed in recursive form as follows.

$$V(s_t) = V(s_t) + \alpha[r_{t+1} + \gamma V(s_{t+1}) - V(s_t)] \tag{3.17}$$

The update is performed using the previous value plus the prediction error scaled by the learning rate α. The prediction error, $r_{t+1} + \gamma V(s_{t+1}) - V(s_t)$, is also called the *Temproal Difference* (TD) error, and $r_{t+1} + \gamma V(s_{t+1})$ is called as TD target. The main difference between state value estimation by (3.17) and DP is that the agent needs to play a real game many times to learn the state value by (3.17), otherwise the state value can be calculated by prior knowledge about the state transition probability, $\mathcal{P}^a_{ss'}$, and rewards, $\mathcal{R}^a_{ss'}$. Similarly, a state-action value function, $Q(s,a)$, can

be estimated. It is formulated as a recursive equation as follows.

$$Q(s_t, a_t) = Q(s_t, a_t) + \alpha[r_{t+1} + \gamma Q(s_{t+1}, a_{t+1}) - Q(s_t, a_t)] \tag{3.18}$$

From the aforementioned recursive formula for state-action value estimation, one RL algorithm (Sarsa or State–action–reward–state–action) can be established without prior knowledge about the environment.

Algorithm 5 Sarsa: on-policy TD control algorithm

Initialize $Q(s, a)$ arbitrarily
repeat
  Initialize the state s arbitrarily
  Choose action a based on ϵ-greedy policy
  repeat
    Take action a and move to the next state $s^{'}$ and receive reward r
    Choose the next action $a^{'}$ at state $s^{'}$ based on ϵ-greedy policy
    $Q(s, a) \longleftarrow Q(s_t, a_t) + \alpha[r + \gamma Q(s^{'}, a^{'}) - Q(s, a)]$
    $s \longleftarrow s^{'}; a \longleftarrow a^{'}$
  until reached to required number of steps or reached to a terminal state
until reached to required number of episodes

By the ϵ-greedy policy, the next action is selected as the action with the highest value in the state-action value function, but random action is also chosen with a small probability ϵ. The drawback of the Sarsa algorithm is that is an on-policy algorithm, since state-action value function estimation is really dependent on the currently followed policy. To improve convergence, an off-policy algorithm is commonly used where the state-action value function does not depend on the current policy and the optimal state-action value function is directly estimated. The off-policy TD algorithm is specified as below and it is also called Q-learning.

Algorithm 6 Q-learning: off-policy TD control algorithm

Initialize $Q(s, a)$ arbitrarily
repeat
  Initialize the state s arbitrarily
  Choose action a based on ϵ-greedy policy
  repeat
    Take action a and move to the next state $s^{'}$ and receive reward r
    $Q(s, a) \longleftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_{a'} Q(s^{'}, a^{'}) - Q(s, a)]$
    $s \longleftarrow s^{'}; a \longleftarrow a^{'}$
  until reached to required number of steps or reached to a terminal state
until reached to required number of episodes

The aforementioned algorithms are categorized as value-based learning algorithms because the algorithms estimate state-action value function and then actions are chosen based on the estimated value function. There exists another type of learning algorithm, i.e. policy based learning algorithm, which is also called as actor-critic algorithm. The actor-critic algorithm includes two parts. One is the actor (or policy) which is to choose an action for each state and the other is the critic which estimates value function. The actor-critic algorithm is described further below.
The $p(s, a)$ is the value at time t of the modifiable policy parameter of the actor and it indicates the tendency to select each action in each state. The RL algorithms mentioned in this section are described based on finite action and state spaces but they can be extended to continuous space problems with universal function approximator such as neural networks to estimate the value function.

3.2 Design of Decision Making Model Based on RL for Highway Driving

For a finite state and action space problem, the estimates of the value functions are stored in a table. For large finite state and action space problem or continuous action and state spaces problem, function approximation is applied to estimate the value function, while neural networks are commonly used for value function estimation. Recently, breakthroughs in RL have been achieved

Algorithm 7 Actor-critic algorithm

Initialize

$p(s,a) \longleftarrow 0 \;\; \forall s \in S, \;\; \forall a \in A$

$\pi(s,a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A|} e^{p(s,a)}} \;\; \forall s \in S, \;\; \forall a \in A$

repeat

Initialize the state s arbitrarily

Choose action a from state s using $\pi(s,a)$

repeat

Take action a and move to the next state $s^{'}$ and receive reward r

$\delta = r + \gamma V(s^{'}) - V(s)$

$V(s) \longleftarrow V(s) + \alpha(r + \gamma V(s^{'}) - V(s))$

$Q(s,a) \longleftarrow Q(s_t, a_t) + \alpha[r + \gamma \max_{a'} Q(s^{'}, a^{'}) - Q(s,a)]$

$s \longleftarrow s^{'}; a \longleftarrow a^{'}$

$p(s,a) \longleftarrow p(s,a) + \beta\delta$

$\pi(s,a) \longleftarrow \frac{e^{p(s,a)}}{\sum_{b=1}^{|A|} e^{p(s,a)}}$ (Gibbs softmax method)

until reached to required number of steps or reached to a terminal state

until reached to required number of episodes

by applying deep neural networks. The state-value function can be rewritten as follows where θ means parameters of the function approximator which need to be tuned by learning algorithms.

$$Q(s,a)^* \approx Q(s,a;\theta) \tag{3.19}$$

Deep Q Network (DQN), which is a Q learning algorithm using deep neural networks as function approximators with an experience replay buffer, has been recently proposed and several variant algorithms such as Double DQN, Dueling DQN and DQN with prioritized experience replay buffer have been introduced. The basic algorithm of DQN is described as follows.

The DQN algorithm can be modified using another neural network for estimating the target value of the state-action value function to overcome the unstable target value problem. Therefore, the

Algorithm 8 DQN algorithms with experience replay

Initialize replay memory $\mathcal{D}$ with capacity N
Initialize neural networks Q with random weights θ for state-value function estimation
for episode$= 1, M$ **do**
  Initialize sequence $s_1 = x_1$
  for $t = 1, T$ **do**
    Choose action a_t based on ϵ-greedy policy
      with probability ϵ, select random action a_t
      otherwise, select action $a_t = \max_a Q^*(s_t, a; \theta)$
    Take action a_t and move to the next state s_{t+1} and receive reward r_t
    Store transition (s_t, a_t, r_t, s_{t+1}) in $\mathcal{D}$
    Sample random mini-batch of transitions (s_j, a_j, r_j, s_{j+1}) from $\mathcal{D}$
    Set $Q_{target,j} = \begin{cases} r_j, & \text{for terminal } s_{j+1} \\ r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta), & \text{for non-terminal } s_{j+1} \end{cases}$
    Perform a gradient descent step on $(Q_{target,j} - Q(s_j, a_j; \theta))^2$
      $\theta \longleftarrow \theta - \alpha \bigtriangledown_\theta (Q_{target,j} - Q(s_j, a_j; \theta))$
    $s_t \longleftarrow s_{t+1}$
  end for
end for

target value of the state-action value function is replaced by $Q_{target,j} = r_j + \gamma \max_{a'} Q(s_{t+1}, a'; \theta^-)$ and target network weights θ^- are periodically copied from Q network weights θ . In this research, several DQN algorithms are applied to design decision making models for highway driving.

3.2.1 Experimental setup

In an RL framework, one needs to establish an environment which can interact with a learning agent in a learning problem. Recently, several open-source driving simulators have been released [69][70][71], while a highway driving simulator was developed by modifying an existing simulator [71]. In our work, the roadway was designed with multiple lanes while a kinematic model is used for vehicles in the simulator. The kinematic vehicle model is described as follows.

$$\dot{x} = v\cos(\theta)$$
$$\dot{y} = v\sin(\theta)$$
$$\dot{\theta} = v\frac{\tan(\delta)}{L} \tag{3.20}$$

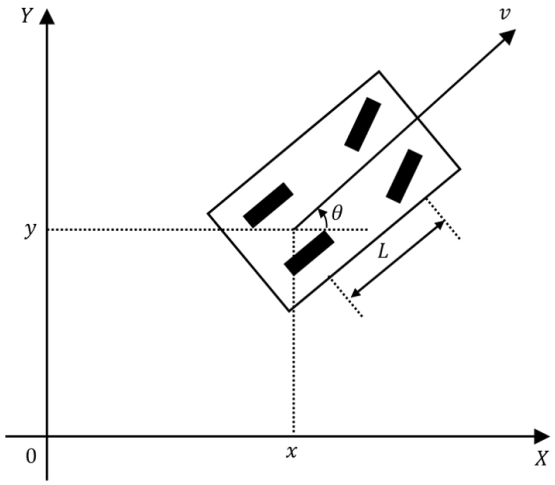


Figure 3.2: Kinematic vehicle model

Decision making models of longitudinal and lateral maneuvers are required for other vehicles in the driving simulation and an Intelligent Driver Model (IDM) in [72] is used for longitudinal decision model and an Minimizing Overall Braking decelerations Induced by Lane changes (MOBIL) model in [73] is used for lateral decision model for other vehicles. The IDM model described in (3.21) produces longitudinal acceleration as a decision based on relative speed and distance to a leading vehicle and its parameters is summarized in the table below.

$$\dot{v} = a\left[1 - \left(\frac{v_0}{v}\right)^{\delta} - \left(\frac{s^*(v, \triangle v)}{s}\right)\right]$$
$$s^*(v, \triangle v) = s_0 + vT + \frac{v\triangle v}{2\sqrt{ab}} \tag{3.21}$$

parameter	nominal value
desired speed v_0	25 m/s
free acceleration exponent δ	4
desired time gap T	1.5 s
jam distance s_0	2.0 m
maximum acceleration a	1.4 m/s^2
desired deceleration b	2.0 m/s^2

Table 3.1: IDM parameters

The MOBIL model described in (3.22) produces the lateral lane change decision based on relative speed and distances to the following vehicle in the current lane and the following vehicle in the next lane. The parameters of the MOBIL model are summarized in the table below.

$$\tilde{a_c} - a_c + p(\tilde{a_n} - a_n + \tilde{a_o} - a_o) > \triangle a_{th} \tag{3.22}$$
$$\tilde{a_n} \geq -b_{safe}$$

In (3.22), a_c and $\tilde{a_c}$ represent ego-vehicle's current and the next acceleration after a lane change respectively, and a_n and $\tilde{a_n}$ represent current and the next acceleration of following vehicle in the adjacent lane, respectively. Also, a_o and $\tilde{a_o}$ represent the current and the next acceleration of following vehicle in the current lane, respectively.

parameter	nominal value
politeness p	$0 \leq p \leq 1$
lane changing threshold $\triangle a_{th}$	0.1 m/s^2
maximum safe deceleration b_{safe}	4 m/s^2

Table 3.2: MOBIL parameters

3.2.2 Problem statement

3.2.2.1 State space definitions

Human drivers perceive the driving situation and make decisions using this information. It is generally understood that human drivers uses information about relative maneuvers with respect to the surrounding vehicles to make decisions. Under this principle, relative maneuver based states are used in our learning algorithms to represent the state of the environment. Highway driving in a multi-lane setting is illustrated in the figure below.

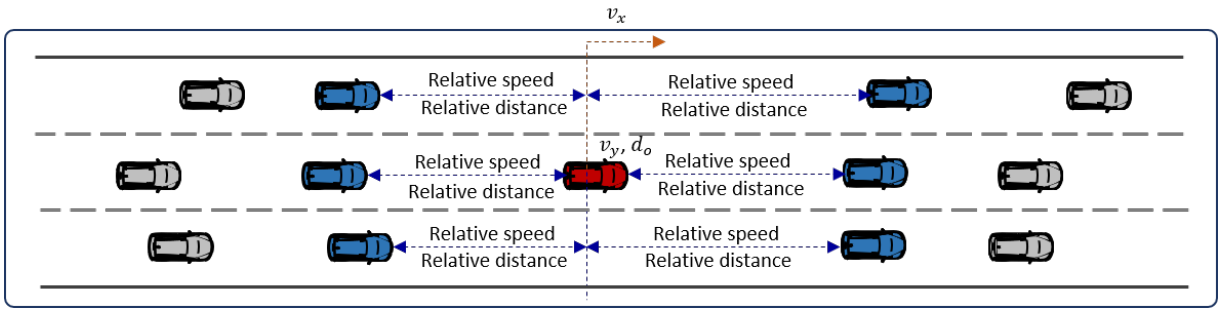


Figure 3.3: Relative maneuver based state definition

The relative maneuver based state space is a 15 dimensional space and includes the ego-vehicle's longitudinal and lateral speeds, its lateral position on the road and the relative speed and distance to the surrounding vehicles.

$$s = [v_x, x_y, d_o, \triangle x_{rel,F}, \triangle v_{rel,F}, \triangle x_{rel,R}, \triangle v_{rel,R}, \triangle x_{rel,FL}, \triangle v_{rel,FL}, \triangle x_{rel,FR}, \triangle v_{rel,FR},$$
$$\triangle x_{rel,RL}, \triangle v_{rel,RL}, \triangle x_{rel,RR}, \triangle v_{rel,RR}] \tag{3.23}$$

Elements	Descriptions
v_x	ego-vehicle's longitudinal speed
v_y	ego-vehicle's lateral speed
d_o	ego-vehicle's distance to lane center
$\triangle x_{rel,F}$	relative distance to front vehicle
$\triangle v_{rel,F}$	relative speed to front vehicle
$\triangle x_{rel,R}$	relative distance to rear vehicle
$\triangle v_{rel,R}$	relative speed to rear vehicle
$\triangle x_{rel,FL}$	relative distance to front left vehicle
$\triangle v_{rel,FL}$	relative speed to front left vehicle
$\triangle x_{rel,FR}$	relative distance to front right vehicle
$\triangle v_{rel,FR}$	relative speed to front right vehicle
$\triangle x_{rel,RL}$	relative distance to rear left vehicle
$\triangle v_{rel,RL}$	relative speed to rear left vehicle
$\triangle x_{rel,RR}$	relative distance to rear right vehicle
$\triangle v_{rel,RR}$	relative speed to rear right vehicle

Table 3.3: Relative maneuver based state definition

In addition, it can be interpreted that the human driver uses the surrounding gap information to make a decision. Thus, another state definition based on surrounding inter-vehicular gaps is also considered as an alternative. Figure 3.4 shows the gap based state definition in highway driving with multiple lanes.

The surrounding gap based state space comprises 17 elements (i.e., forms a 17 dimensional space) and includes the ego vehicle's longitudinal and lateral velocities, its lateral position on the road, the length of the surrounding gaps, the time rate of change of the surrounding gaps and distances to these gaps.

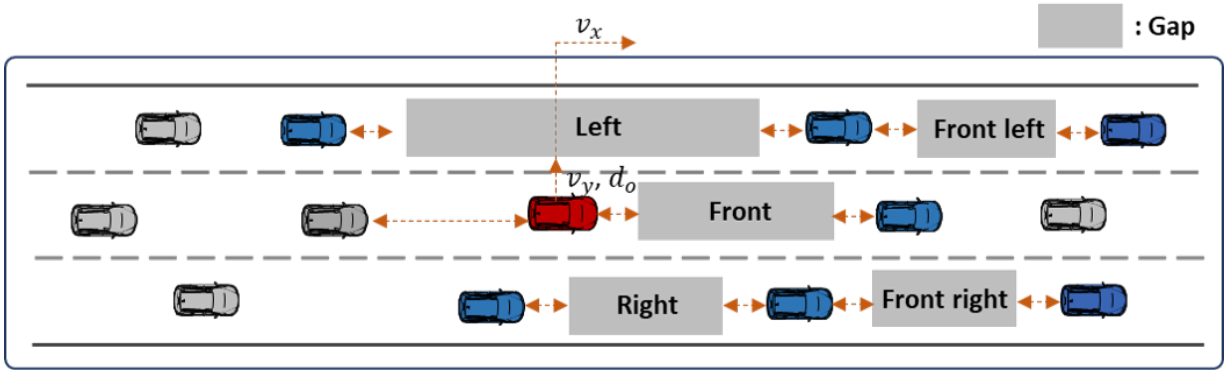


Figure 3.4: Gap based state definition

$$
\begin{aligned}
s = [&v_x, v_y, d_o, Gap_F, \dot{Gap}_F, Gap_{FL}, \dot{Gap}_{FL}, distance_{FL}, \\
&Gap_{FR}, \dot{Gap}_{FR}, distance_{FR}, Gap_{RL}, \dot{Gap}_{RL}, distance_{RL}, \\
&Gap_{RR}, \dot{Gap}_{RR}, distance_{RR},]
\end{aligned} \tag{3.24}
$$

In an RL problem, the state space has to be defined carefully to reflect the real situation but it is very hard to define the state space appropriately. Previously defined state space definitions (relative maneuver based and surrounding gap based definitions) are designed manually as feasible ways that reflect the real driving situation on highways. Convolutional neural networks (CNN) are capable of extracting features (states) from raw data which can be formulated in image-like format. Therefore, an occupancy grid based state space is also defined so as to facilitate the application of CNN. The occupancy grid based definition is illustrated below where its format is similar to the three channel image data.

In the structure of CNN, actual features (states) are extracted in the convolutional layers and then the extracted features are fed into the final fully connected layers.

