COLLISION AVOIDANCE FOR AUTONOMOUS CARS BASED ON HUMAN INTENTION

By

DENIS OSIPYCHEV

Bachelor of Science in Electrical Engineering

Moscow Power Engineering Institute

Moscow, Russia

2004

Submitted to the Faculty of the
Graduate College of
Oklahoma State University
in partial fulfillment of
the requirements for
the Degree of
MASTER OF SCIENCE
July, 2015

COLLISION AVOIDANCE FOR AUTONOMOUS CARS

BASED ON HUMAN INTENTION

Thesis Approved:

Dr. Weihua Sheng

Committee Chair and Thesis Advisor

Dr. Carl Latino

Committee member

Dr. Girish Chowdhary

Committee member

Acknowledgments

First and foremost, I would like to thank my family. My wife Natalya Osipycheva helped me all along the way to this accomplishment. Without her support, I would never make a decision to go back to school. My son Lucas gave me a new look at the world and ability to learn and see things from a different angle. I am also grateful to my parents for their strengths to let me stay on another side of the Earth and take the people so loved by them with me.

I want to thank my adviser, Dr. Weihua Sheng and my committee members, Dr. Girish Chowdhary, Dr. Carl Latino, Dr. Martin Hagan. You did not only share your experience, but let me found a new passion in robotics and control algorithms. Without your guidance and help I would never reach these results.

I would also like to acknowledge all of the members of the Advanced Sensing, Computation and Control Laboratory (ASCC Lab) and the Distributed Autonomous Systems Laboratory (DASLab). Your contribution made this research possible. I want to thank Allan Axelrod who has provided valuable review to my research and was always happy to share his great ideas.

Name: DENIS OSIPYCHEV

Date of Degree: JULY, 2015

Title of Study: COLLISION AVOIDANCE FOR AUTONOMOUS CARS BASED ON HUMAN INTENTION

Major Field: ELECTRICAL ENGINEERING

Abstract: This thesis considers a problem of controlling an autonomous car cooperating with human-driven cars. Proposed proactive collision avoidance system incorporates human-driver's intentions transfered via Vehicle-to-Vehicle (V2V) communication. This system utilizes multi-step Vector Gaussian Processes (VGP) and Stochastic Transition models in order to learn the resulting transitions for each given intention and update them on-line from the real driving scenarios to provide an adaptive intention-based trajectory prediction for any kind of driving manner and road/weather condition. Such an approach allows us to use stochastic behavioral model for a numerical evaluation of the risk of collision and pose the question of control of the vehicle as an optimization problem. This formulation makes it possible to utilize various existing optimization techniques what is proven by the use of both a single-step cost function minimization and sequential decision making algorithm based on Markov Decision Process (MDP). The effectiveness of this concept is supported by a variety of simulations utilizing the real human driving in various scenarios considering both an intersection and highway scenarios in specially developed Matlab driving simulation and highly realistic third-party developed car simulator Carnetsoft.

184 words

Table of Contents

List of Figures

CHAPTER 1

Introduction

1.1 Motivation

A century after the invention of automobiles, the land transportation remains to be the most dangerous way to travel and move goods. According to the Bureau of Transportation Statistics, the total number of vehicles in the US raised from 74 million in 1960 to 254 million in 2012 [2]. Because of the constantly improving safety, the rate of deaths caused by car accidents declined in the same period of time. Since the first vehicle was built, many improvements have been made in the cars. They got powerful and efficient engines, better dynamics, controllability and stability on the road. However, the total number of fatalities remains to be the same high: over 32 thousand people die on the road every year and the US economy lose 277 billion dollars [3]. Despite the fact that the cars get safer and easier to control, they require more attention from the driver. Heavy traffic, high speed and fast maneuvers need the full concentration and reaction from the people controlling the vehicles. Natural human abilities become the major limit which does not allow to act fast and take urgent actions when it is needed. Recent improvements showed the success of assistance to the driver which extends the reaction and driving abilities of the human [4]. Advanced sensors allowing to see in dark, detectors, fast control logic already found its place on the market.

The next level improvement is an driving advising system watching the driver and environment, and taking control of the car when it is needed or even doing a fully autonomous driving [5, 6]. According to some optimistic prediction that the use of fully autonomous cars may reduce the total number of accidents by 90% and lead to enormous economic impact to the car industry [7, 8], which pushes the leaders of the industry to develop and introduce an autonomous cars. However, developing such a vehicle is a complex task that requires to solve many challenges. The first difficulty is localization and mapping: it is hard to determine the exact location of the vehicle on the road. Road works change the map and make it irrelevant. The second problem is sensing. This is a very large topic related to a computer vision and object recognition from the

sensor's data. The third challenge is related to control of the autonomous vehicle and especially cooperation with human drivers. The transition from now to the days when all vehicles on the road will be autonomous may take decades. Until then, both human drivers and autonomous cars will be present on the road and have to be able to cooperate with each other.

In this work, we develop a control algorithm for autonomous cars allowing to perform safe autonomous driving, cooperate with human drivers and take actions to avoid collision with them.

1.2 Problem Context

People drive as an every day task, carrying them from home to work and back. Most of the time they do it without any trouble. The rate of the collisions is high due to the high number of vehicles, particularly during rush hours. Unfortunately, human-driven cars have been and will still be the majority of vehicles on the road, so we have to consider human driving cars when we develop a safe autonomous vehicle-driving algorithm. Even though we assume to use robotic cars on public roads in the nearest future, human drivers would remain to be the main problem which has to be taken into account [9]. In this section, the advantage and disadvantage of human driving will be discussed, which will be considered in the design and implementation of intelligent, self-driving cars or driving assistance systems. In order to solve the task of safe driving, it is necessary to assess the risk of the traffic situation around and make a decision based on this assessment. The ability of the human to estimate risk relies on one's ability to predict changes in the situation on the road, to avoid not only accidents, but also the possibility of it. Lets look at this problem closer.

The statistics of accidents might give us a clue what the human drivers are doing wrong and what should we focus on developing an intelligent vehicle. According to the recent report of US Census Bureau, accident rate for licensed drivers under 19 years old is 20% while the highest result 23% is for 16 years old, while the accident rate for people 20-24 years old is about 14%, older than 25 is 4-9%. Furthermore, 38% of all fatalities on the road were caused by the age group under 19 years old [10]. Another highly important factor of the accidents rate is a driver's distraction, 16% of all accidents were caused by driver distraction. Nearly 448000 people were injured in crashes resulted from driver's distraction, which is 20% of all injured. The third important factor is alcohol; 22% of all accidents happened with the drivers whose blood alcohol content (BAC) was higher than 0.08% which were considered as drunk driver in the US. Understanding of the factors causing the collisions will help us to take them into account when autonomous cars cooperate with a human-driver. It also lets us focus on the most problematic areas of the human role model which we utilizes for our system.