Elements	Descriptions
v_x	ego-vehicle's longitudinal speed
v_y	ego-vehicle's lateral speed
d_o	ego-vehicle's distance to lane center
Gap_F	front gap
$\dot{Gap}_F$	rate of front gap
Gap_{FL}	front left gap
$\dot{Gap}_{FL}$	rate of front left gap
$distance_{FL}$	distance to front right gap
Gap_{FR}	front right gap
$\dot{Gap}_{FR}$	rate of front right gap
$distance_{FR}$	distance to front right gap
Gap_{RL}	rear left gap
$\dot{Gap}_{RL}$	rate of rear left gap
$distance_{RL}$	distance to rear left gap
Gap_{RR}	rear right gap
$\dot{Gap}_{RR}$	rate of rear right gap
$distance_{RR}$	distance to rear right gap

Table 3.4: Gap based state definition

3.2.2.2 Action space definition

Self-driving systems have to be capable of several driving tasks. Therefore, a hierarchical architecture is widely used for self-driving. The self-driving system generally consists of four modules, i.e. sensing , perception, planning and control modules. The sensing module as the name implies, senses the environment using several sensors installed on the vehicle while the perception module processes the information obtained from sensed data and produces physically meaningful information. The planning module includes a decision making sub-module and a path planning sub-module. The decision making sub-module produces the behavioral decisions using information from the perception module while the path planner generates a target trajectory to be followed. The control module controls the vehicle to follow the trajectory generated by the path planning module. Under the hierarchical architecture of self-driving, the complicated self-driving tasks can be handled efficiently, hence a hierarchical architecture is widely used. In another type of architecture, termed *end-to-end learning* for self-driving, raw data from sensors are fed into a

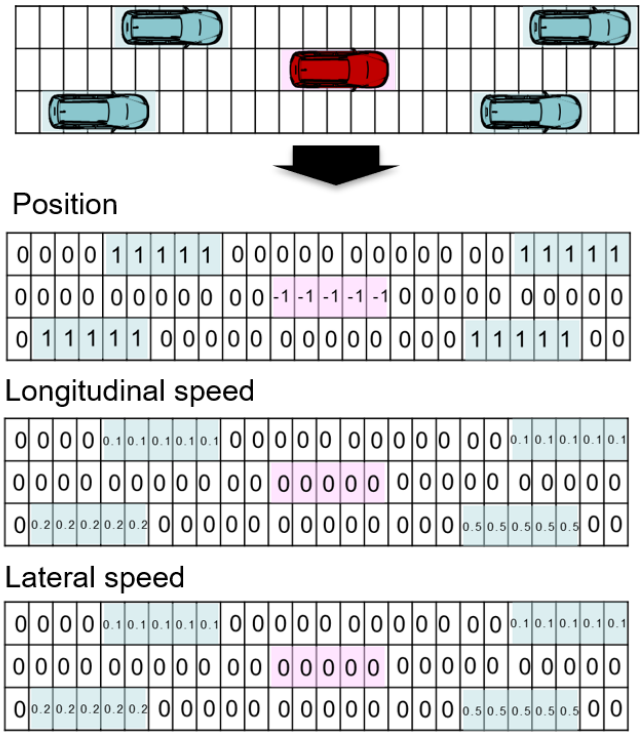


Figure 3.5: Occupied grid based state definition

deep neural networks and the final control outputs are directly produced by the network. In this architecture, control stability or robustness cannot be guaranteed. Therefore, the hierarchical architecture is more appropriate for self-driving, since driving tasks are decomposed and the control module can be designed to guarantee system stability (albeit within some limits) while also offering some level of interpretability. The behavioral decision making is focused on in this research where the action space constitutes the high-level behavioral decision domain. In other words, the action space is defined to include 5 elements as described below.

- *keep lane*
- *left lane change*
- *right lane change*
- *acceleration* ($2m/s^2$)
- *deceleration* ($-2m/s^2$)

3.2.2.3 *Reward function definition*

The definition of the reward function is critical for learning performance. If the reward function is not defined appropriately, it is very hard to learn the so-called value function. Regarding the driving problem, generally speaking higher speeds are preferred but collision must be avoided. Therefore, the reward function is defined to encourage an agent to drive with high speed without a collision. The reward function consists of three parts, speed reward, action reward and distance reward, as described follows.

- *speed reward* R_s: $w_s \dfrac{(v_x - v_{min})}{(v_{max} - v_{min})}$
- *acceleration reward* R_a: $w_a a_x$
- *distance reward* R_d: $w_d f_d(x)$

The $f_d(x)$ is a reward function dependent on the distance to the leading vehicle and it is defined below, where x is the distance to the front vehicle; a and b are user-defined parameters.

$$f_d(x) = -\exp(-a * (x - b)) \tag{3.25}$$

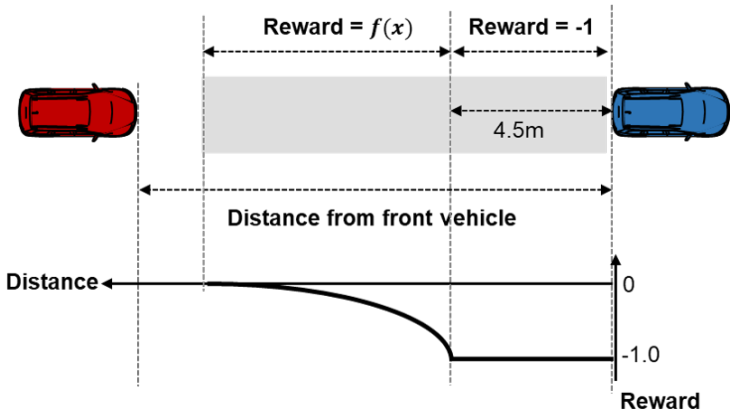


Figure 3.6: Distance reward function

The final rewards are determined as the summation of the aforementioned three sub-rewards.

$$R = R_s + R_a + R_d \tag{3.26}$$

By this reward definition, a higher reward is obtained for higher speeds and the agent can get an additional reward for an acceleration action. Also, the agent gets a negative reward (or penalty) based on its proximity to the front vehicle. Therefore, it is prevented from driving too close to the front vehicle.

3.2.3 Learning results

3.2.3.1 Driving scenario

Simulation is performed in highway driving on three lanes road. One stopped vehicle is placed in front of the ego-vehicle's initial position. The other vehicles are operated based on IDM and MOBIL model. The vehicles' position, lane and speed are randomly chosen and the speed range is defined to be between 20m/s and 30m/s.

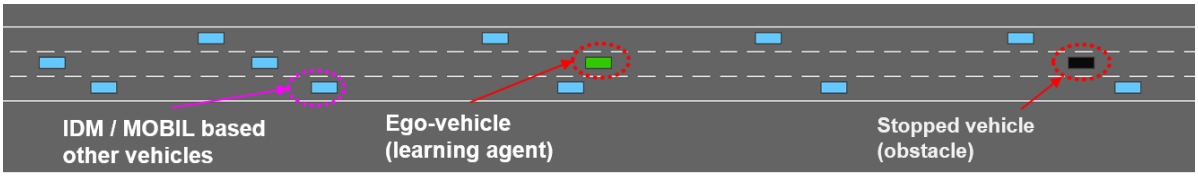


Figure 3.7: Driving scenario

3.2.3.2 Neural networks structures

Several variant algorithms of DQN are applied to this problem. For relative maneuver based and gap based state definitions discussed previously, a fully connected neural network with 2 hidden layers is used while a convolutional neural network with 3 convolutional layers is used to extract features from the occupied grid and then fully connected networks with 2 hidden layers is

placed after the convolutional layers. The neural network structure used in each learning algorithm is summarized in the table below.

Algorithms	Relative maneuver (Neural Networks)	Gap (Neural Networks)	Occupied grid (Convolutional Neural Networks)
DQN	Input: 15 Hidden: - fully connected: 256 - fully connected: 256 Output: 5	Input: 15 Hidden: - fully connected: 256 - fully connected: 256 Output: 5	Input 3X24X88 Hidden: - convolutional : [3,32,2,1,1] - maxpooling : [2,2,0] - convolutional : [3,32,2,1,1] - maxpooling : [2,2,0] - convolutional : [3,32,2,1,1] - maxpooling : [2,2,0] - fully connected: 256 - fully connected: 256 Output: 5
Double DQN	↑	↑	↑
Dueling DQN	↑	↑	↑
DQN with PER	↑	↑	↑
DQN with Noisy Net	↑	↑	↑

Table 3.5: Neural networks structures

3.2.3.3 *Learning results*

Learning curves are compared with respect to several algorithms applied in this problem. Figure 3.8 shows the comparison results for several DQN algorithms with a relative maneuver based state definition. There is no significant difference and the algorithms show similar levels of performance. Also, learning curves are compared with respect to three different state definitions – relative maneuver, gap and occupied grid. The comparison results with respect to state definitions are shown in Figure 3.9. It is found that the three definitions of the state space show similar performance. Relative maneuver and gap based states enable manually defined features while learning results show these state space definitions appropriately capture the driving situation. Features are extracted automatically by a convolutional neural network from the occupied grids and it is understood that the extracted features show similar level of performance to manually defined features in relative maneuver and surrounding gap based state space formulations.

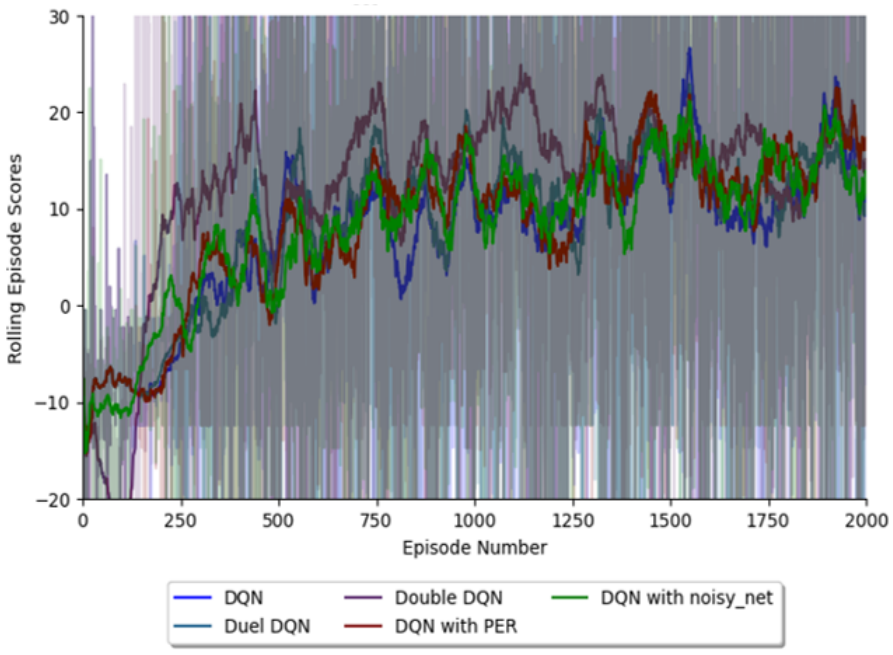


Figure 3.8: Learning curves - comparison w.r.t algorithms

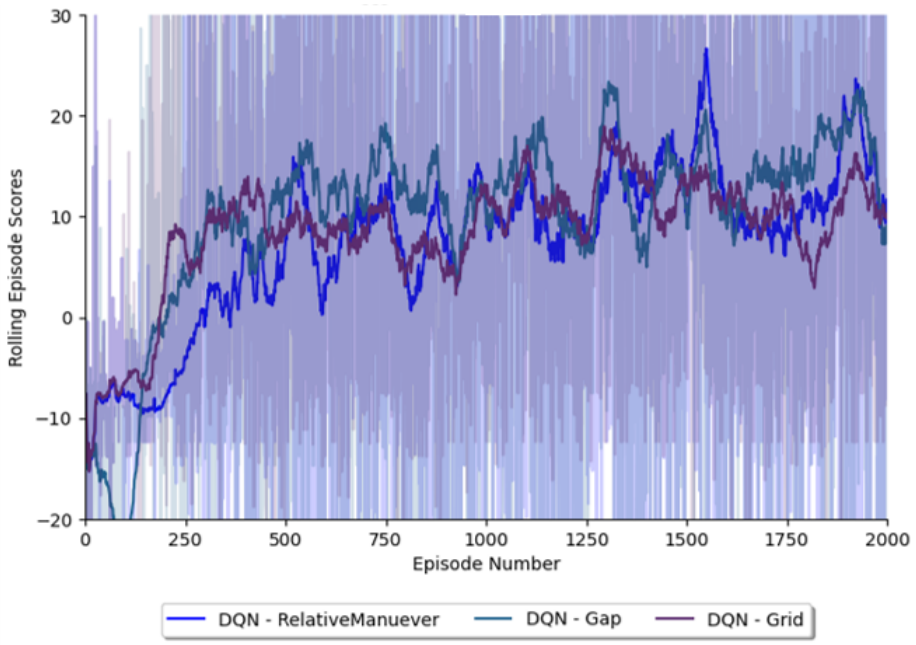


Figure 3.9: Learning curves - comparison w.r.t state definitions

3.3 Comparison studies with other decision making models

In this subsection, comparison studies are addressed to verify performance of designed RL based decision model. Three types of other decision models are also designed and taken into account in this study to compare performances with the designed RL based decision model. Firstly, conventional human driver simulation models, which are Intelligent Driver Model (IDM) for longitudinal behavior and Minimizing Overall Braking Induced by Lane Changes (MOBIL) model for lateral behavior, are developed. A rule-based decision model, which is a still the mainstay of industrial practice, is constructed for comparison purpose. Finally, a data-driven decision model using real-world dataset is designed based on recurrent neural networks (RNNs) and particle swarm optimization (PSO).

3.3.1 Human driver model

Intelligent Driver Model (IDM) is a conventional car-following model commonly used for traffic simulation [72]. In other words, IDM is used for longitudinal behavior model of human

driver in traffic simulation. Minimizing Overall Braking Induced by Lane Changes (MOBIL) model is a conventional human driver model for lane change considering the benefit with respect to longitudinal acceleration after a lane change [73]. IDM is specified in Equation (3.21) and Table 3.1 and MOBIL is illustrated in Equation (3.22) and Table 3.2 in the previous section. In this comparison studies, IDM and MOBIL models are used as conventional human driver simulation models for both of longitudinal and lateral behaviors.

3.3.2 Rule-based decision model

In regards to the rule-based model, a finite state machine (FSM) based model is designed. Rule-based decision models are highly dependent on specific rules employed in the model. Consequently, the performance of the rule-based model can vary according to the internal rules. In this study, a rule based model is designed under an FSM framework with transition conditions considering information about driving situations, such as collision probability and lane change feasibility. Because rule-based models require a very dedicate job to define the rules in consideration of all expected situations, it is very hard to design a standard rule-based decision model. Therefore, a rule-based model is designed for comparison purposes considering highway driving situation and is used as a representative rule-based model in this comparison studies. In total five states are considered including "follow desired speed", "follow leading vehicle", "left lane change", "abort left lane change", "right lane change" and "abort right lane change", while transition conditions are defined in consideration of collision probability, lane change feasibility and lane change status. Figure 3.10 below show the designed rule-based model using an FSM.

3.3.3 Data-driven decision model

In this study, data-driven decision model is also designed using real-world highway driving datasets. Next Generation SIMulation (NGSIM) dataset which is commonly used for studies on highway driving are used [68]. NGSIM data are processed to extract relative maneuver information with respect to surrounding vehicles and then recurrent neural networks (RNNs) based models are trained using the processed data to produce lane change decision. For longitudinal behavior,

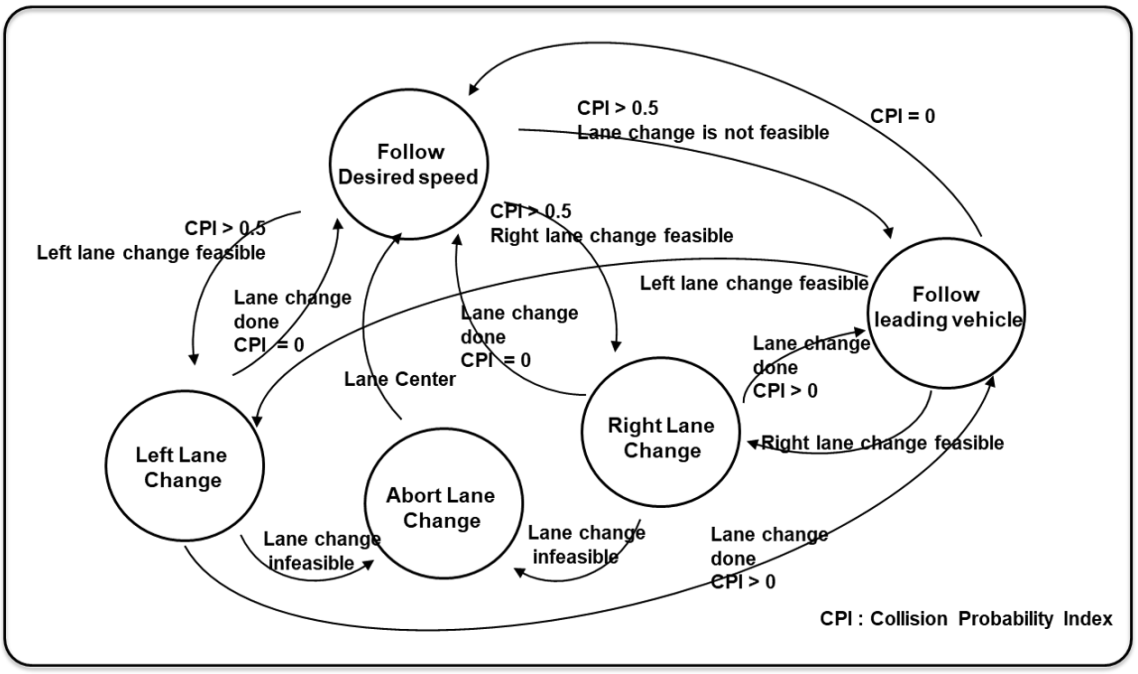


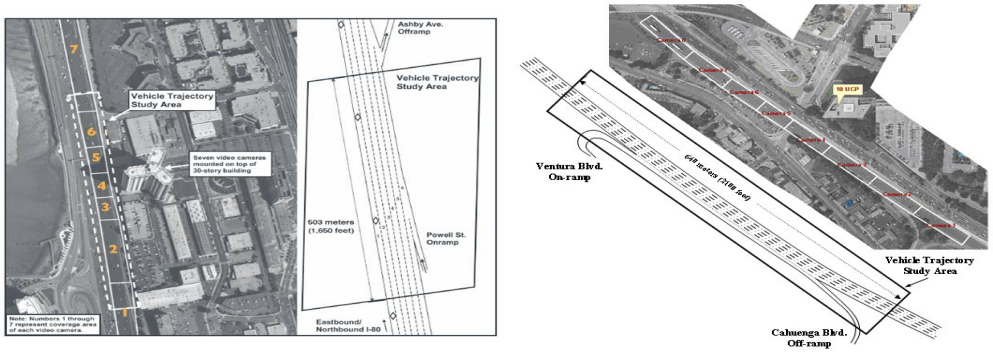
Figure 3.10: Rule-based model by FSM

parameters of IDM are calibrated by particle swarm optimization (PSO) using NGSIM. RNN based lane change decision model and IDM with calibrated parameters are used as data-driven decision model.