Statistics show that the rate of accidents correlates with the age of the driver. Young drivers have been involved in to collision up to 7 times more often than the experienced drivers older than 25 years old. Experience plays the principal role in the safety. Meanwhile, up to 16% of all accidents were caused by driver distraction. Drinking and the use of cell-phones while driving cause the loss of information and a significant delay in reaction and potentially the instant change in the road situation. Even small distraction can increase the reaction time up to 2 times [11]. Concentration on the road activity is another highly important factor affecting the safety. The intelligent vehicle is intended to mitigate these human weaknesses. The goal of this work is to develop an intelligent system which can performs an autonomous driving without the human driver. This system should have an ability to be trained, to update its experience from observations, to solve problems based on this experience, to proactively avoid the collision if necessary. In this context, these systems can secure ground transportation by operating independently and supporting drivers who are subject to distractions.

1.3 Literature Review

There are many different approaches performing a driving assistance and reactive safety is one of them. It warns the driver about difficulties on a road or even executes the urgent actions to avoid an accident [12, 13]. These improvements were developed to surpass the human in time of reaction or excellence of sensors. Because of the use of modern detectors and fast computer logic, such systems had many successful implementations and prevented up to 80% of simulated collisions [14, 15]. An advanced example of a completely reactive robotic system is a vehicle called ALVINN which utilizes the images from the cameras and Neural Networks for reactive control [16]. Another great example is a collision avoidance system using optical pattern growth rate which applies brake when the relative size of the object in front is growing on the image from the front view camera [17]. Further safety improvements require increasing the sensitivity of the reactive systems, but that leads to an increase in the number of false alarms. Also, most of those systems were non-optimal and annoying to the passengers and obviously they cannot control the car all alone. However, these methods could find its place as an emergency low-level control.

Besides the reactive control, an autonomous vehicle should have an ability to plan its action ahead with respect to the intention of others. This makes such an algorithm proactive and allows to achieve a higher sensitivity to a potentially danger situation while taking softer actions. Despite the use of both proactive and reactive methods in mobile robotics research, it is still a challenge for its adoption in transportation vehicles due to the difficulties described below.

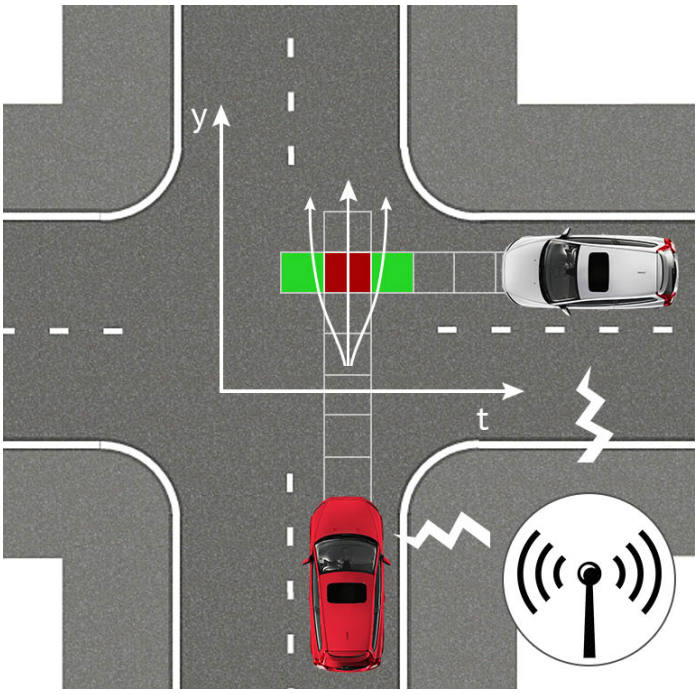

Figure 1.1: An example of a possible collision avoidance scenario involving the use of an intersection's infrastructure. An anticipating change in the velocity is proposed to be enough to avoid collision at intersection.

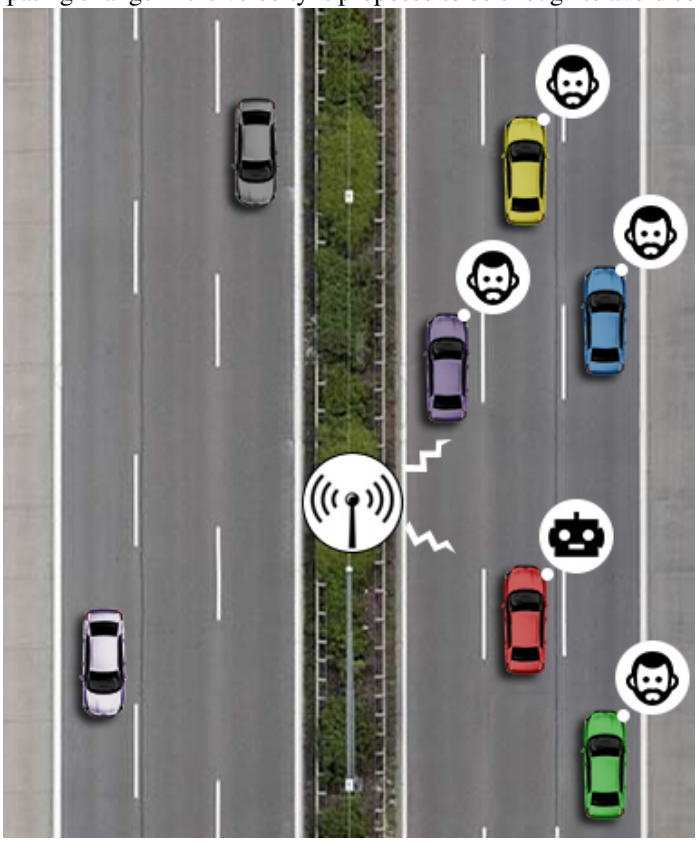

Figure 1.2: An example of the highway scenario: an autonomous vehicle surrounded by human-driven cars

First, the ability of a system to plan the actions in advance requires to know the intention of human drivers. This task has been previously solved using classification algorithms analyzing human activity and giving the intention or warning about possible unintended actions resulted by drowsiness or distraction [18, 19]. The second challenge in developing a proactive algorithm is modeling of the human behavior which is required for prediction of the trajectory of the vehicle. This task may be solved by learning-based behavior models, for example, utilizing the Gaussian model [20, 21]. The third problem is the evaluation of the corrective action based on the possible outcomes which it can result. This task could be solved as an optimization task using various single-step and sequential decision-making techniques such as tree search [22]. Using a tree search might be inconsistent and requires many function evaluations which can slow down the decision making. Another common way of solving this problem is using the potential fields method [23]. This method is representing the dynamic obstacles as objects with potential fields forcing the autonomous vehicle to move from them, however this method assumes the use of the continuous model of the world and the knowledge of the derivative of this fields. More natural solution to collision avoidance task is using a sequential Markov Decision Process (MDP) and a Partially Observable MDP (POMDP) where hidden intentions affect on obstacle behavior and transitions. Most of these works consider the world as partially observed or completely hidden where the motivation and dynamics of the processes are not available while only the effects of certain actions can be observed [24, 25]. In theory, these methods would give an elegant solution, but they are very hard to solve, require many samples to establish the hidden links and are hard to check the correctness of the solution, which is highly important in the road safety aspect. This paper evaluates the use of both single-step cost function optimization and classic Markov Decision Process (MDP) to solve the problem. In this way, it allows us to find the best actions given full knowledge of the parameters of speed, direction and position for all involved vehicles. This condition can be satisfied by establishing radio-frequency (RF) connections between all cars and transferring the data to each other using vehicle-to-vehicle (V2V) or vehicle-to-infrastructure (V2I) communication as has been explained in the work [26].

1.4 Solution Overview and Outline

This work focuses on the task of controlling an intelligent self-driven car surrounded by human-driven cars. Here, we consider two general cases: a highway scenario where all cars are driving the same direction on a congested highway, as shown in Fig. 1.2 and an intersection scenario where human-driven cars are moving in transverse direction to the autonomous vehicle as shown in Fig. 1.1. We proposed, developed and tested the Collision Avoidance System (CAS) with respect to those two general tasks. To achieve this goal, the work

has been divided into two general subtasks as shown in Fig. 1.3: prediction and decision making. In the first task, we need to track the behavior of the other drivers and make a prediction of their trajectories in the near future. For that reason, we introduced behavior models which store both human-driven car and autonomous car activity. In the second task, the proposed CAS system has to choose an action in order to continue the driving and be safe in the desired scenario. This job may be done by utilizing an optimization algorithm which defines the best action to take and shown by a separate block in Fig. 1.3. That desired action will be applied to the car dynamics using the vehicle's actuators such as steer, throttle and brake.

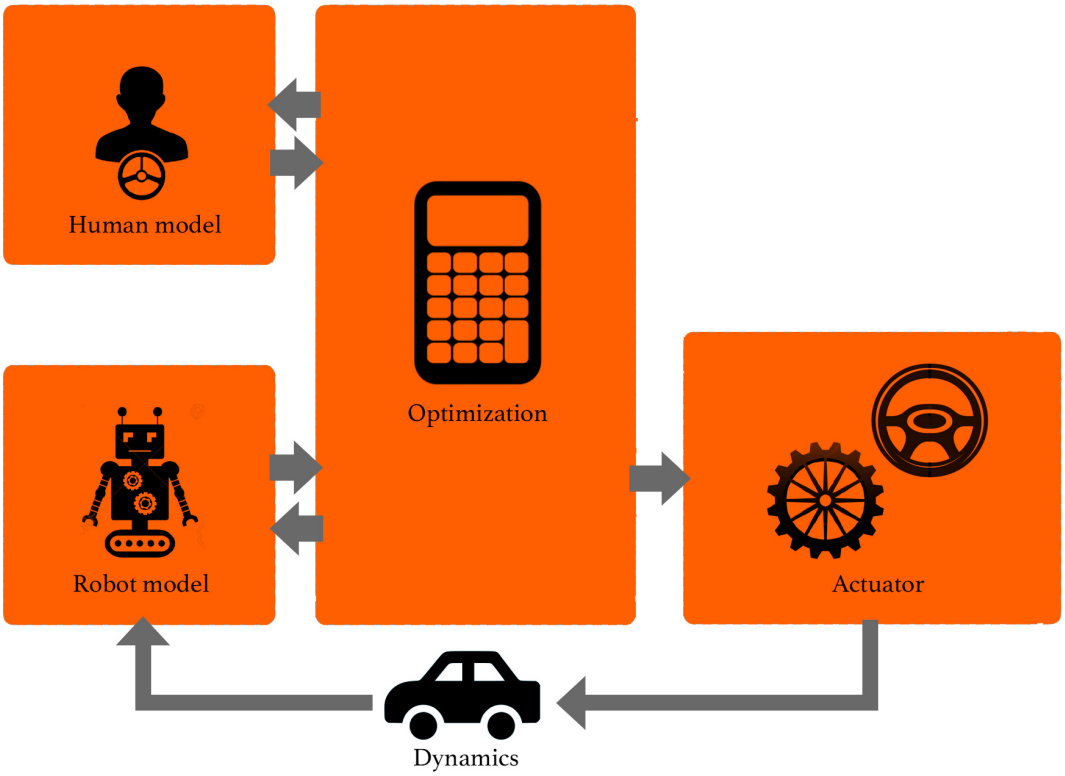

Figure 1.3: General overview of the proposed system. Prediction phase represented by Human and Robot models are developed to estimate trajectories of human-driven car and autonomous vehicle. Optimization phase evaluates the outcome of all possible actions.

This thesis provides and describes the complete solution for an autonomous driving task. In Chapter 2, we discuss the proposed behavior models, describe their structure, training and prediction processes. Chapter 3 of this thesis explains the optimization techniques what may be used for car collision avoidance. Chapter 4 provides the details of the simulations which we use to prove the work of the proposed system and shows the results of these simulations. In Chapter 5 we will conclude the work and indicate the topics for future research.

CHAPTER 2

Vehicle Behavior Modeling

2.1 Introduction

How do we expect to reproduce the experience of a driver? This chapter discusses the components which the driving experience consists of. The first is so called vehicle behavior model - knowledge of the behavior of the controlled car and understanding how it responds to control signals. The second is contextual knowledge of the effect of different environment conditions such as driving at night, in bad weather conditions, driving on a slippery road, etc., which affects on the braking distance and trajectory of the car. The third is the experience of driving in traffic where one must take into account the possibility of other human drivers changing lane without using a turning signal,possible changes in the environment or in intent of another driver. The last one may even lead to the generation of otherwise unexpected trajectories such as rerouting due to the missed turn or uneven driving while talking on the phone. The last type of experience, but not the least is the awareness of potentially dangerous zones on the road, such as intersections without a traffic light, lack of visibility from the perspective of our driver, and the blind spots of other car drivers.

How could we represent all this experience so the control algorithm can use? This work proposes to continuously learn the vehicle behavior model from observations. The key idea is to represent the knowledge of experienced drivers as Behavior Models which explicitly describe the action-consequences relation. As we can see later, it is true even for the cases when the intent action is unknown. All the experience of the control algorithm will be stored in two separate behavior models: human-driver behavior model (HBM) and autonomous vehicle behavior model (ABM). Then, HBM represents the knowledge of all possible resulting states when the intention of a human is given, while the ABM stores the future states of all the actions available to the autonomous vehicle. The following sections will detail how behavior isrepresented.

2.2 Terms Definitions and Assumptions

Before we start modeling the behavior, we have to define the terms that we are going to use. The behavior models are going to store the information about transition of the vehicle from one location to another in time. To explicitly describe this transition we have to know the physical state of the vehicle. However, it is impossible to keep track of many parameters of the car at the same time and learn their effect on the transition itself. To consider only important parameters, the Markov assumption seems to be a reasonable assumption to take. Hence, it is assumed that at any certain time all information about the physical state of the car is complete and enough to predict the next state. To reproduce the motion of the car, we need to know its previous location on the road in X, Y coordinates, angle of steering wheel and vehicle orientation, velocity and acceleration of the vehicle. By knowing all of these readings we can carefully predict the new state of the car after some finite time, assuming that the acceleration and steering wheel have not been changed in this time.