3.3.3.1 NGSIM dataset

The NGSIM dataset has been a widely used public dataset for research on driver models [74] [75] [76] [77] [78]. This dataset was collected by the Federal Highway Administration of the U.S. Department of Transportation from 2003 to 2005 and consists of four different sub-datasets, southbound US-101 and Lankershim Boulevard in Los Angeles, California, eastbound I-80 in Emeryville, California and Peachtree Street in Atlanta, Georgia. All sub-datasets in NGSIM are released to the public at the NGSIM website [68]. Among these four sub-datasets, I-80 and US-101 datasets were collected on highways while the others have urban scenes. In this paper, only the I-80 and US-101 datasets are used to focus on highway driving. In I-80, total 5648 vehicle trajectories were collected on about a 500m section of the road with six lanes and are divided into

three 15 minutes intervals: 4:00 pm to 4:15 pm, 5:00 pm to 5:15 pm, and 5:15 pm to 5:30 pm. In total 6101 vehicle trajectories were recorded on about 640m section of the road with six lanes on US-101 and are segmented into three 15 minutes intervals: 7:50 am to 8:05 am, 8:05 am to 8:20 am, and 8:20 am to 8:35 am. Figure 3.11 shows the overview of study areas on I-80 and US-101 datasets [68]. The data NGSIM dataset was collected by 0.1 second sampling interval.



(a) I-80 dataset

(b) HS-101 dataset

Figure 3.11: Overview of I-80 and US-101 datasets in NGSIM [6]

3.3.3.2 Lane change decision model

Recurrent neural networks (RNNs) are applied for lane change decision model using processed data from NGSIM dataset. Recurrent neural networks (RNNs) are a class of neural networks and capable of processing time-series data and other sequential data. Basic architecture of RNNs is shown in Figure 3.12. The key idea behind RNNs is the self-loop which allow RNNs to have internal hidden state.

But basic RNNs suffer from vanishing gradient and computational explosion problems as well as difficulty with long-term dependencies. Long Short-Term Memory (LSTM) was introduced to solve the problems in the basic RNNs [79]. LSTM networks have three gates, which are input,

[6] source: "Next Generation Simulation" by Department of Transportation, Available at www.http://ngsim.fhwa.dot.gov

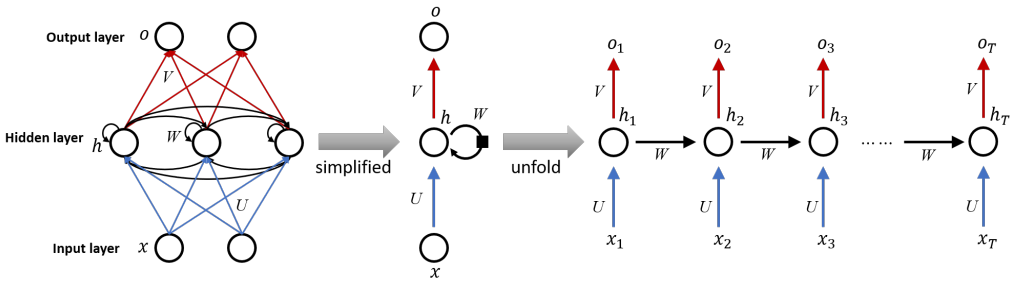
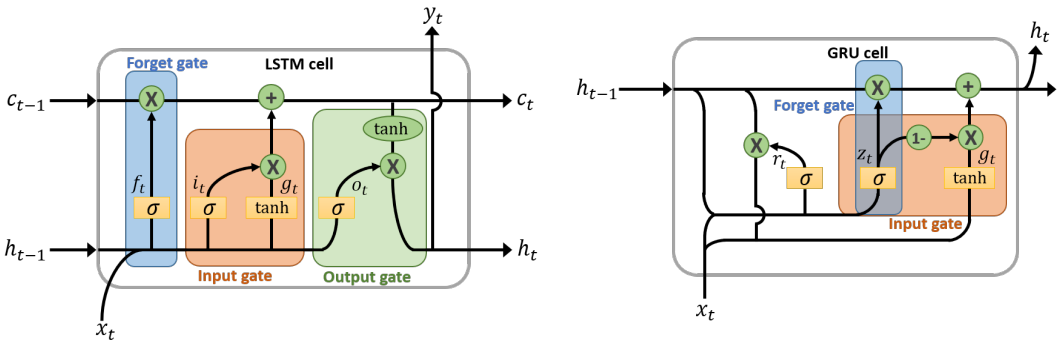


Figure 3.12: Recurrent neural network architecture

output and forget gates, and can regulate the flow of information using these gates. Therefore, LSTM networks have an ability of selectively read, write and forget information. A Gated Recurrent Unit (GRU) network is a simplified version of LSTM and has two gates, update and reset. It has been reported that GRU networks are simpler in structure but they show similar performance as LSTM networks [80]. In Figure 3.13a and 3.13b, the internal structures of LSTM and GRU cells are shown respectively.



(a) LSTM

(b) GRU

Figure 3.13: Internal structure of LSTM/GRU cells

The original NGSIM dataset does not include exact information about lane change start execution time, which needs to be obtained from existing information in the original dataset. The dataset has lane information which lets us know in which lane a vehicle is being driven. Lane information is changed at the lane crossing time. Therefore, lane change execution start time and lane change

execution end time can be obtained in accordance with lateral vehicle movements. Lateral moving distance can be calculated as the integral of the vehicle lateral speed. Therefore, lane change execution start time can be obtained as

$$t_{lc,start} = \arg\max_{t_*} \int_{t_*}^{t_{lc}} V_{lat}\ \mathrm{d}t > 1.5 \tag{3.27}$$

where, $t_{lc,start}$ is the lane change execution start time, V_{lat} is the lateral vehicle speed, and t_{lc} is the lane crossing time included in the original dataset. In (3.27), the right-hand side term, 1.5, is a threshold for lateral moving distance (meter) and it is defined to be slightly smaller than the half of a lane width. Lane width is approximately from 3.6m to 3.8m in the dataset. Similarly, the lane change execution end time, $t_{lc,end}$, is obtained as shown in (3.28)

$$t_{lc,end} = \arg\min_{t_*} \int_{t_{lc}}^{t_*} V_{lat}\ \mathrm{d}t > 1.5 \tag{3.28}$$

Figure 3.14 shows one example of a lane change status which is obtained by the aforementioned method.

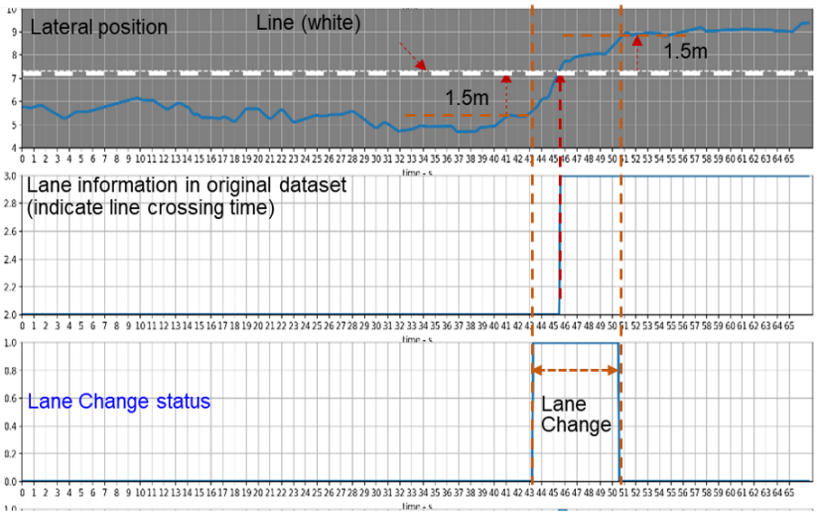


Figure 3.14: Produced lane change status information

Additionally, vehicle data including lane changes more than two are excluded, because the trajectories have a high possibility to include lane change maneuver which is required when a vehicle enters the road through an expressway ramp. Also, vehicle trajectories are divided into several segments considering six surrounding vehicles in order that one segment has same surrounding vehicles. The surrounding vehicles are front, rear, front left, rear left, front right and rear right vehicles, because information about relative maneuvers for the six surrounding vehicles can be effective to make a decision for lane change. Consequently, data used to train models are extracted in consideration of lane change status and segments information as illustrated in Figure 3.15.

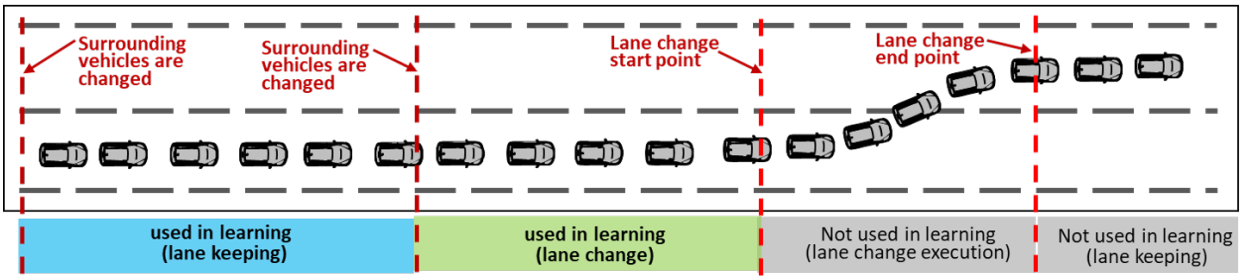


Figure 3.15: Data extraction

In other words, lane keeping data is selected as the data segment before the last segment before lane change execution start time, and lane change data is selected as the last segment before lane change execution start time. Finally, in accordance with maneuvers, the extracted data are labeled by three classes shown below. While most previous works in [74] [75] [76] [77] [78] only consider two classes (lane keeping and lane change) and did not consider lane change direction, in this work three classes including lane change direction are considered which is considered realistic.

- lane keeping
- left lane change
- right lane change

Also, relative maneuver information with respect to six neighboring vehicles is extracted as feature information to be used in training and is illustrated in Figure 3.16.

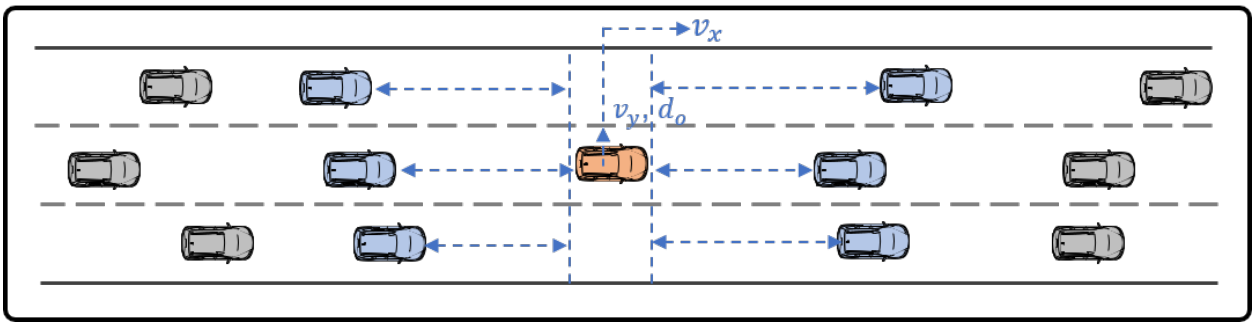


Figure 3.16: Relative distance and speed based feature definition

$$f_{rel} = (v_x, v_y, d_y, \Delta v_{rel}, \Delta x_{rel}) \tag{3.29}$$

where, $\Delta v_{rel} = \{\Delta v_{rel,i} | i = \{$*front, rear, front left, rear left, front right, rear right*$\}\}$

$\Delta x_{rel} = \{\Delta x_{rel,i} | i = \{$*front, rear, front left, rear left, front right, rear right*$\}\}$

where, v_x is the longitudinal speed, v_y is lateral speed, d_y is a distance to lane center, $\Delta v_{rel,i}$ and $\Delta x_{rel,i}$ are relative speed and distance with respect to surrounding vehicles respectively.

In this study, LSTM and GRU type of recurrent neural network, and their combined structure with SVM are used to design the lane change decision-making model. Several previous researchers have been trying to introduce SVM in a neural network architecture [81][82]. In [82], softmax in the final output layer of a GRU model is replaced with linear SVM, and this idea is adopted in this research. By introducing the SVM in the final output layer, the combined model use the SVM loss instead of cross entropy loss. In [82], L2 type of SVM loss is used rather than soft margin SVM,

because soft margin SVM is not differentiable. L2 type of SVM loss is expressed by

$$\min_{\mathbf{w},b} \frac{1}{2}\|\mathbf{w}\|^2 + C\sum_{i}^{N} \max(0, 1 - y_i(\mathbf{w}^{\mathrm{T}}\mathbf{x} + b))^2 \tag{3.30}$$

Figure 3.17 shows the combined structure of RNNs and SVM.

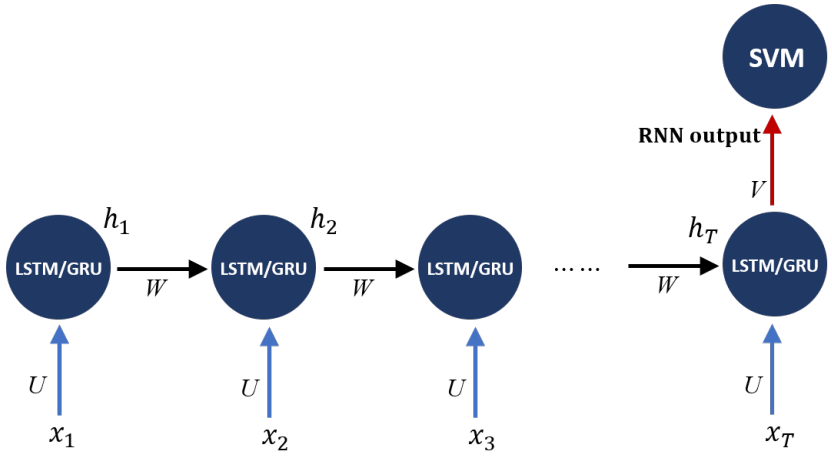


Figure 3.17: RNN models structure

In Table 3.6, four different structures of designed classifiers are summarized. Combined structure of RNNs and SVM has multilayer recurrent layers (LSTM or GRU), linear layer and linear SVM successively. The number of features in the hidden state of LSTM/GRU is defined as 512 and the number of hidden layers in LSTM/GRU is chosen as 3.

In addition, a conventional SVM classifier is designed and compared as the baseline scheme. Figure 3.18 shows learning curves of four different types of RNNs based models, and validation accuracies of each model are compared in Table 3.7. From the results, it is found that RNNs based models have better performance than conventional SVM model.