However, the use of all these readings would make the model very complex. So in favor of a simple proof-of-concept approximation, this work takes into consideration only location X, Y and total velocity V as a state of the car, assuming that all other readings are approximated to relatively small values. For clarity, we note that our assumptions treat the dynamics and control of the vehicle as allowing an immediate change of steering angle and acceleration. For example, assume that the vehicle is located in coordinates (x,y) and has velocity v in time t, then its state defined as $s(x,y,v,t)$. The behavior model carries the transition to the next state $s'(x',y',v',t+1)$.

The probabilistic transition from one state to another stores the probability distribution of the next location of the vehicle over the whole state space if the initial state is given. The transition probability may be unique for each input vector, which is given by the tuple (s,a). Action a might be given by any intentional and unintentional policy-action from the action-set A. For example, some driver had an unexpected trajectory shown in Fig. 2.1 and his intention has been recognized as an erratic driving. The future trajectory would be predicted based on the probabilistic transition of that driver where all these states would have some probability of being visited; so in the aforementioned scenario, a probabilistic approach is used to avoid a collision with the erratic or crazy driver.

Obviously the prediction system requires defining the position on the road carefully; this might be done by using the Computer Vision, cameras, LIDAR sensors or RF tracking of the car using infrastructure. However, the localization problem is beyond the scope of this work so we assume full knowledge of the location of the car with respect to the road. Note that longitudinal Y and lateral positions X of the vehicle were chosen

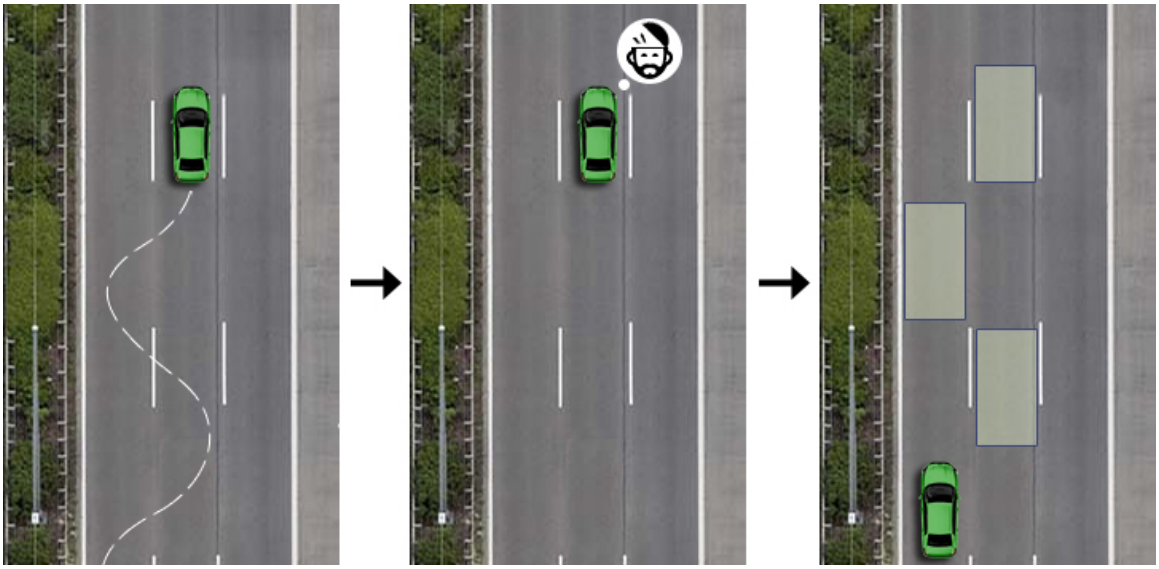

Figure 2.1: The Markov assumption allows to store all previous experience in the behavior model. An unexpected trajectory observed from the real driving is adopted to a Markov transition probability in order to predict the future possible trajectory.

as the location parameters.

2.3 Human Behavior Model

Let us define what a human behavior is first. In his paper, Ortiz defined a driver's behavior as saying that the set of actions caused by the aim of the person [21]. This can be interpreted as the behavior of the human can be divided which effectively separates the intentions of a driver and the driver's resultant actions. Let us consider the Human Behavior Model (HBM) first. As noted in the introduction, the main purpose of this model is to reproduce the transitions of the car from the current state to the next one. The actual transition observed from real-time driving behavior is used to train the model to make it be able to reproduce this transition and predict it in future.

For many years, human drivers use visual signals to notify others about the maneuver which they are going to perform. Such knowledge is important to plan a trajectory with respect to the future changes in the environment and increases the safety as well as making the driving comfort. The use of only visible communication signals makes it hard to share various intentions between drivers. Nowadays, more and more transportation systems are introducing the communication using an RF channel between vehicles and/or the road infrastructure. This communication is often called V2V for direct communication between vehicles or V2I for the use of the road infrastructure as the hub concentrating all data [27]. This technology allows vehicles equipped with radio transmitters to share as many intentions as we want using RF channel. However,

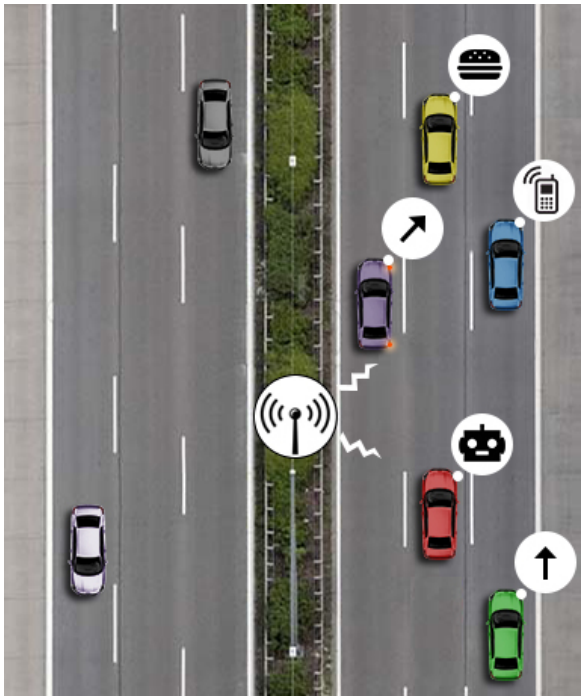

Figure 2.2: Vehicles shares their data between road users via V2I communication. This data includes the driver's intention expressing their will to change the trajectory shortly.

having so much data may cause confusion for other drivers. The data will need to be sorted and filtered for things like the level of hazard or group of recipients in a timely or real-time manner. While it is hard for human to handle all of these problems, they could be solved by an intelligent car which will use this information. In the example shown in Fig. 2.2, all cars may share their data between each other using wireless V2I communication. This data can include the conventional intentions of immediate driving actions such as changing lane, adjusting speed, performing a full stop. It could also include additional data showing how attentive the drivers are, if they are drowsy or distracted, and the global intention of drivers such as the current route based on the data from the in-car satellite navigation device of each vehicle. All of this information can give other participants a clue of next maneuver the driver wants to make, and possible next locations as a result. The known intentional driving actions of others can be transformed to the possible future locations of all cars around as in the example shown in Fig. 2.3. Meanwhile, the vehicle with the burger shown on Fig. 2.3 may be heading to the restaurant by following the route created by the in-car GPS device. This route will increase the probability of changing lane and taking the exit that is dictated by the navigation device.

To predict the next maneuver of the driver, we can utilize the turning signals which driver should use to express his intention. However, many drivers ignore the use of just these two existing signals. The better results can be reached by using modern classification algorithms such as Neural Networks (NNs), Hidden

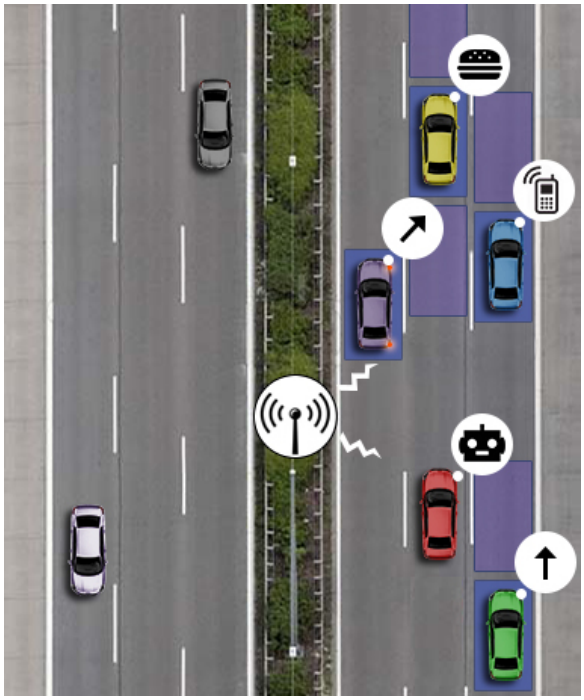

Figure 2.3: The prediction of future occupied locations is made based on the shared intention

Markov Models (HMMs), Support Vector Machines (SVMs) - all are well known for human activity recognition tasks [18]. This proposed behavior model works in cooperation with a human intention recognition system based on HMM classification algorithm developed by Duy Tran for ASCC laboratory [19]. This system chooses actions from the list of actions by observing the human driver motions and the state of the vehicle.

2.4 Autonomous Vehicle Behavior Model

This section covers the Autonomous Vehicle Behavior Model (ABM) - the model of behavior of the agent itself. This model, similar to the HBM, stores the transitions of the autonomous car with respect to the action selected. As it was discussed in Section 1.2, driving skills are the part of a driver's experience. This experience internalizes a relation between the actuating signal coming from the driver and the reaction of the car to that signal. It manifests in things as simple as a driver knowing that the car reduces speed if the driver pushes the brakes. The intelligent vehicle utilizes low level controlling signals to control the car such as steering, gas and brake, but this is not sufficiently effective for high-level decision making. For this reason, the driving task has been decomposed into two levels. At the low level, a control algorithm allows the car to follow the lane and keep the chosen speed. This task can be easily done by using proportional-derivative PD controller minimizing the difference between target values and actual lateral location and velocity of the

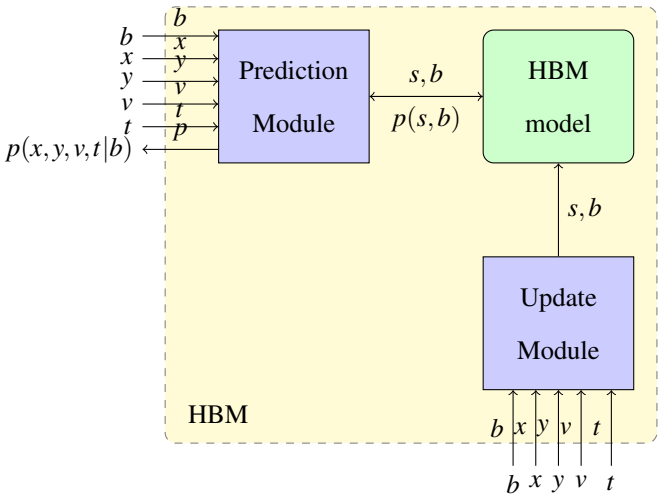

Figure 2.4: Proposed HBM system utilizes readings (x, y, v) of the vehicle to build a relation with an intention and time tuple (b, t). This tuple may be used to predict future (x, y, v) readings

.

vehicle.

At a higher level, a decision-making algorithm chooses an action from the set of actions available to the vehicle such as changing lanes and speed. This method is similar to rules of chess, where any action from the list of all possible actions could be selected to move you over the board. In the CAS system, the algorithm may know that the car should change lanes, but it may not know the way that this happens, how long this takes, or the interim states of the car between actual state and the target state. To learn these "rules," the decision-making algorithm requires a probabilistic transition model connecting the initial state with the desired state via all interim states.

We now consider a robotic car in state $s(x, y, v, t)$. Even if we know the target location given by $s'(x', y', v', t+1)$, the interim state between these two states are unknown because the time resolution doesn't allow us to track the car's motion. The coarse time resolution causes an uncertainty in the location of the autonomous car, even if the action and actual state are known for sure.

We find that the ABM model may be learned as effective as the HBM model. In a simplified case, there are several ways to store the model of the behavior. One way is to merge all possible transitions to the average and store its value and the variance value as being representative of the transition probability of every state in the state space. Another method is to store every transition probability in a table. Since the action spaces are different, it should be trained separately from the HBM, but the internal structure of both models are absolutely the same and would be described in next sections.

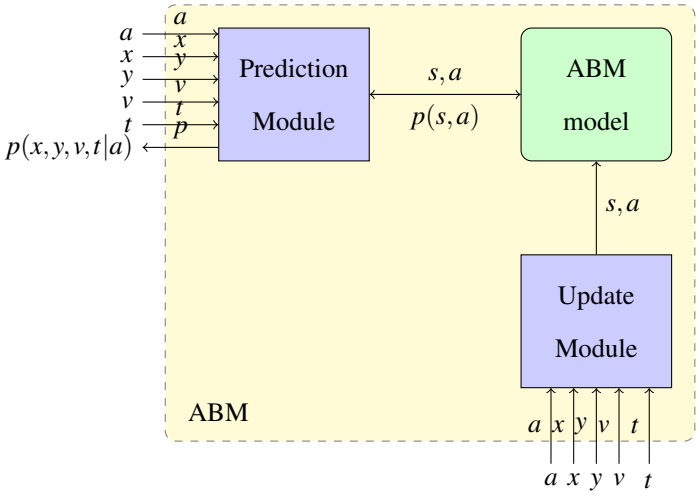

Figure 2.5: Proposed ABM system utilizes readings (x,y,v) of the autonomous vehicle to build a relation with an action and time tuple (a,t). This tuple may be used to predict future (x,y,v) readings.

2.5 Gaussian Model

Consider the task of moving from the point with coordinates (x,y) to (x',y'). There are an infinite number of routes that can be built from these two points. However, we can make an assumption that even if we have some number of drivers, their routes from point (x,y) to (x',y') will be the Gaussian distribution against the single most probable route between these two points. This assumption is based on the assumption that the road path is free between two points and there is no impossible locations on the road for any limited time.