Model	Structure	RNNs Parameters
LSTM	Multilayer LSTM + Linear layer + Softmax	hidden size: 512 number of layers: 3 sequence length: 10
LSTM+SVM	Multilayer LSTM, + Linear layer + Linear SVM	hidden size: 512 number of layers: 3 sequence length: 10
GRU	Multilayer GRU + Linear layer + Softmax	hidden size: 512 number of layers: 3 sequence length: 10
GRU+SVM	Multilayer GRU, + Linear layer + Linear SVM	hidden size: 512, number of layers: 3 sequence length: 10

Table 3.6: RNNs Model Structure

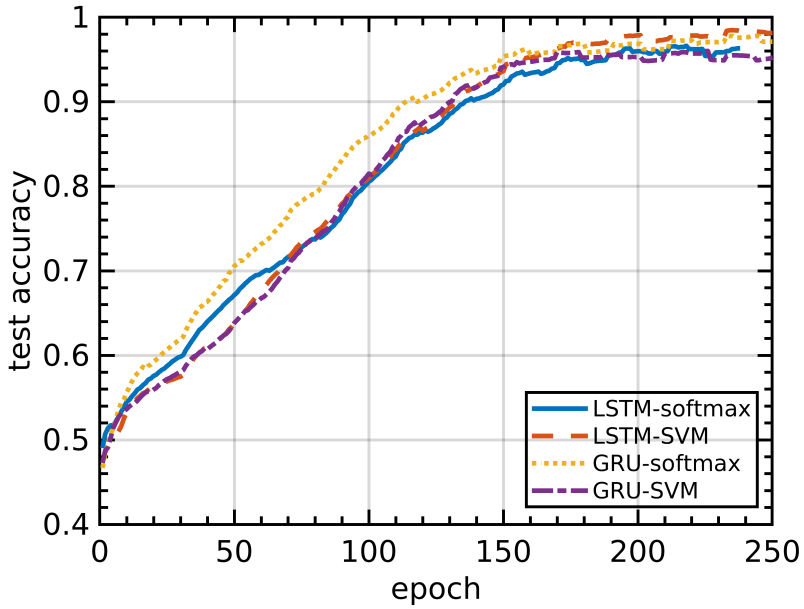


Figure 3.18: Leaning curve - loss vs epoch (NGSIM)

Model	Test Accuracy
SVM	0.8549
LSTM+softmax	0.9836
LSTM+SVM	0.9719
GRU+softmax	0.9789
GRU+SVM	0.9844

Table 3.7: Comparison of validation accuracy

3.3.3.3 Longitudinal behavior model

IDM is a commonly used human driver's longitudinal behavior model for traffic simulation. IDM has several parameters as specified in Table 3.1. But the nominal values of IDM parameters in Table 3.1 don't represent real drivers behaviors. Therefore, IDM parameters are calibrated using NGSIM data by particle swarm optimization (PSO) which is a metaheuristic algorithm inspired by the concept of swarm (group) intelligence [83]. PSO is a powerful optimization tool to solve complicated mathematical problems. In PSO, a group of particles are considered, and each particle is dealt as a candidate solution and it is characterized by a position vector and velocity vector. The particles are updated as candidate solutions during optimization process to minimize or maximize a cost function which is also called as fitness function. Each particle keeps its best parameter (position vector) during optimization and the global best parameters are also saved among the entire population of particles. Basic PSO algorithm is specified in Algorithm 9.

The history of global cost is shown in Figure 3.19, and calibration results of IDM parameters are given in Figure 3.20. The calibrated parameters are distributed as illustrated in in Figure 3.20 and representative values are chosen as Table 3.8 for performance comparison with other decision models.

parameter	nominal value
desired speed v_0	27 m/s
desired time gap T	0.5 s
jam distance s_0	7.0 m
maximum acceleration a	1.0 m/s^2
desired deceleration b	6.5 m/s^2

Table 3.8: Calibrated IDM parameters

Algorithm 9 PSO algorithm [84]

Initialize N numbers of particles
- Randomize position vector x_i within parameter boundaries
- Set velocity vector v_i as zeros for particle i
- Set the best position vector for each particle as $pBest_i = x_i$
- Set global minimum cost, $J_{best} = \infty$

Minimize the cost function, $J = f(x)$
for iteration$= 1, M$ **do**
  for particle $i = 1, N$ **do**
    Compute current cost $J_i = f(x_i)$ of particle i
    Save the best parameters of particle i as $pBest_i$
      if $J_i < f(pBest_i)$, ***then*** $pBest_i = x_i$
    Save global best parameters $gBest$ and global minimum cost as J_{best}
      if $f(pBest_i) < J_{best}$, ***then*** $gBest = pBest_i$, $J_{best} = f(gBest)$
  end for
  for particle $i = 1, N$ **do**
    Update particle i's position and velocity vectors
      $x_i \leftarrow x_i + v_i$
      $v_i \leftarrow wv_i + c_1 rand_{(0,1)}(pBest_i - x_i) + c_2 rand_{(0,1)}(gBest - x_i)$
  end for
end for

3.3.4 Performance comparison

In this comparison studies, four different types of decision models are compared. For each decision model, 1000 simulations were conducted in highway driving situations with randomly generated vehicles to setup realistic highway driving environments. In driving situations, higher speed is preferred and a collision should be prevented. Therefore, performance of decision models can be evaluated by mean speed and moving distance of ego-vehicle per simulation. Mean speed can be regarded as a criterion for evaluating a performance with respect to higher speed. On the other hands, moving distance can be considered as a criterion for performance in terms of collision, because a simulation without a collision results in longer moving distance of the ego-vehicle. Therefore, mean speeds and moving distances were collected during simulations for the

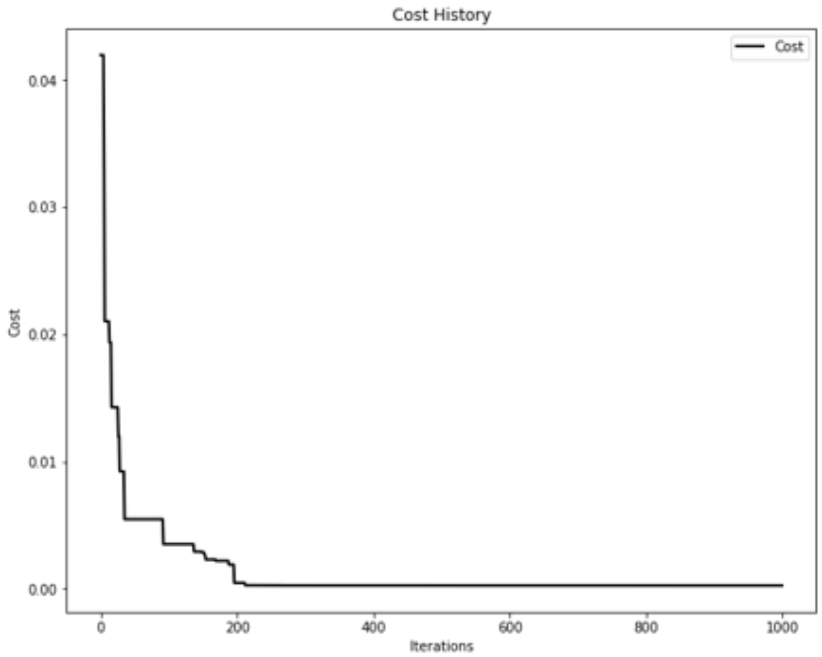


Figure 3.19: History of cost during optimization

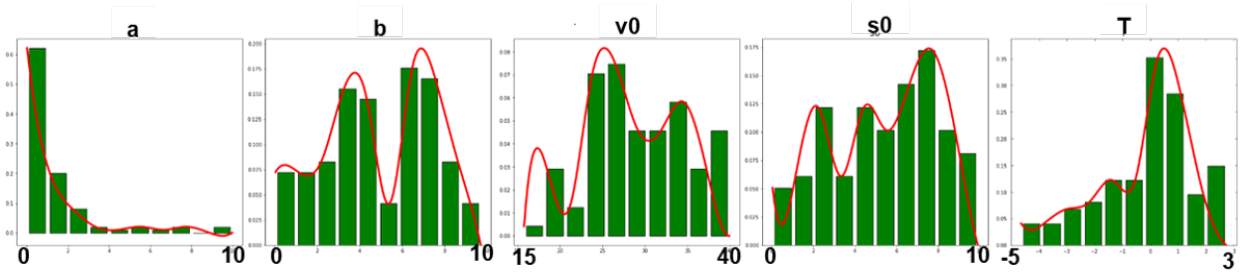


Figure 3.20: Distribution of calibrated IDM parameters

purpose of performance comparison. The comparison results are shown in Figure 3.21 for mean speed and in Figure 3.22 for moving distance. As shown in Figure 3.21 and 3.22, RL based decision model has the best performance. Namely, RL based model shows higher speed of ego vehicle than other models and moving distance is also the longest among the considered four types of decision models. The other models show similar level of performance and it is observed that rule-based decision model is slightly better than IDM and MOBIL based human driver behavior model and data-driven model.

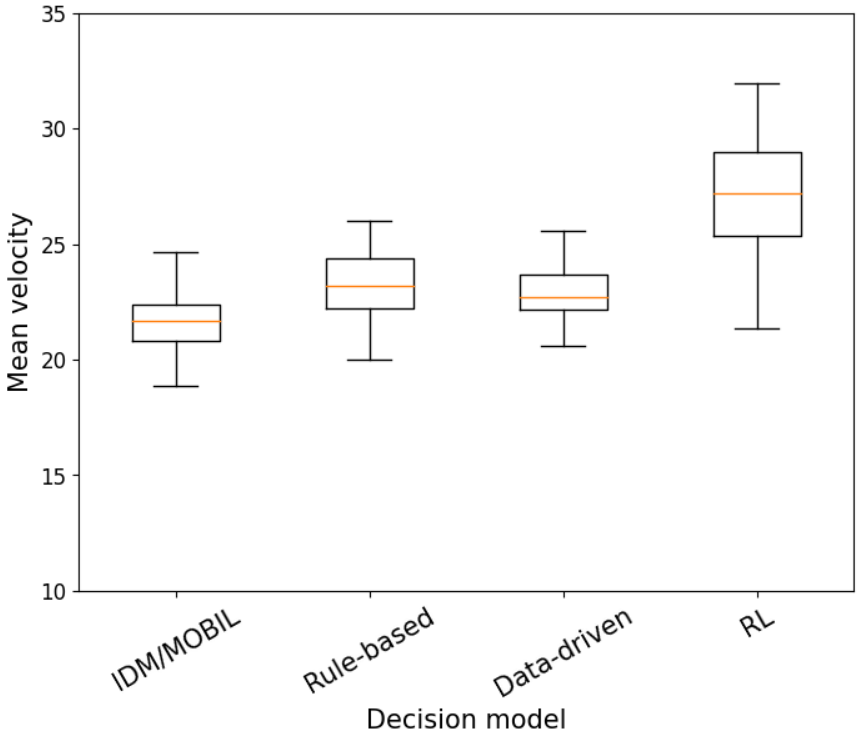


Figure 3.21: Performance comparison - mean speed

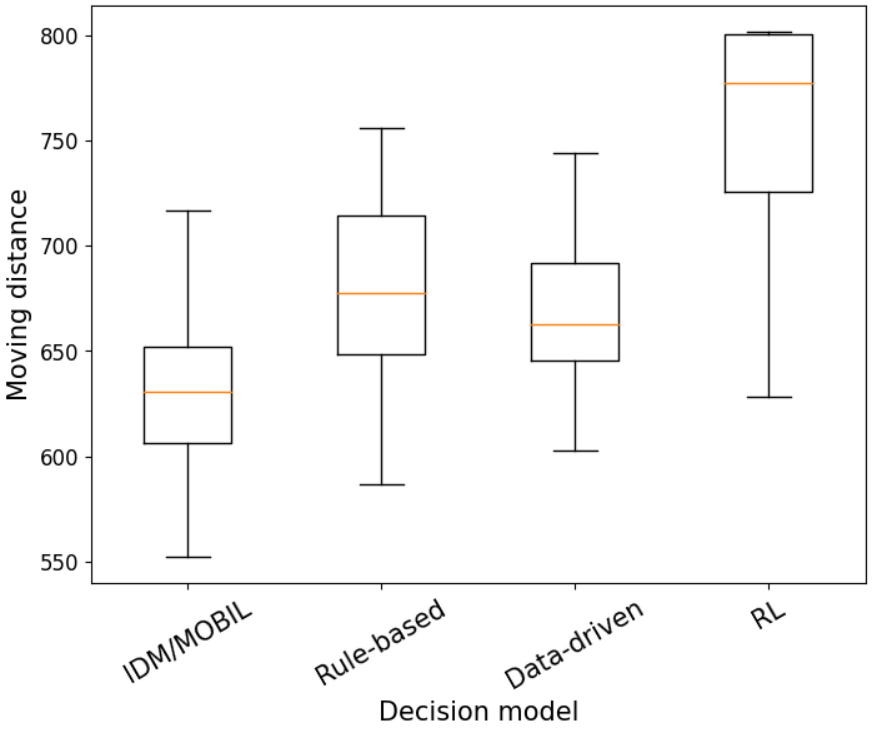


Figure 3.22: Performance comparison - moving distance

3.4 Conclusions

In this research, RL based decision making for self-driving is studied. Firstly, RL based decision making models are designed for highway driving problem. The highway driving environment is defined as an MDP framework for RL problem. The three different types of state spaces are defined such as relative maneuver based, surrounding gap based and occupied grid state space definition. Also, reward functions consisting of three sub-rewards and high-level action space with 5 elements are defined. Several DQN based algorithms are applied for the problem. The RL algorithms successfully trains the agents and produces decision making model based on neural networks. The three different state spaces show the similar performance and it means defined state spaces represent real driving situations appropriately. Comparison studies with other decision models, such as traditional human driver behavior model, rule based model and data-driven model using real-world dataset, presents RL based model has better performance than the other models.

4. APPLICATION OF REINFORCEMENT LEARNING BASED DECISION MODEL WITH SHARED CONTROL

In this section, RL based decision model is applied with shared control in the highway driving environment. In Sections, 3 and 2, design of low level vehicle controller based on game theoretic MPC and development of decision model based on RL were the main purposes, but a combinatorial study of game theoretic MPC based shared controller and RL based decision model is focused in this section. That is, RL based decision model is used for the machine's decision model and the game theoretic MPC based shared controller is used as a low level vehicle controller in this study. To emulate human driver's decisions, human driver models (IDM/MOBIL) are used as a human driver's decision model. Accordingly, a more realistic simulation configuration is achieved in this study in consideration of human driver' behaviors by human driver models and machine's behavioral decision model by well-developed RL based decision model. Figure 4.1 shows a hierarchical structure which is considered in this study. Behavioral decision model, path planner and low level controller are composed under a hierarchical structure. Data acquisition and processing systems are neglected in this research and it is assumed that high-level driving information is given. Path planner generates target vehicle trajectory corresponding to a behavioral decision from the decision model. Path planning is beyond the scope of this research. Several path planning methods have been proposed such as spline curve based methods [85][86][87] or polynominal spiral based methods [88][89]. As such the boundary spline based planning method in [87] is used in this research. In this study, it is focused on application of RL based decision model with game theoretic shared controller and evaluation of its functionality in terms of decision and control authority in traffic situations. Therefore, kinematic vehicle model in (3.20) is used with the purpose of computation efficiency during simulations. Game theoretic MPC based controller in Section 2, is applied as low-level vehicle controllers in consideration of human driver and machine as two game players.

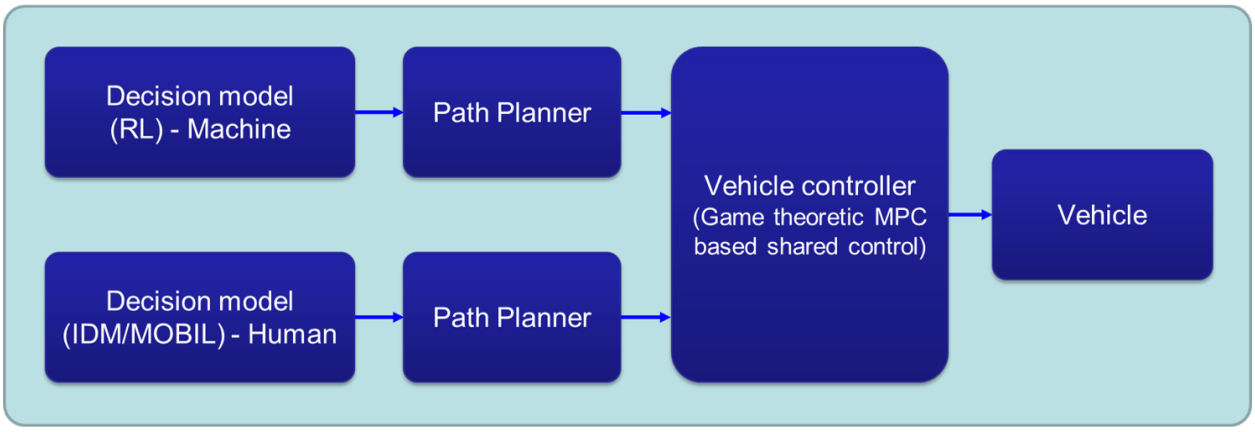


Figure 4.1: Simulation setup with Hierarchical architecture

4.1 Simulation studies

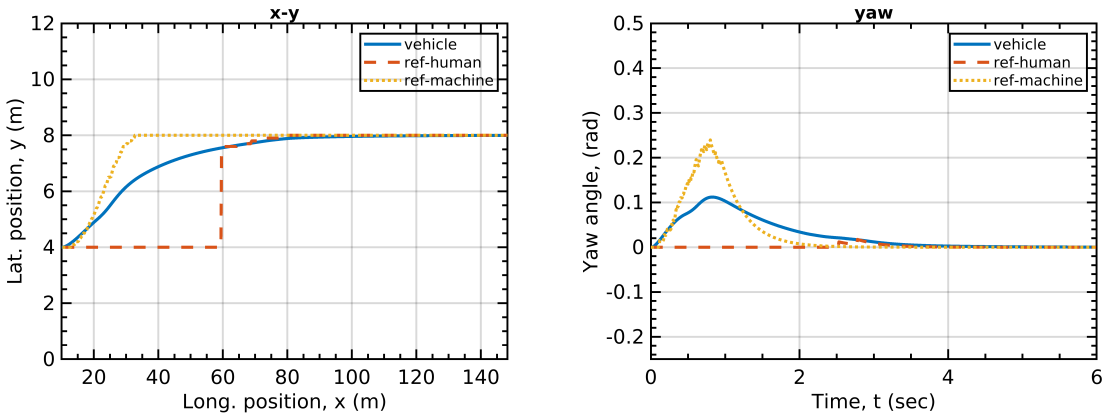
Several simulation scenarios are considered. Firstly, lane change scenario to avoid a collision with a front stopped vehicle is investigated. In this scenario, four simulations were done with different settings of simulation regarding game type and human driver's intention. In addition, more complicated traffic driving scenarios were conducted with randomly generated other vehicles in highway driving environment with different simulation setups.

4.1.1 Lane change scenario with a front stopped vehicle

In this scenario, four simulations were conducted with a front stopped vehicle with the assumption that the human driver does not recognize the stopped vehicle. In this scenario, the lane change action is expected to avoid the front vehicle.

4.1.1.1 Under cooperative game framework

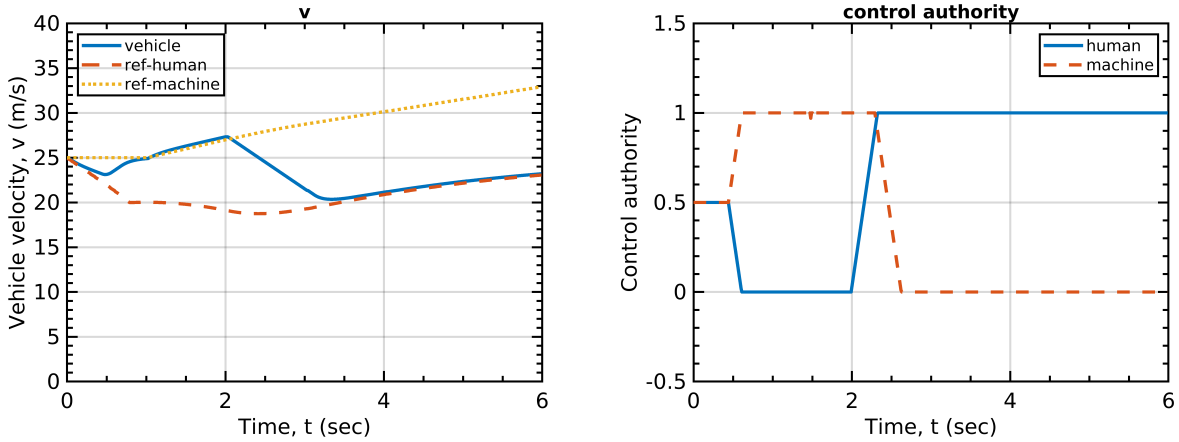
The first simulation was conducted under a cooperative game framework. This means that the human driver cooperates with the machine even though he does not make the same decision as the machine. After the lane change is successfully completed according to the machine's decision, the human driver has the intention to control the vehicle and therefore the machine returns the control authority to the human. Figures 4.2 and 4.3 below show the simulation results.



(a) Vehicle position (x-y)

(b) Yaw

Figure 4.2: Simulation results - lane change with a stopped vehicle under cooperative game



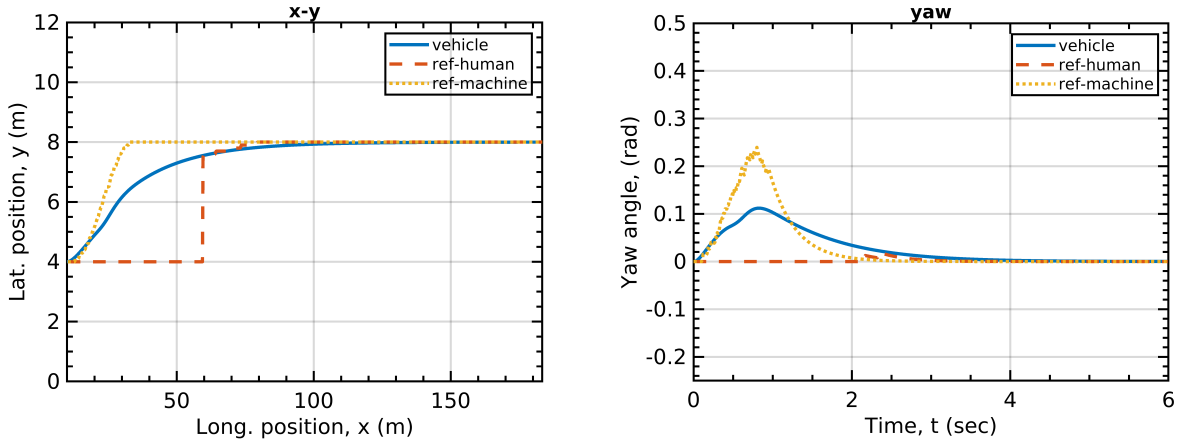
(a) Vehicle velocity

(b) Control authority

Figure 4.3: Simulation results - lane change with a stopped vehicle under cooperative game

As shown in Figures 4.2 and 4.3, the human driver reduces his/her control authority to the follow machine's decision, even though the vehicle maneuver does not follow his/her demand. After the lane change is successfully completed, the human driver increases his/her control authority again to control the vehicle by his/her own decision, and the machine reduces its control authority to allow the human driver to control the vehicle.

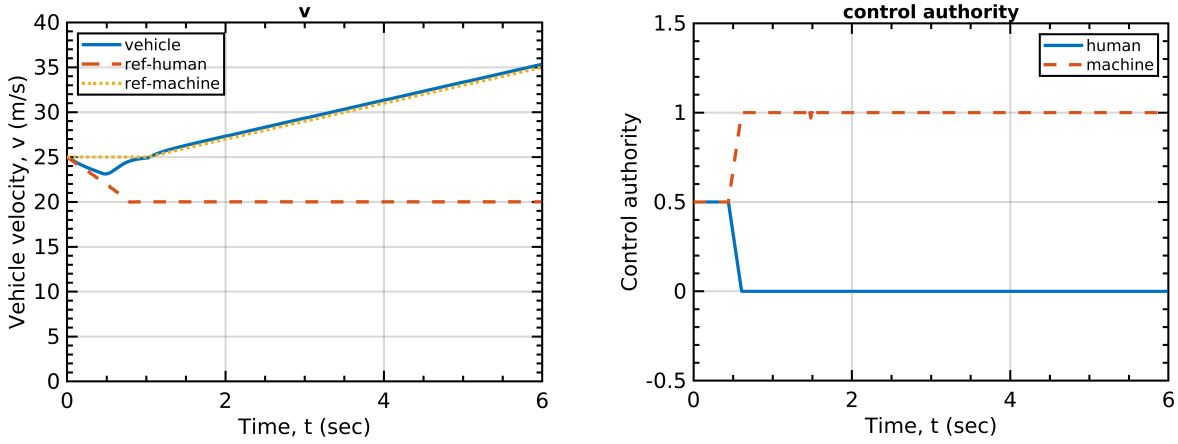
In addition, another simulation was conducted with the assumption that the human driver continues to follow the machine after the lane change. In this case, it is expected that the machine does not reduce its control authority even though the lane change is completed. Figures 4.4 and 4.5 below show the simulation results.



(a) Vehicle position (x-y)

(b) Yaw

Figure 4.4: Simulation results - lane change with a stopped vehicle under cooperative game



(a) Vehicle velocity

(b) Control authority

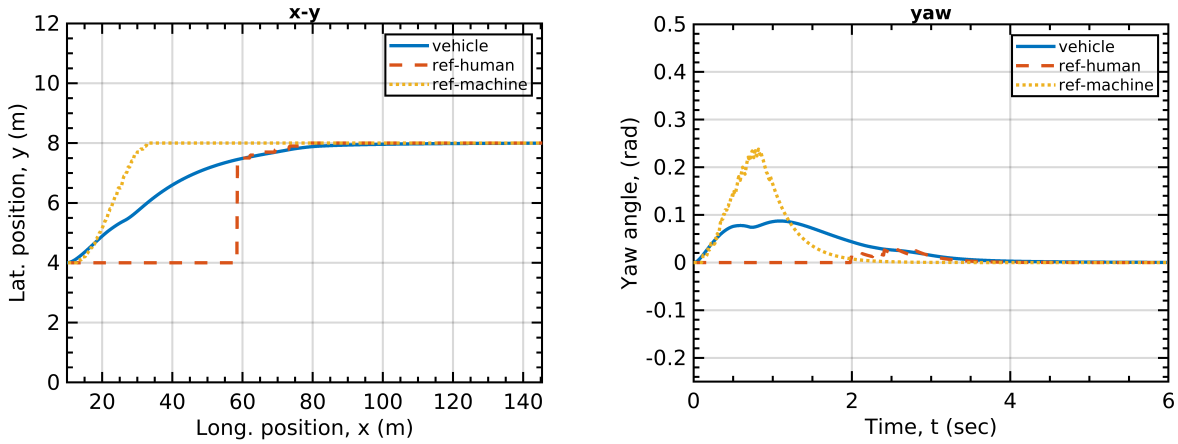
Figure 4.5: Simulation results - lane change with a stopped vehicle under cooperative game

As shown in Figures 4.4 and 4.5, the human driver reduces his/her control authority to follow the machine's decision and keeps following the machine' decision after the lane change. On the other hands, the machine increases its control authority to avoid a collision and maintains its control authority in order to continue to control the vehicle even after the lane change is completed.

4.1.1.2 Behavior under the game transition framework

Same simulation scenario was considered with game transition. Firstly, the human driver and the machine play their game in cooperative mode but the human driver does not cooperate with a machine when his/her decision is different from the machine's decision. Therefore, it is expected that the game type must be changed to non-cooperative and eventually the machine needs to take all control authority to avoid a dangerous situation. Figure 4.6, 4.7 and 4.8 below show the simulation results.

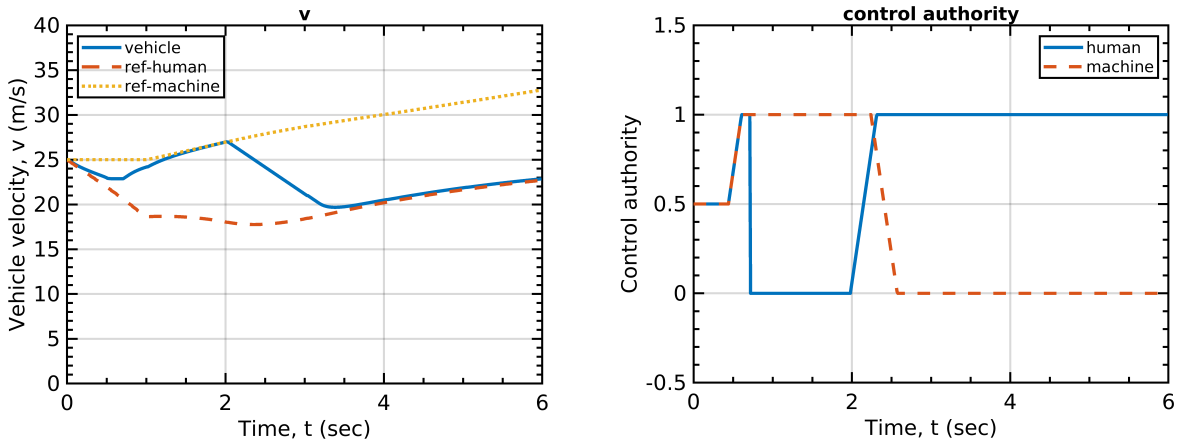
From Figure 4.8, it is found that the game type is changed according to the driving situation and finally the machine takes all control authority. It is assumed that the human driver wants to keep driving in the current lane after the lane change. After the lane change is completed, the human driver increases his/her control authority again to control the vehicle by himself/herself, and the machine reduces its control authority.



(a) Vehicle position (x-y)

(b) Yaw

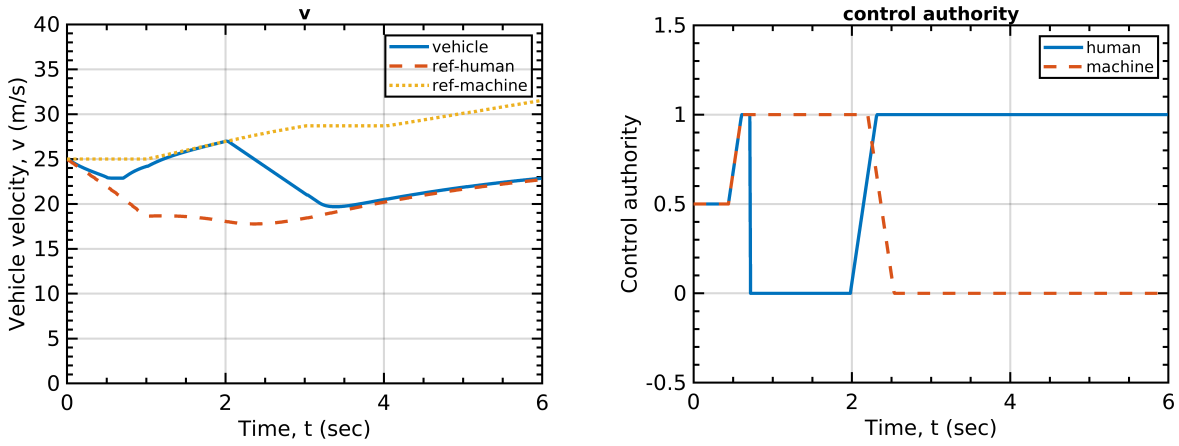
Figure 4.6: Simulation results - lane change with a stopped vehicle under game transition



(a) Vehicle velocity (b) Control authority

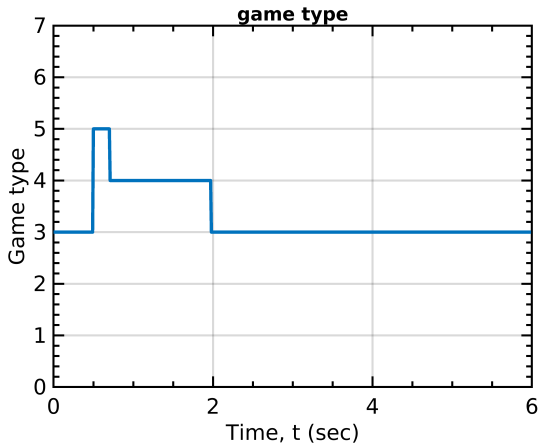
Figure 4.7: Simulation results - lane change with a stopped vehicle under game transition

In addition, another simulation was conducted but it is presumed that the human driver wants to return to his/her original target lane after the lane change. Figures 4.9, 4.10 and 4.11 below show the simulation results. The simulation results are the same as the previous results but the ego-vehicle returns to the human driver's original target lane after the lane change is completed to avoid the front vehicle.



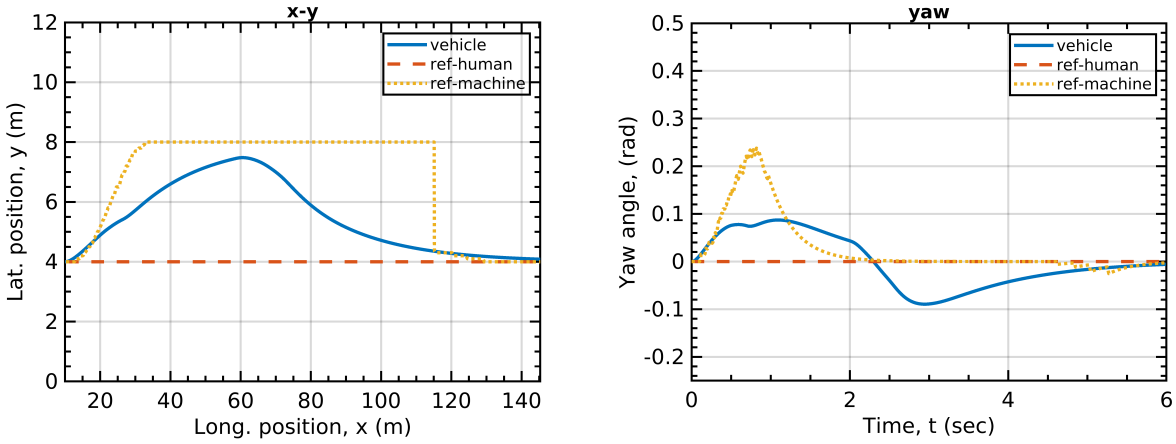
(a) Vehicle velocity (b) Control authority

Figure 4.10: Simulation results - lane change with a stopped vehicle under game transition



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

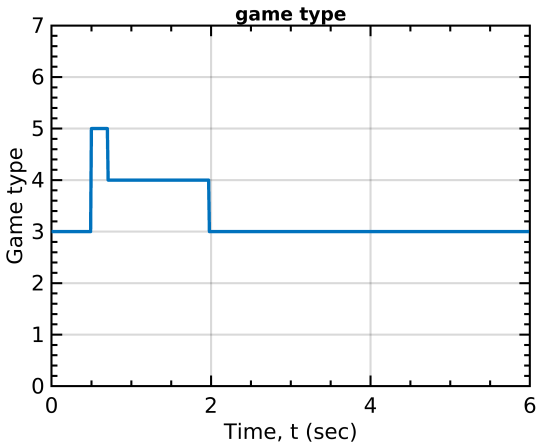
Figure 4.8: Simulation results - lane change with a stopped vehicle under game transition



(a) Vehicle position (x-y)

(b) Yaw

Figure 4.9: Simulation results - lane change with a stopped vehicle under game transition



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

Figure 4.11: Simulation results - lane change with a stopped vehicle under game transition

4.1.2 Traffic scenarios in highway driving

In these scenarios, four simulations were conducted in highway driving with randomly generated other vehicles. In the first two simulations, it is assumed that the human driver is careless and cannot properly deal with the expected driving situation. Therefore, it is required that the machine handle the diving situation to avoid a dangerous situation. On the other hand, in the other two simulations, it is assumed that the human driver acts incorrectly to lead to a collision with other vehicles. The machine needs to adjust its control authority depending on the driving situation and the driver's behavior, and it is examined in these simulations that the control authority is properly shared according to the diving situation to prevent falling into a dangerous condition.