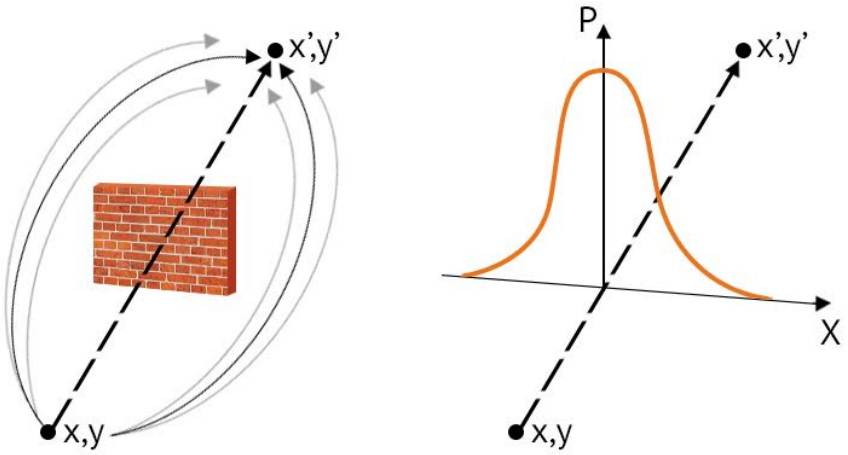

Figure 2.6: Problem with Gaussian distribution of the path causes the average trajectory comes through the obstacle if the time step is too large.

In the example shown in Fig. 2.6, we have an obstacle between points A and B. Half of the drivers may

prefer to drive around by one side, while the others may to drive around another side. Then the Gaussian distribution will give us the most probably route coming through the obstacle, which is wrong. However, if we divide the route between A and B by very small pieces smaller than the size of obstacle, we can assume that every piece of this route has a Gaussian distribution over the maximum likelihood.

To store the transitions, it might be possible to find separate Gaussian distributions over X and Y separately. However, this probability distribution will be symmetric against the axes while the actual trajectory of changing lane is a diagonal. For this reason, we use the Vectored Gaussian Processes (VGP) which stores the covariance between X and Y states which may make the trajectory diagonal.

2.6 Gaussian Processes

A Gaussian Process (GP) is a supervised learning method widely used for mapping an input to a corresponding output. The general idea of any supervised learning algorithm is to learn a relation between input/output pairs from a training dataset in such a way to be able to predict the output when given system inputs. There are two general subclasses of this machine learning method different by use of a parametric or nonparametric models. The parametric model assumes that known the nature of the relation which might be given by a function of a certain prespecified complexity and the task is to define parameters of such a function in order to fit the dataset. Nonparametric model is used when the nature of the function is unknown. In the CAS, the dataset is represented by the transition from one point to another, mainly due to the intent of the human performing this transition, so the nonparametric model is a general approach to model this relatively unknown and potentially very complex decision process. The GP is a Bayesian nonparametric method operating in Reproducing Kernel Hilbert Space widely used for signal estimation in control systems [28] and can be written as:

$$f(x) \sim GP(m(x), k(x,x')) \tag{2.1}$$

where $m(x)$ is the mean of dataset given by $D(x,y)$, and $k(x,x')$ is a covariance kernel used for approximating the covariance of the dataset $x \in X$ and other values x'. This work utilizes the Radial Basis Function kernel for $k(x,x')$ as a commonly used kernel function shown in Eq. 2.2. The prior of the GP is assumed to be zero, while the posterior distribution updates using Bayes law, where posterior distribution has a mean shown in Eq. 2.3 and a covariance in Eq. 2.4.

$$K(x,x') = \exp\left(-\frac{||x-x'||^2}{2\sigma^2}\right) \tag{2.2}$$

$$m' = ((K(X,X)+\omega^2 I)^{-1}y)^T k(X,x_{k+1}) \tag{2.3}$$

$$k' = k(x_{i+1},x_{i+1}) - K^T(X,x_{i+1})(K(X,X)+\omega^2 I)^{-1}k(X,x_{k+1}) \tag{2.4}$$

In order to budget the number of kernels, we use the sparsification method proposed by Csato et al. [29] which enforces an upper bound on the cardinality of the basis vector and allocates RBFs so as to reduce the regression error. This basis vector set is updated only if the novelty of information shown in Eq. 2.5 for the new incoming data is above some threshold. If the threshold is not superseded, then only the weights and covariance are updated.

$$\gamma = K(x_{i+1},X) - k(x_{i+1},x_{i+1})((K(X,X)+\omega^2 I)^{-1}y) \tag{2.5}$$

In our system we want to built two-stage VGP. The first stage predicts change in the velocity $\triangle V$ when GP input is given time t and behavior b. Then, the second stage predicts the future trajectory (x,y) when GP input is give time t and velocity $V_{\text{initial}} + \triangle V$.

2.7 Markov Decision Process Transition Model

Another model which can be used for storing the information about transitions from one state to another is a Markov transition model. This transition model for Markov Decision Process (MDP) is stored in form of matrix shown in Eq. 2.6.

$$T = \begin{pmatrix} p_{1,1} & p_{1,2} & \cdots & p_{1,j} \\ p_{2,1} & p_{2,2} & \cdots & p_{2,j} \\ \cdots & \cdots & \cdots & \cdots \\ p_{i,1} & p_{i,2} & \cdots & p_{i,j} \end{pmatrix} \tag{2.6}$$

This matrix is a table showing the probability $p_{i,j}$ of transitioning from state i to state j. As it was discussed above, the Markov assumption states that the knowledge about the current state is enough to define the transition to a new state. In this case, we can store exact probability for each state-state transition without averaging over state space, but it is impractical to define every transition in a continuous world. My research

will actually be defining a subset of HMM as a continuous "Markov surface." This gives a spacial resolution for every dimension of the state space and all the small differences between states will be averaged which leads to an uncertainty in the transition probabilities. In the example shown in Fig. 2.7, one exact state given by (x, y, v, t) may result in the transition to some other states with different (x', y', v') tuples, but since the true initial location inside one step is unknown we have some position uncertainty as well.

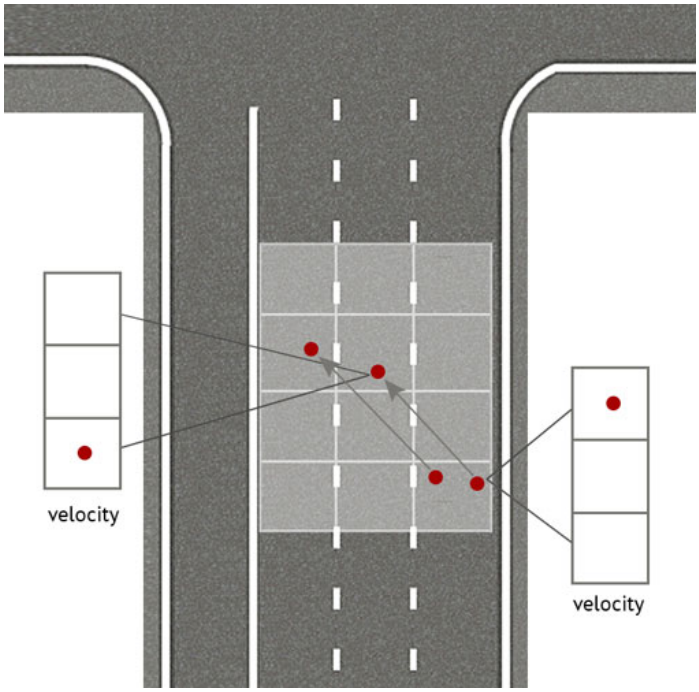

Figure 2.7: Uncertainties in the transitions from one state may or may not result in different states due to uncertainty of the autonomous vehicle's location inside the initial state

The transition probability matrix needs to store a large amount of data, particularly due to the fact that each parameter of the state forms a dimension in the table. For example, since it was chosen to keep a track of 4 parameters with 10 discrete states each, this gives us 10^4 possible input and output states and the transition table with 10^8 elements total. Even using this discrete transition probability matrix is very memory consuming so even though we store the real probability of transitions without approximating it to Gaussian distribution, this is achieved at the expense of responsive or real-time computation. For driving in particular, a focus should be on the generation of real-time solutions for the CAS.