4.1.2.1 Traffic scenarios with a careless driver

Two simulations were conducted with a careless driver. In these simulations, it is presumed that the human driver of the ego-vehicle does not recognize certain situations so as to make a suitable decision. In the first simulation, a lane change is required for the ego-vehicle and the human driver and the machine make the decision for lane change, but a front vehicle also makes a decision to change lanes to the same lane as the ego-vehicle. In this situation, a lane change to the previous

lane is required, but the human driver does not deal with this situation well and does not control the vehicle to return to the previous lane. Therefore, it is required that the machine should control the vehicle to handle this situation in order to avoid a dangerous condition. The simulation scenario is briefly illustrated in Figure 4.12.

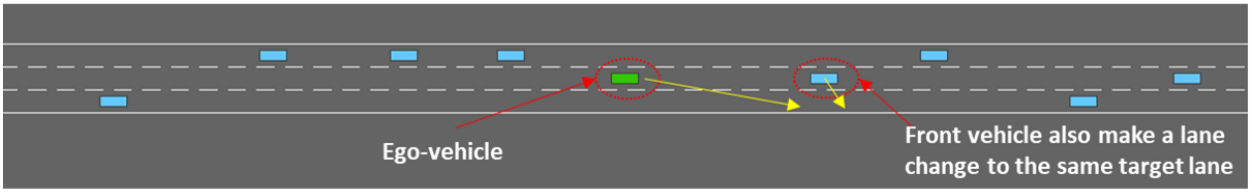
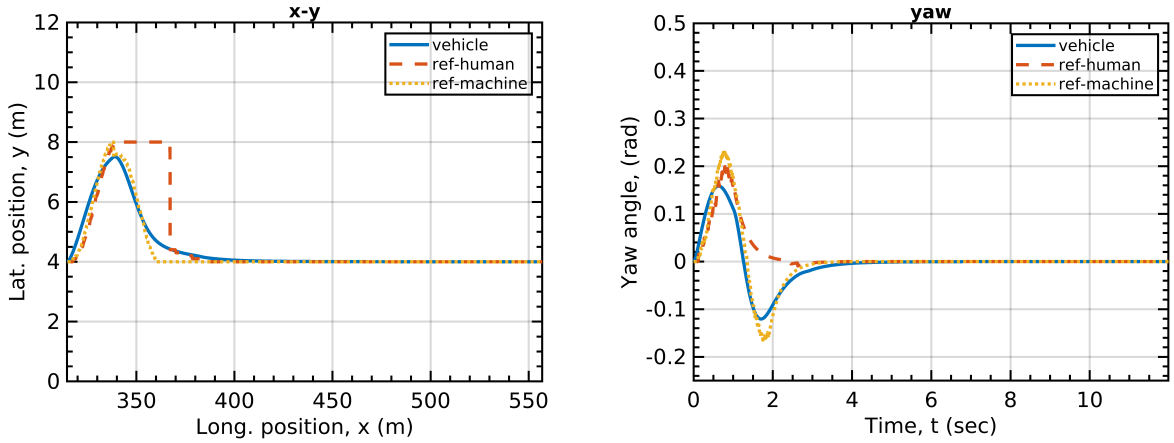


Figure 4.12: Traffic scenario 1 with a careless driver

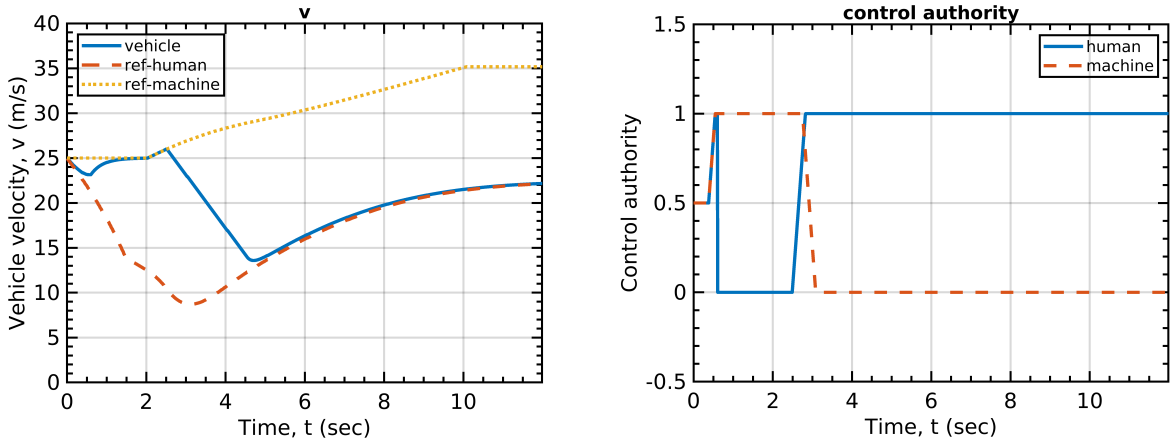
The simulation starts under a cooperative game and the human driver follows his/her own decision when there is a conflict between the human's and the machine's decisions. Figures 4.13, 4.14 and 4.15 show the simulation results.



(a) Vehicle position (x-y)

(b) Yaw

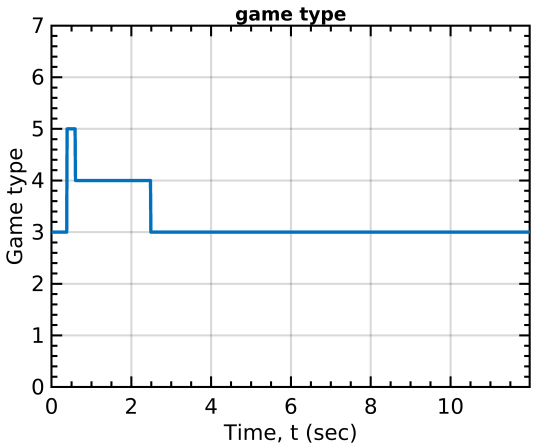
Figure 4.13: Simulation results - Traffic scenario 1 with a careless driver



(a) Vehicle velocity

(b) Control authority

Figure 4.14: Simulation results - Traffic scenario 1 with a careless driver



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

Figure 4.15: Simulation results - Traffic scenario 1 with a careless driver

From Figures 4.13, 4.14 and 4.15, it is observed that the machine increases its control authority after the first lane change and the game is changed from cooperative to non-cooperative, because the human driver also keeps its control authority to control the vehicle by himself. Due to the conflict between the human and the machine, the machine removes the human driver's action and takes all control authority when the collision risk become high. After the ego-vehicle returns to the previous lane and the driving situation is safe, the game is transitioned to the cooperative mode and the machine reduces its control action, as the human driver increases his/her control authority.

In addition, another simulation was performed in the condition that a front left vehicle suddenly gets into the lane of the ego vehicle to block the ego-vehicle. In this situation, lane change is desired but it is assumed that the human driver does not recognize this situation. Therefore, the machine should control the vehicle to deal with this situation by lane change instead of the human driver. The simulation scenario is illustrated in Figure 4.16 and the simulation results are shown in Figures 4.17, 4.18 and 4.19.

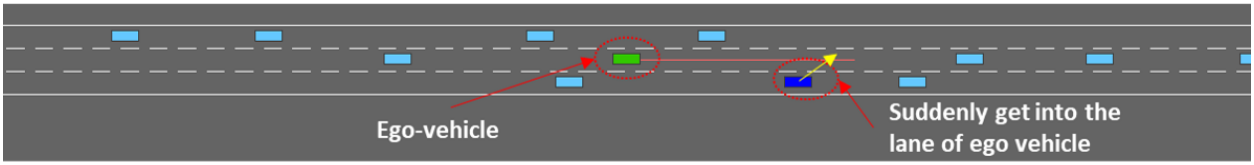
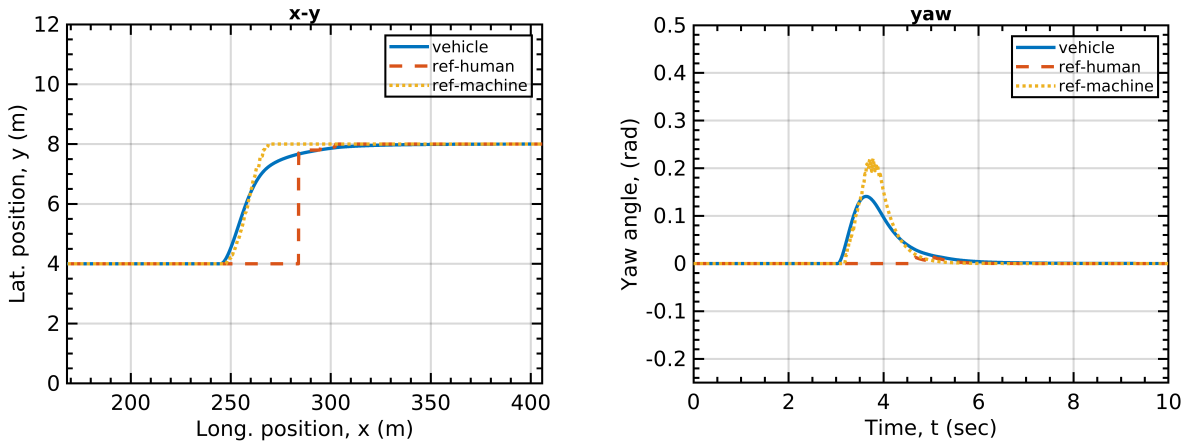


Figure 4.16: Traffic scenario 2 with a careless driver

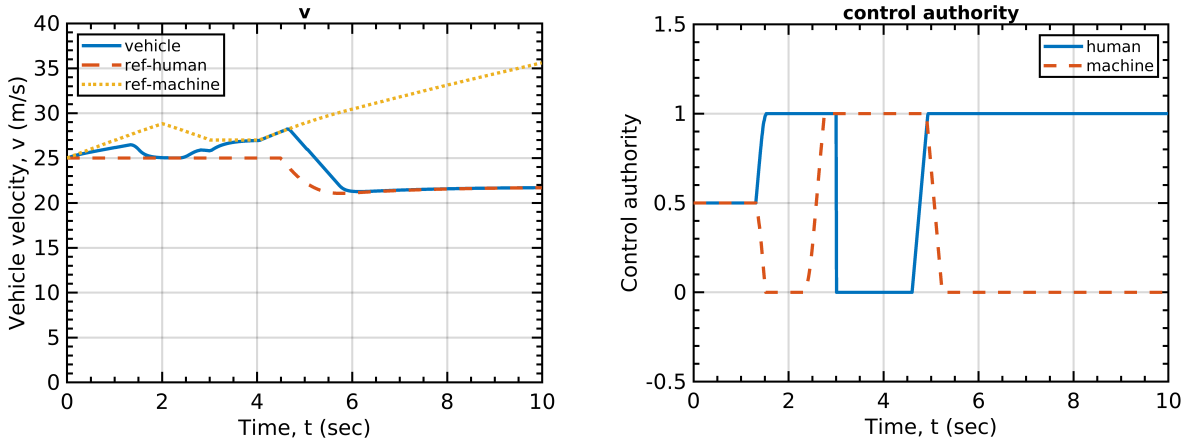
From Figures 4.17, 4.18 and 4.19, it is found that the machine makes a decision to change lanes after the front left vehicle suddenly gets into the same lane and blocks the ego-vehicle. Because the human driver's action is to keep driving in the current lane, their game is changed from cooperative to non-cooperative , due to conflict between two players. Their conflict continues in a non-cooperative game and the machine eventually has to fully control the vehicle by itself. After successfully changing lanes to left, the game is returned to the cooperative mode and the machine lowers its control authority as the human driver starts to control the vehicle by increasing his/her



(a) Vehicle position (x-y)

(b) Yaw

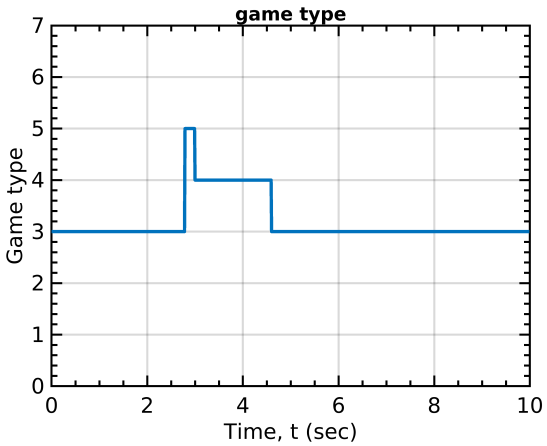
Figure 4.17: Simulation results - Traffic scenario 2 with a careless driver



(a) Vehicle velocity

(b) Control authority

Figure 4.18: Simulation results - Traffic scenario 2 with a careless driver



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

Figure 4.19: Simulation results - Traffic scenario 2 with a careless driver

control authority.

4.1.2.2 Traffic scenarios with incorrect actions of the driver

In real driving situations, the human driver may actively take an action but his/her action maybe incorrect. As a result, the human driver's incorrect action can lead to a dangerous driving condition. These traffic scenarios are aimed as investigating the aforementioned situation caused by incorrect driver's action. Two simulations were conducted with an incorrect action of the human driver.

In the first simulation, the human driver intends to make a lane change to the right but there exists a front vehicle in the right lane. Accordingly, the human driver's undesired action leads to a rapid motion towards the front right vehicle. Hence, the machine has to take control authority to handle the situation. The simulation scenario is illustrated in Figure 4.20 and simulation results are given in Figures 4.21, 4.22 and 4.23.

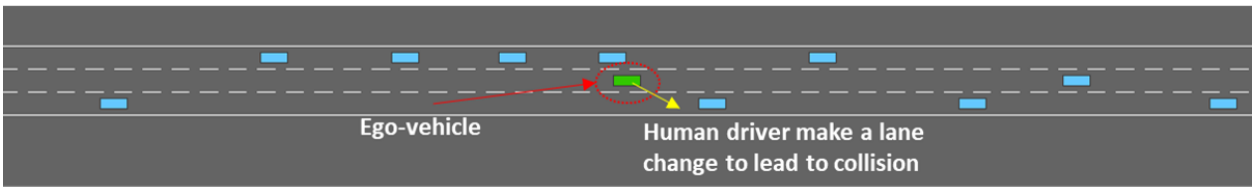
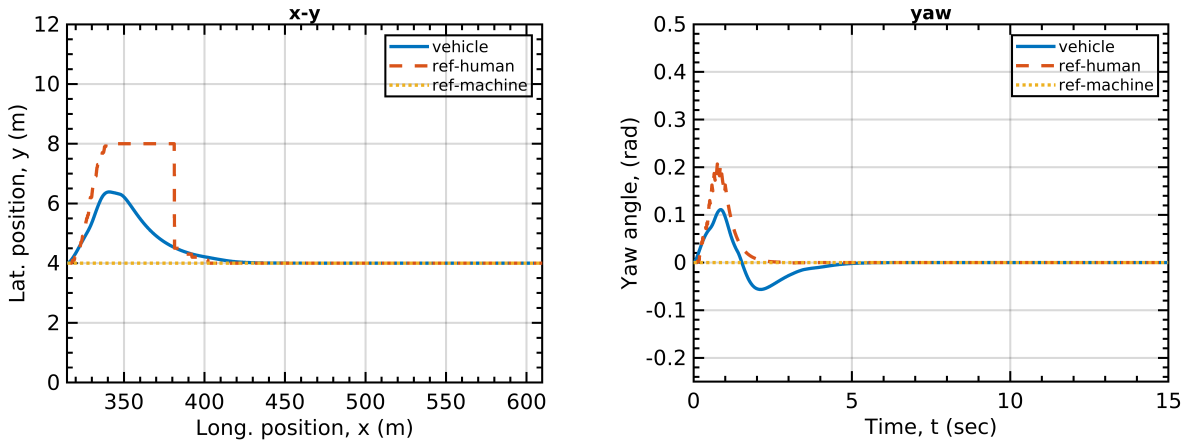


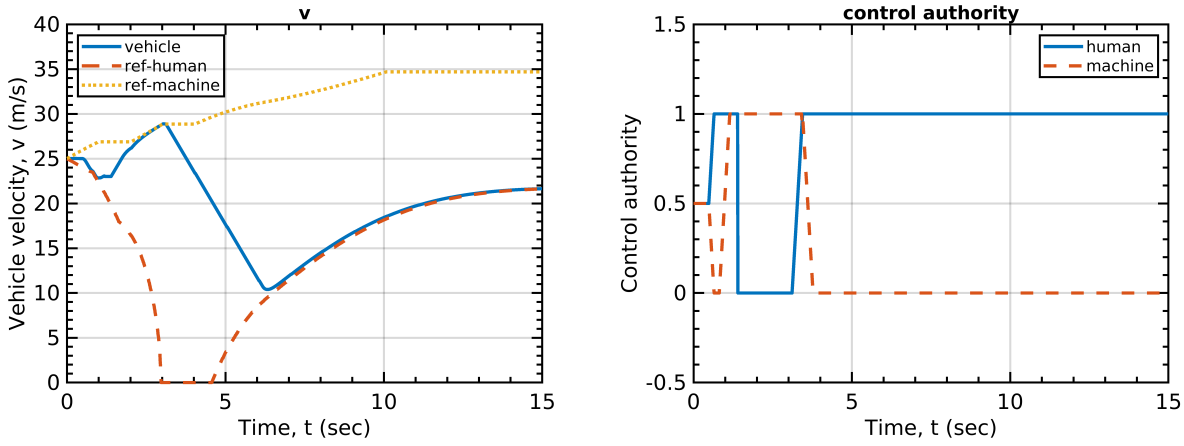
Figure 4.20: Traffic scenario 1 with incorrect actions of a driver



(a) Vehicle position (x-y)

(b) Yaw

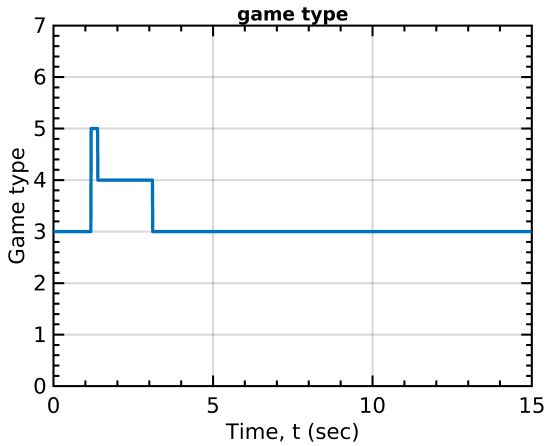
Figure 4.21: Simulation results - Traffic scenario 1 with incorrect actions of a driver



(a) Vehicle velocity

(b) Control authority

Figure 4.22: Simulation results - Traffic scenario 1 with incorrect actions of a driver



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

Figure 4.23: Simulation results - Traffic scenario 1 with incorrect actions of a driver

As shown in Figure 4.21, 4.22 and 4.23, the machine prevents a lane change by the human driver toward the front right vehicle through increasing its control authority. The game is changed to non-cooperative mode during the lane change and then finally transitioned to fully autonomous driving mode to avoid a collision with the front right vehicle. After the collision risk disappears, the game is return to cooperative mode and the human driver regains all control authority.