2.7.1 Direct Learning of a Discrete Transition Model

In this thesis, to represent a dynamical state of the autonomous vehicle as a static state we choose 4 parameters: lateral and longitudinal locations on the road, velocity of the vehicle and time. These parameters forms a 4 dimensional set of non-overlapping states. For now, other potential parameters such as acceleration and

the vehicle orientation are left out of the analysis for simplicity. Future work will move towards including these parameters in a computationally realizable CAS.

The resulting state-action transition matrix $T(s,s',a)$ is very large and increases in size with the number of states. For the case considered in this thesis, the set of all states forms 10 x 3 x 10 x 10 matrix, with 3000 initial states and same number of possible states for each of the 10 actions. This lead to a very large dimensional MDP with 90 millions elements (3000x3000x10). It should be noted that the dimensionality of the discretized state-space can be reduced by increasing the range over which the states are discretized, but this leads to other complexities such as high uncertainties in the transition and location.

To learn the ABM as a Markov Transition model, this paper proposes the Monte-Carlo based learning Algorithm 1, where one time step of CAS is divided to 10 incremental time steps equal to 0.1 second. Then the Dynamic Simulation function, described in Section 4.3, simulates the path with these steps and returns the $[x,y]$ data of all 10 steps. This coordinates are linearly applied to all possible initial points $[Loc_x, Loc_y \in Road]$ equally distributed inside the one discrete location state and give the expected paths from these points. The obtained paths are being classified to the discrete states. The numbers of visits to these discrete states by taking one action give the conditional probability distribution of the vehicle inside one time step of the CAS. This process requires a lot of computational work, but the transition matrix T has to be obtained just once, and remains to be the same while dynamic model and parameters of the grid world are still valid.

2.7.2 Indirect Learning of a Discrete Transition Model

It is not wholly practical to train the transition matrix using Monte Carlo simulations since we cannot control other drivers. Learning the transitions can be done directly from observation of other drivers behavior. When the observed transition is translated into grid-world transition, it updates the transition matrix, but this requires that the infinitely large number of transitions observed and registered. Otherwise, such a model would have discontinuities due to the very large state space. For example, for any three discrete states following by each other, the probability of all three states being discovered from 3 observations is 22.2%. Undiscovered states would give a discontinuity in state space and would attract or repel the optimization algorithm from this state depend on the value assigned to this state. For that reason, in this work we prefer to learn a discrete transition model by utilizing the GP model first and then translate this GP model into a probabilistic transition matrix.

Data: Car dynamic model D

Result: Transition model T

for *every action* $a \in A$ **do**

  for *every velocity* $v \in \boldsymbol{R}$ **do**

    x = 0, y = 0, t = 0 ;

    while $t_{inc} \leq t_{CAS}$ **do**

      $[x_n, y_n, v_n, t_n] = D(x, y, t, t_{inc}, v)$;

      $t_{inc} = t_{inc} + t_{CAS}/10$;

    end

  end

end

for $Loc_x, Loc_y, time \in \boldsymbol{R}$ **do**

  $s = [Loc_x, Loc_y, v, time]$;

  $s'_n = [x_n + Loc_x, y_n + Loc_y, v_n, t_n + t]$;

  $T(s, a, s') = \frac{\sum_n (s \to s'_n \in S)}{n}$;