In the second simulation, undesired human driver's longitudinal action is considered. The human driver undesirably produces full acceleration as a longitudinal action, even though a front vehicle exists in close proximity. Therefore, the machine should take proper action to deal with this situation to be safe. The simulation scenario is illustrated in Figure 4.24 and Figure 4.25, 4.26 and 4.27 show simulation results.

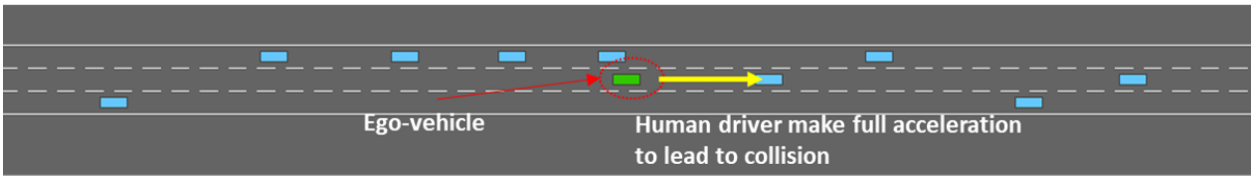
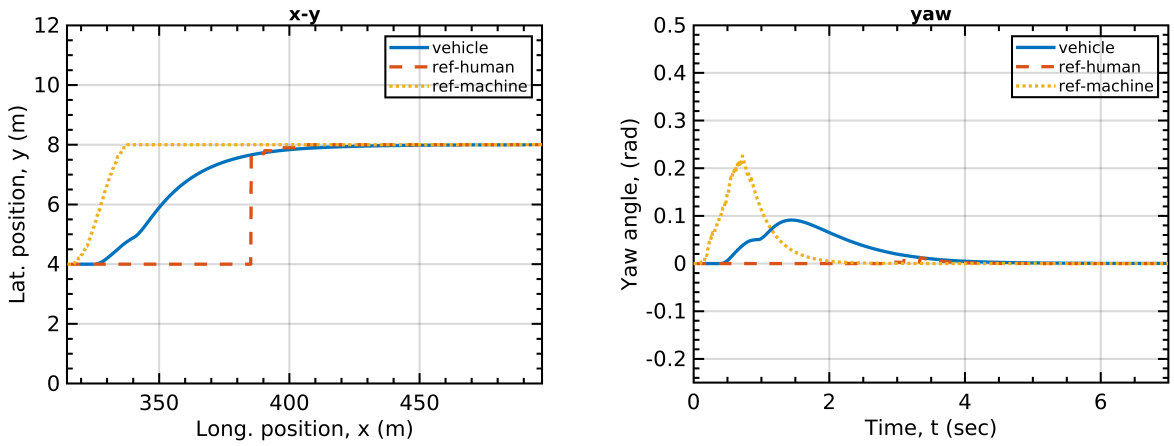


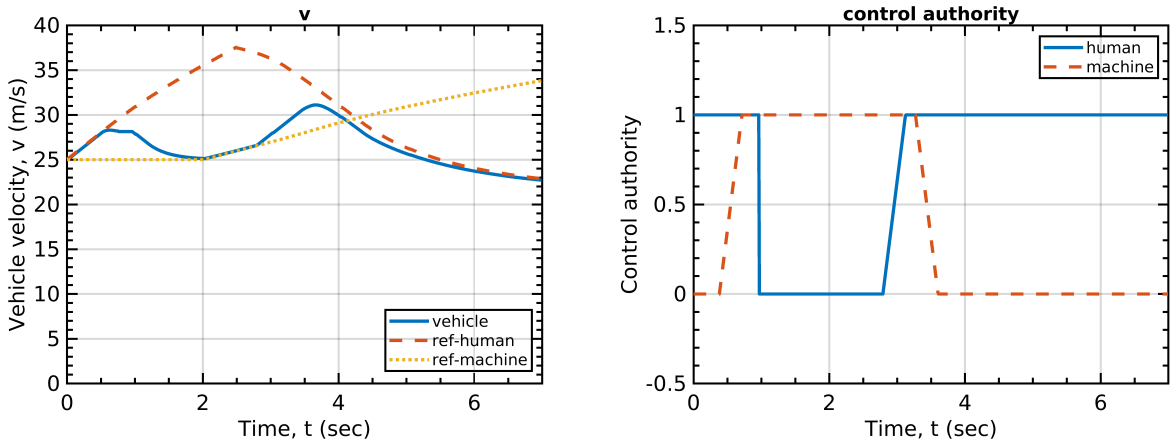
Figure 4.24: Traffic scenario 2 with incorrect actions of a driver



(a) Vehicle position (x-y)

(b) Yaw

Figure 4.25: Simulation results - Traffic scenario 2 with incorrect actions of a driver

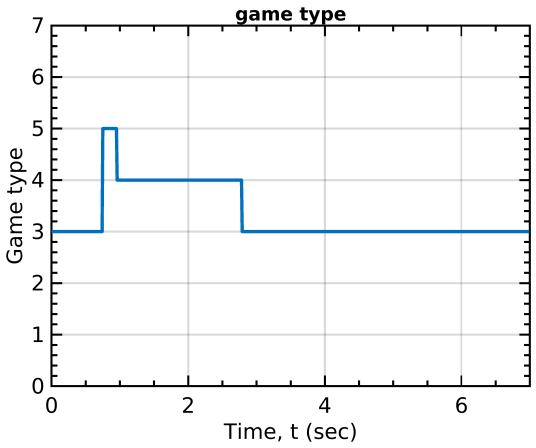


(a) Vehicle velocity

(b) Control authority

Figure 4.26: Simulation results - Traffic scenario 2 with incorrect actions of a driver

In Figure 4.25, 4.26 and 4.27, it is found that the machine makes a decision for lane change and comes into conflict with the human driver. Due to their conflict, the game is transitioned to non-cooperative mode and finally the machine fully controls the vehicle to overcome a situation with high collision risk. After that, the ego-vehicle returns to a safe driving situation and the game



(a) Game type (3: cooperative, 5: non-cooperative, 4: fully autonomous mode)

Figure 4.27: Simulation results - Traffic scenario 2 with incorrect actions of a driver

is changed to cooperative mode. Finally, the machine reduces its control actions, as the human driver intends to control the vehicle by himself through increasing his control authority.

4.2 Conclusion

In this study, an RL based decision model is applied with game theoretic MPC based shared controller. In a hierarchical structure, the RL based decision model, path planner and shared controller is incorporated. Several simulations in lane change scenarios and traffic scenarios were conducted to evaluate performance of the RL based decision model and the shared controller. From the simulation results, it is observed that the ego-vehicle can avoid dangerous situations by the machine's action when the human driver produces incorrect actions causing dangerous situations or does not take a proper action. In other words, the machine is capable of changing its control authority to cooperate or compete with the human driver as circumstances demand in order to keep vehicle safe in safety-critical conditions and not to interrupt the human driver in the condition that human driver's decision does not correspond to the machine's decision but condition is not safety-critical.

5. SUMMARY AND CONCLUSIONS

In this research, reinforcement learning based decision making for self-driving and shared control human driver and machine are studied.

First of all, the game theoretic model predictive control (MPC) based shared controller is proposed with four different types of games. To deal with the nonlinearity of the vehicle dynamics including nonlinear tire characteristics, a successive linearization technique is applied to establish a linearized system at an operating point. Also, the game transition model is designed based on a finite state machine (FSM) framework, and a shared control strategy is proposed to adjust the control authority in consideration of collision risk and tracking error. The proposed shared controller was evaluated in several simulation scenarios and it is shown that machine is able to properly change its control authority to cooperate with the human driver in non-safety critical conditions and to compete with a human driver in safety-critical conditions. In other words, machine takes more control authority when the driving situation is dangerous but reduces its control authority in condition without collision risk.

The second part of this research is a study on reinforcement learning (RL) based decision making model. To train the learning agents, highway driving environments were established with appropriate state space, action space and reward functions. Three different types of state spaces are defined as relative maneuver based, surrounding gap based and occupancy grid based. The action space is defined via five behavioral actions for highway driving, and the reward function is defined to encourage the RL agents to have a high vehicle speed without a collision during the driving. Deep Q Networks (DQN) based RL algorithms was applied to train the learning agents and simulations were performance with the trained agents. The performance of the RL based decision model is compared to other decision models which are conventional human driver model, rule-based model and data-driven decision model. The comparison studies show the RL based decision model has the best performance and the ego-vehicle is able to travel a longer distance with higher vehicle speed.

Finally, the application of RL based decision models with shared control was investigated in the highway driving environment. An RL based decision model, a path planner and a game theoretic MPC based shared controller were integrated in a hierarchical structure, and simulations were conducted in lane change scenario and traffic simulation scenarios. The simulation results shows that the machine is capable of handling its control task according to the human driver's action in consideration of the vehicle driving conditions. Namely, the machine is able to cooperate or not cooperate with the human driver by adjusting its control authority depending on the driving situation. In dangerous situations, the machine takes all control authority to prevent a collision but reduces its control action in non-dangerous situations to follow the human driver's decision when the human driver has a different decision in mind.

In conclusion, this research strives to contribute on understanding the interaction between the human driver and the machine based on a game theoretic MPC framework and also tries to develop a behavioral decision making models using the state-of-the-art reinforcement learning algorithms. The proposed shared controller and RL based decision model were evaluated in several simulation scenarios and their performance is verified by examining vehicle safety and performance.

5.1 Future works

In this research, highway driving were considered but other various driving situations need to be considered as future works. Also, it was assumed that the human driver behaves like an MPC based feedback controller in this study but other approaches for representing the real human drivers' characteristic need to be studied as future works.

Also, model uncertainty was not explicitly dealt with in this research although certain margins of safety were considered. The model uncertainty can be a challenge of MPC based methods. Implementation of improved safety margins with existing knowledge about the target system can be one way to handle model uncertainty, or robust model predictive control (RMPC), which is a branch of the research in MPC to deal explicitly with modeling uncertainty [90][91][92], can be another approach to treat model uncertainty. These studies about model uncertainty are expected as future works.

In this research, the nonlinearity of the model is considered by successive linearization. Simulations were conducted including severe maneuvers, which can reflect nonlinear characteristics of the system, such as lane change with high speed and full acceleration. But more systematic studies need to be investigated in regard to model nonlinearity.

Moreover, the surrounding vehicles are considered as feature information to produce a behavioral level of decision, or features are extracted using convolutional neural networks (CNNs) in this research. But more systematic studies need to be taken into account as future works to understand broader traffic situations in terms of upstream and downstream traffic in the decision-making process.

Finally, the performance of RL based models varies depending on the purpose of learning, and RL models are able to produce the optimal actions by learning under a given desired learning environment setup. In this research, the learning environment was established by the purpose for higher speed without a collision and learned RL models showed the expected performance with respect to the purpose. But the learning purposes can vary according to the demanded objectives to achieve by learning. In other words, driving efficiency in terms of vehicle speed can be more desirable in some cases and driving comfort more significant in other cases. Therefore, the optimal action can vary according to the demanded purpose and the decision model needs to be designed according to the demanded purpose. Thus, studies considering other aspects of driving need to be considered as future works.

REFERENCES

[1] M. Harris, "Google has spent over 1.1 billion dollars on self-driving tech." Available at https://spectrum.ieee.org/cars-that-think/transportation/self-driving/google-has-spent-over-11-billion-on-selfdriving-tech.

[2] D. Primack and K. Korosec, "Gm buying self-driving tech startup for more than 1 billion dollars." Available at https://fortune.com/2016/03/11/gm-buying-self-driving-tech-startup-for-more-than-1-billion/.

[3] "Ford invests in argo ai, a new artificial intelligence company, in drive for autonomous vehicle leadership." Available at https://media.ford.com/content/fordmedia/fna/us/en/news/2017/02/10/ford-invests-in-argo-ai-new-artificial-intelligence-company.html.

[4] W. H. Organization, "Global status report on road safety 2018: Summary," tech. rep., World Health Organization, 2018.

[5] "Automated vehicles for safety." Available at https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety.

[6] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[7] C. Urmson, J. Anhalt, D. Bagnell, C. Baker, R. Bittner, M. Clark, J. Dolan, D. Duggins, T. Galatali, C. Geyer, *et al.*, "Autonomous driving in urban environments: Boss and the urban challenge," *Journal of Field Robotics*, vol. 25, no. 8, pp. 425–466, 2008.

[8] A. Bacha, C. Bauman, R. Faruque, M. Fleming, C. Terwelp, C. Reinholtz, D. Hong, A. Wicks, T. Alberi, D. Anderson, *et al.*, "Odin: Team victortango's entry in the darpa urban challenge," *Journal of field Robotics*, vol. 25, no. 8, pp. 467–492, 2008.

[9] T. Gindele, D. Jagszent, B. Pitzer, and R. Dillmann, "Design of the planner of team anieway's autonomous vehicle used in the darpa urban challenge 2007," in *2008 IEEE Intelligent Vehicles Symposium*, pp. 1131–1136, IEEE, 2008.

[10] M. Buehler, K. Iagnemma, and S. Singh, *The DARPA urban challenge: autonomous vehicles in city traffic*, vol. 56. springer, 2009.

[11] F. W. Rauskolb, K. Berger, C. Lipski, M. Magnor, K. Cornelsen, J. Effertz, T. Form, F. Graefe, S. Ohl, W. Schumacher, *et al.*, "Caroline: An autonomously driving vehicle for urban environments," *Journal of Field Robotics*, vol. 25, no. 9, pp. 674–724, 2008.

[12] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[13] S. Brechtel, T. Gindele, and R. Dillmann, "Probabilistic mdp-behavior planning for cars," in *2011 14th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, pp. 1537–1542, IEEE, 2011.

[14] S. Ulbrich and M. Maurer, "Probabilistic online pomdp decision making for lane changes in fully automated driving," in *16th International IEEE Conference on Intelligent Transportation Systems (ITSC 2013)*, pp. 2063–2067, IEEE, 2013.

[15] J. Wei, J. M. Dolan, J. M. Snider, and B. Litkouhi, "A point-based mdp for robust single-lane autonomous driving behavior under uncertainties," in *2011 IEEE International Conference on Robotics and Automation*, pp. 2586–2592, IEEE, 2011.

[16] T. Bandyopadhyay, K. S. Won, E. Frazzoli, D. Hsu, W. S. Lee, and D. Rus, "Intention-aware motion planning," in *Algorithmic foundations of robotics X*, pp. 475–491, Springer, 2013.

[17] H. Kita, "A merging–giveway interaction model of cars in a merging section: a game theoretic analysis," *Transportation Research Part A: Policy and Practice*, vol. 33, no. 3-4, pp. 305–312, 1999.

[18] H. X. Liu, W. Xin, Z. Adam, and J. Ban, "A game theoretical approach for modelling merging and yielding behaviour at freeway on-ramp sections," *Transportation and traffic theory*, vol. 3, pp. 197–211, 2007.

[19] A. Talebpour, H. S. Mahmassani, and S. H. Hamdar, "Modeling lane-changing behavior in a connected environment: A game theory approach," *Transportation Research Part C: Emerging Technologies*, vol. 59, pp. 216–232, 2015.

[20] J. C. Harsanyi, "Games with incomplete information played by "bayesian" players, i–iii part i. the basic model," *Management science*, vol. 14, no. 3, pp. 159–182, 1967.

[21] P. Bajari, H. Hong, and S. P. Ryan, "Identification and estimation of a discrete game of complete information," *Econometrica*, vol. 78, no. 5, pp. 1529–1568, 2010.

[22] H. Yu, H. E. Tseng, and R. Langari, "A human-like game theory-based controller for automatic lane changing," *Transportation Research Part C: Emerging Technologies*, vol. 88, pp. 140–158, 2018.

[23] M. Wang, S. P. Hoogendoorn, W. Daamen, B. van Arem, and R. Happee, "Game theoretic approach for predictive lane-changing and car-following control," *Transportation Research Part C: Emerging Technologies*, vol. 58, pp. 73–92, 2015.

[24] D. Katzourakis, J. C. de Winter, S. de Groot, and R. Happee, "Driving simulator parameterization using double-lane change steering metrics as recorded on five modern cars," *Simulation Modelling Practice and Theory*, vol. 26, pp. 96–112, 2012.

[25] J. H. Yoo and R. Langari, "A stackelberg game theoretic driver model for merging," in *Dynamic Systems and Control Conference*, vol. 56130, p. V002T30A003, American Society of Mechanical Engineers, 2013.

[26] N. Li, D. W. Oyler, M. Zhang, Y. Yildiz, I. Kolmanovsky, and A. R. Girard, "Game theoretic modeling of driver and vehicle interactions for verification and validation of autonomous vehicle control systems," *IEEE Transactions on control systems technology*, vol. 26, no. 5, pp. 1782–1797, 2017.

[27] F. Meng, J. Su, C. Liu, and W.-H. Chen, "Dynamic decision making in lane change: Game theory with receding horizon," in *2016 UKACC 11th International Conference on Control (CONTROL)*, pp. 1–6, IEEE, 2016.

[28] S. Coskun, Q. Zhang, and R. Langari, "Receding horizon markov game autonomous driving strategy," in *2019 American Control Conference (ACC)*, pp. 1367–1374, IEEE, 2019.

[29] D. Isele, R. Rahimi, A. Cosgun, K. Subramanian, and K. Fujimura, "Navigating occluded intersections with autonomous vehicles using deep reinforcement learning," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2034–2039, IEEE, 2018.

[30] P. Wang and C.-Y. Chan, "Formulation of deep reinforcement learning architecture toward autonomous driving for on-ramp merge," in *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 1–6, IEEE, 2017.

[31] P. Wang, C.-Y. Chan, and A. de La Fortelle, "A reinforcement learning based approach for automated lane change maneuvers," in *2018 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1379–1384, IEEE, 2018.

[32] H. An and J.-i. Jung, "Decision-making system for lane change using deep reinforcement learning in connected and automated driving," *Electronics*, vol. 8, no. 5, p. 543, 2019.

[33] X. Li, X. Xu, and L. Zuo, "Reinforcement learning based overtaking decision-making for highway autonomous driving," in *2015 Sixth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 336–342, IEEE, 2015.

[34] X. Li, X. Qiu, J. Wang, and Y. Shen, "A deep reinforcement learning based approach for autonomous overtaking," in *2020 IEEE International Conference on Communications Workshops (ICC Workshops)*, pp. 1–5, IEEE, 2020.

[35] A. E. Sallab, M. Abdou, E. Perot, and S. Yogamani, "End-to-end deep reinforcement learning for lane keeping assist," *arXiv preprint arXiv:1612.04340*, 2016.

[36] J. Zhang, H. Chen, S. Song, and F. Hu, "Reinforcement learning-based motion planning for automatic parking system," *IEEE Access*, vol. 8, pp. 154485–154501, 2020.

[37] P. Zhang, L. Xiong, Z. Yu, P. Fang, S. Yan, J. Yao, and Y. Zhou, "Reinforcement learning-based end-to-end parking for automatic parking system," *Sensors*, vol. 19, no. 18, p. 3996, 2019.

[38] R. Li, Y. Li, S. E. Li, E. Burdet, and B. Cheng, "Indirect shared control of highly automated vehicles for cooperative driving between driver and automation," *arXiv preprint arXiv:1704.00866*, 2017.

[39] C. Sentouh, S. Debernard, J.-C. Popieul, and F. Vanderhaegen, "Toward a shared lateral control between driver and steering assist controller," *IFAC Proceedings Volumes*, vol. 43, no. 13, pp. 404–409, 2010.

[40] A.-T. Nguyen, C. Sentouh, and J.-C. Popieul, "Driver-automation cooperative approach for shared steering control under multiple system constraints: Design and experiments," *IEEE Transactions on Industrial Electronics*, vol. 64, no. 5, pp. 3819–3830, 2016.

[41] B. Soualmi, C. Sentouh, J.-C. Popieul, and S. Debernard, "Automation-driver cooperative driving in presence of undetected obstacles," *Control engineering practice*, vol. 24, pp. 106–119, 2014.

[42] S. Inoue, T. Ozawa, H. Inoue, P. Raksincharoensak, and M. Nagai, "Cooperative lateral control between driver and adas by haptic shared control using steering torque assistance combined with direct yaw moment control," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 316–321, IEEE, 2016.

[43] C. Guo, C. Sentouh, J.-C. Popieul, and J.-B. Haué, "Mpc-based shared steering control for automated driving systems," in *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 129–134, IEEE, 2017.

[44] C. Guo, C. Sentouh, J.-C. Popieul, and J.-B. Haué, "Predictive shared steering control for driver override in automated driving: A simulator study," *Transportation research part F: traffic psychology and behaviour*, vol. 61, pp. 326–336, 2019.

[45] E. D. Dickmanns, *Dynamic vision for perception and control of motion*. Springer Science & Business Media, 2007.

[46] L. Saleh, P. Chevrel, F. Claveau, J.-F. Lafay, and F. Mars, "Shared steering control between a driver and an automation: Stability in the presence of driver behavior uncertainty," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 2, pp. 974–983, 2013.

[47] Z. Ercan, A. Carvalho, M. Gokasan, and F. Borrelli, "Modeling, identification, and predictive control of a driver steering assistance system," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 700–710, 2017.

[48] S. M. Erlien, S. Fujita, and J. C. Gerdes, "Shared steering control using safe envelopes for obstacle avoidance and vehicle stability," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, no. 2, pp. 441–451, 2015.

[49] K. Iwano, P. Raksincharoensak, and M. Nagai, "A study on shared control between the driver and an active steering control system in emergency obstacle avoidance situations," *IFAC Proceedings Volumes*, vol. 47, no. 3, pp. 6338–6343, 2014.

[50] R. Li, Y. Li, S. E. Li, E. Burdet, and B. Cheng, "Driver-automation indirect shared control of highly automated vehicles with intention-aware authority transition," in *2017 IEEE Intelligent Vehicles Symposium (IV)*, pp. 26–32, IEEE, 2017.

[51] R. Li, S. Li, H. Gao, K. Li, B. Cheng, and D. Li, "Effects of human adaptation and trust on shared control for driver-automation cooperative driving," tech. rep., SAE Technical Paper, 2017.

[52] J. Jiang and A. Astolfi, "Shared-control for a rear-wheel drive car: Dynamic environments and disturbance rejection," *IEEE Transactions on Human-Machine Systems*, vol. 47, no. 5, pp. 723–734, 2017.

[53] J. Jiang and A. Astolfi, "Shared-control for the kinematic model of a rear-wheel drive car," in *2015 American Control Conference (ACC)*, pp. 1155–1160, IEEE, 2015.

[54] J. Jiang and A. Astolfi, "Shared-control for the lateral motion of vehicles," in *2018 European Control Conference (ECC)*, pp. 225–230, IEEE, 2018.

[55] X. Na and D. J. Cole, "Linear quadratic game and non-cooperative predictive methods for potential application to modelling driver–afs interactive steering control," *Vehicle System Dynamics*, vol. 51, no. 2, pp. 165–198, 2013.

[56] X. Na and D. J. Cole, "Game-theoretic modeling of the steering interaction between a human driver and a vehicle collision avoidance controller," *IEEE Transactions on Human-Machine Systems*, vol. 45, no. 1, pp. 25–38, 2014.

[57] X. Ji, Y. Liu, X. Na, and Y. Liu, "Research on interactive steering control strategy between driver and afs in different game equilibrium strategies and information patterns," *Vehicle system dynamics*, vol. 56, no. 9, pp. 1344–1374, 2018.

[58] X. Ji, K. Yang, X. Na, C. Lv, and Y. Liu, "Shared steering torque control for lane change assistance: A stochastic game-theoretic approach," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3093–3105, 2018.

[59] A. Zhakatayev, B. Rakhim, O. Adiyatov, A. Baimyshev, and H. A. Varol, "Successive linearization based model predictive control of variable stiffness actuated robots," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1774–1779, IEEE, 2017.

[60] J. B. Rawlings and D. Q. Mayne, *Model predictive control: Theory and design*. Nob Hill Pub., 2009.

[61] A. Matsumoto, F. Szidarovszky, *et al.*, *Game theory and its applications*. Springer, 2016.

[62] L. Wang, *Model predictive control system design and implementation using MATLAB®*. Springer Science & Business Media, 2009.

[63] Y. Zhang, E. K. Antonsson, and K. Grote, "A new threat assessment measure for collision avoidance systems," in *2006 IEEE Intelligent Transportation Systems Conference*, pp. 968–975, IEEE, 2006.

[64] Y.-L. Chen, "An explicit and novel forward collision probability index," in *2015 IEEE 10th Conference on Industrial Electronics and Applications (ICIEA)*, pp. 1778–1782, IEEE, 2015.

[65] M. Ławryńczuk, "Computationally efficient model predictive control algorithms," *A Neural Network Approach, Studies in Systems, Decision and Control*, vol. 3, 2014.

[66] P. Tøndel and T. A. Johansen, "Complexity reduction in explicit linear model predictive control," *IFAC Proceedings Volumes*, vol. 35, no. 1, pp. 189–194, 2002.

[67] Y. Wang and S. Boyd, "Fast model predictive control using online optimization," *IEEE Transactions on control systems technology*, vol. 18, no. 2, pp. 267–278, 2009.

[68] D. of Transportation, "Next generation simulation." Available at `www.http://ngsim.fhwa.dot.gov`.

[69] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "Carla: An open urban driving simulator," in *Conference on robot learning*, pp. 1–16, PMLR, 2017.

[70] S. Shah, D. Dey, C. Lovett, and A. Kapoor, "Airsim: High-fidelity visual and physical simulation for autonomous vehicles," in *Field and service robotics*, pp. 621–635, Springer, 2018.

[71] E. Leurent, "An environment for autonomous driving decision-making." `https://github.com/eleurent/highway-env`, 2018.

[72] M. Treiber, A. Hennecke, and D. Helbing, "Congested traffic states in empirical observations and microscopic simulations," *Physical review E*, vol. 62, no. 2, p. 1805, 2000.

[73] M. Treiber and D. Helbing, "Mobil: General lane-changing model for car-following models," *Disponıvel Acesso Dezembro*, 2016.

[74] E. Balal, R. L. Cheu, and T. Sarkodie-Gyan, "A binary decision model for discretionary lane changing move based on fuzzy inference system," *Transportation Research Part C: Emerging Technologies*, vol. 67, pp. 47–61, 2016.

[75] H. Bi, T. Mao, Z. Wang, and Z. Deng, "A data-driven model for lane-changing in traffic simulation.," in *Symposium on Computer Animation*, pp. 149–158, 2016.

[76] J. Nie, J. Zhang, X. Wan, W. Ding, and B. Ran, "Modeling of decision-making behavior for discretionary lane-changing execution," in *2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC)*, pp. 707–712, IEEE, 2016.

[77] Y. Liu, X. Wang, L. Li, S. Cheng, and Z. Chen, "A novel lane change decision-making model of autonomous vehicle based on support vector machine," *IEEE Access*, vol. 7, pp. 26543–26550, 2019.

[78] Y. Zhang, Q. Lin, J. Wang, S. Verwer, and J. M. Dolan, "Lane-change intention estimation for car-following control in autonomous driving," *IEEE Transactions on Intelligent Vehicles*, vol. 3, no. 3, pp. 276–286, 2018.

[79] J. Schmidhuber and S. Hochreiter, "Long short-term memory," *Neural Comput*, vol. 9, no. 8, pp. 1735–1780, 1997.

[80] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," *arXiv preprint arXiv:1406.1078*, 2014.

[81] A. Alalshekmubarak and L. S. Smith, "A novel approach combining recurrent neural network and support vector machines for time series classification," in *2013 9th International Conference on Innovations in Information Technology (IIT)*, pp. 42–47, IEEE, 2013.

[82] A. F. M. Agarap, "A neural network architecture combining gated recurrent unit (gru) and support vector machine (svm) for intrusion detection in network traffic data," in *Proceedings of the 2018 10th International Conference on Machine Learning and Computing*, pp. 26–30, 2018.

[83] B. S. G. de Almeida and V. C. Leite, "Particle swarm optimization: A powerful technique for solving engineering problems," *Swarm Intelligence-Recent Advances, New Perspectives and Applications*, 2019.

[84] Z. A. B. S. . S. I. Klancar, G., *Wheeled mobile robotics: from fundamentals towards autonomous systems*. Butterworth-Heinemann, 2017.

[85] T. Gu, J. Snider, J. M. Dolan, and J.-w. Lee, "Focused trajectory planning for autonomous on-road driving," in *2013 IEEE Intelligent Vehicles Symposium (IV)*, pp. 547–552, IEEE, 2013.

[86] Y. Cong, O. Sawodny, H. Chen, J. Zimmermann, and A. Lutz, "Motion planning for an autonomous vehicle driving on motorways by using flatness properties," in *2010 IEEE international conference on control applications*, pp. 908–913, IEEE, 2010.

[87] M. Elbanhawi, M. Simic, and R. Jazar, "Improved manoeuvring of autonomous passenger vehicles: simulations and field results," *Journal of Vibration and Control*, vol. 23, no. 12, pp. 1954–1983, 2017.

[88] D. Madås, M. Nosratinia, M. Keshavarz, P. Sundström, R. Philippsen, A. Eidehall, and K.-M. Dahlén, "On path planning methods for automotive collision avoidance," in *2013 IEEE intelligent vehicles symposium (IV)*, pp. 931–937, IEEE, 2013.

[89] R. Kala and K. Warwick, "Motion planning of autonomous vehicles in a non-autonomous vehicle environment without speed lanes," *Engineering Applications of Artificial Intelligence*, vol. 26, no. 5-6, pp. 1588–1601, 2013.

[90] M. B. Saltık, L. Özkan, J. H. Ludlage, S. Weiland, and P. M. Van den Hof, "An outlook on robust model predictive control algorithms: Reflections on performance and computational aspects," *Journal of Process Control*, vol. 61, pp. 77–102, 2018.

[91] A. A. Jalali and V. Nadimi, "A survey on robust model predictive control from 1999-2006," in *2006 International Conference on Computational Inteligence for Modelling Control and Automation and International Conference on Intelligent Agents Web Technologies and International Commerce (CIMCA'06)*, pp. 207–207, IEEE, 2006.

[92] Y. J. Wang and J. B. Rawlings, "A new robust model predictive control method i: theory and computation," *Journal of Process Control*, vol. 14, no. 3, pp. 231–247, 2004.

APPENDIX A

NONLINEAR VEHICLE DYNAMICS

A.1 Nonlinear vehicle dyamics

$$\begin{aligned}\dot{\mathbf{x}} &= \mathbf{f}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) \\ \mathbf{y_m} &= \mathbf{g_m}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) \\ \mathbf{y_d} &= \mathbf{g_d}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d})\end{aligned} \tag{A.1}$$

$$\mathbf{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \end{bmatrix} = \begin{bmatrix} X \\ Y \\ \psi \\ v_x \\ v_y \\ \omega \\ \theta_s \\ \dot{\theta}_s \end{bmatrix}, \quad \mathbf{u_m} = \begin{bmatrix} u_{m,1} \\ u_{m,2} \end{bmatrix} = \begin{bmatrix} a_{x,m} \\ \tau_m \end{bmatrix}, \quad \mathbf{u_d} = \begin{bmatrix} u_{d,1} \\ u_{d,2} \end{bmatrix} = \begin{bmatrix} a_{x,d} \\ \tau_d \end{bmatrix} \tag{A.2}$$

where, $f_1(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_4 \cos x_3 - x_5 \sin x_3$

$$f_2(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_4 \sin x_3 + x_5 \cos x_3$$

$$f_3(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_6$$

$$f_4(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_6 x_5 + u_{m,1} + u_{d,1}$$

$$f_5(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = -x_6 x_5 + \frac{1}{m}\left(F_{y,f} \cos \frac{x_7}{G} + F_{y,r}\right)$$

$$f_6(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = \frac{1}{I_z}\left(L_f F_{y,f} \cos \frac{x_7}{G} - L_r F_{y,r}\right)$$

$$f_7(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_8$$

$$f_8(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = \frac{1}{J_s + G_p^2 M_r}\left(u_{d,2} + G_g u_{m,2} - K_r G_p^2 x_7 - (B_c + B_r G_p^2) x_8 - \frac{\eta_t F_{y,f}}{G}\right)$$

$$g_{m,1}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_1, \qquad g_{m,1}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_1$$

$$g_{m,2}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_2, \qquad g_{m,2}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_2$$

$$g_{m,3}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_3, \qquad g_{m,3}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_3$$

$$g_{m,4}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_4, \qquad g_{m,4}(\mathbf{x}, \mathbf{u_m}, \mathbf{u_d}) = x_4$$

$$F_{y,f} = D_f \sin(C_f \arctan B_f \alpha_f)$$

$$F_{y,r} = D_r \sin(C_r \arctan B_r \alpha_r)$$

$$\alpha_f = \frac{x_7}{G} - \arctan\left(\frac{L_f x_6 + x_5}{x_4}\right)$$

$$\alpha_r = -\arctan\left(\frac{L_r x_6 - x_5}{x_4}\right)$$