end

Algorithm 1: Direct learning of the Transition Model

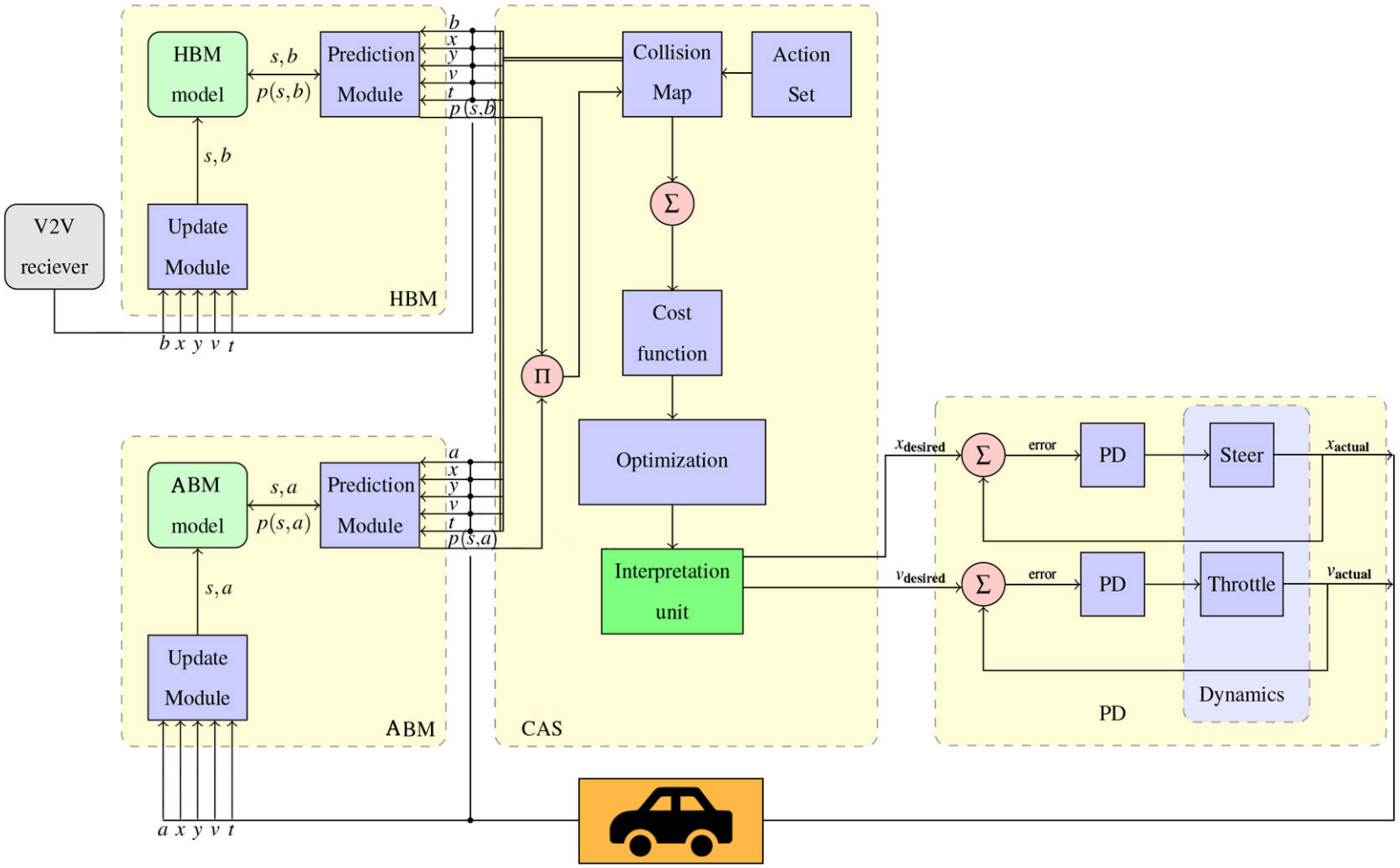

Figure 2.8: Control system overview. Two behavior models predict the future trajectories of each vehicle. These trajectories generate the collision map needed to compute the total probability of collision. The cost function incorporates that probability and additional costs of actions, and is used for optimization to define the best action for this moment. The resulted action is sent to the interpretation unit to get translated into desired parameters of the autonomous vehicle which are applied to the car dynamics by PD controller

CHAPTER 3

CAS algorithm

3.1 Introduction

The previous chapter discusses the behavior models of the human-driven vehicles and autonomous vehicles. This chapter incorporates this model in order to avoid collisions and develops the algorithm which allows to make routine driving safer. By coming back to the statistics in Section 1.2, the advantage of the experienced drivers is a higher propensity to make careful and rational decisions fast; this is attributed to learned behaviors which consistently consider the outcomes of actions and pick the right decision. Such behaviors will help to save lives on the road in dangerous situations and should be a part of an intelligent vehicle. Obviously, people don't compute the outcomes in a consciously numerical manner, but they might assign some risk levels to objects or events, and compare them together to choose the best one. This interpretation makes the data more suitable for human analysis. For artificial intellect, it does not play the key role and this method works even in general case when the numerical values of probabilities are compared.

The proposed system for controlling the vehicle has several levels of control shown in Fig. 2.8 on page 19. The low level control is represented by Proportional Derivative (PD) control which operates the steering, throttle and brake to maintain the desired speed and trajectory on the road. The Collision Avoidance System (CAS) is a high level control algorithm for operating the car by choosing the actions available from the list of simple actions such as changing lane and changing speed, whose main purpose is to decrease the probability of collision. These PD and CAS blocks are interconnected by an Interpretation Unit which translates the desired actions into PD control parameters. Meanwhile, the feedback is given by measurement of the observed state of the vehicle in order to continuously update the current state of the vehicle as well as the probabilistic ABM behavior model. The same update method works for HBM model by monitoring other vehicles behavior and making changes in the model on the fly. These on-line updates allow to build an adaptive control algorithm and adjust the models to any change in the dynamics of the vehicle and the

environment.

3.1.1 Collision Probability

A collision between vehicles happens when two or more vehicles come to the same location at the same time. We let the probability of car c_1 being in state $s \in S(x,y,v,t)$ be defined as a probability $p_{c1}(s)$, and the probability of car c_2 being in state s be defined as a probability $p_{c2}(s)$. For now, we consider these two probabilities are independent since they are caused by intentional actions of driving policies that will not incorporate feedback. They represent just a dynamics of the car with respect to the action taken, but not the action itself. Then, the probability of collision is a conditional probability when both vehicles are in the same location:

$$p(\text{collision in } s) = p(c_1 = s | c_2 = s) = p_{c1}(s) p_{c2}(s) \tag{3.1}$$

For each particular time the probability of collision will be given by:

$$p(\text{collision}) = \sum_S \left(p_{c1}(s) p_{c2}(s) \right), \tag{3.2}$$

while the total probability of collision will be:

$$p(\text{collision}) = \int_{t=0}^{t=t_{max}} \int_{x=x_{min}}^{x=x_{max}} \int_{y=y_{min}}^{y=y_{max}} \left(p_{c1}(x,y|t) p_{c2}(x,y|t) \right) dx\, dy\, dt \tag{3.3}$$

where t_{max} is time horizon and $x, y \in$ Road Space.

If there are multiple human-driven cars in the area surrounding the autonomous vehicle, the total risk of collision will take the following form:

$$p(\text{collision}) = \sum_S \left(p(c_1 = s | c_2 = s \text{ OR } c_3 = s \text{ OR } c_n = s) \right) \tag{3.4}$$

$$= \int_{t=0}^{t=t_{max}} \int_{x=x_{min}}^{x=x_{max}} \int_{y=y_{min}}^{y=y_{max}} \left(p_{c1}(x,y|t) \sum_{c_2}^{c_n} p_{c_i}(x,y|t) \right) dx\, dy\, dt \tag{3.5}$$

where p_{c_i} is probability of the human-driven car $c_i \in c_2..c_n$ being in state (x,y,t) assuming that there is only one human-driven car that can occupy this state at the time. In other words, we assume that there is no collision between human-driven cars.

3.1.2 Optimization Formulation

Comparison of the outcomes caused by the choice of action is a classic optimization problem in mathematics whose goal is to find the best solution from all feasible solutions. The standard form of optimization is the task of minimizing the cost function J. This cost function represents the probability of collision and has additional parameters which will allow us to define preferences in actions, location on the road and potentially preferences for other behaviors. This thesis proposed to use of a penalty C for each action according to their preferences in addition to using probability of collision as a penalty for the cost function.

The optimization problem is shown in Eq. 3.6:

$$a = \arg\min_{a \in A}(J) \tag{3.6}$$

where the cost function is:

$$J = \sum_{T} \sum_{x',y'} \left[P_a(x',y'|x,y,v,t,a,A) P_h(x',y'|x,y,v,t,b,B) \right] + C(x',y',v',a) \tag{3.7}$$

The additional cost C is associated with the cost of action in each particular situation such that:

$$C(x',y',v',a) = \text{Cost}(a) + \text{Penalty}(v') + \text{Penalty}(x',y') \tag{3.8}$$

where Cost(a) is a cost of the action itself according to the rank of preferences (less annoying actions have less cost), Penalty(v') is a penalty for driving with the speed different from the one desired by the passenger, Penalty(x',y') is a penalty for being off-road to motivate the car follow the road.

$$\text{Cost}(a) = \begin{cases} C_{a_1}, \text{if } a = 1. \\ C_{a_2}, \text{if } a = 2. \\ C_{a_n}, \text{if } a = n. \end{cases} \tag{3.9}$$

$$\text{Penalty}(v') = ||V_{\text{desired}} - v'||P_v \tag{3.10}$$

$$\text{Penalty}(x',y') = \begin{cases} P_{\text{out}}, \text{if } x',y' \notin \text{Road}. \\ 0, \text{if } x',y' \in \text{Road}. \end{cases} \tag{3.11}$$

where C_{a_n}, P_v, P_{out} are manually defined penalty coefficients for constrained optimization problem.

3.1.3 Primitive Action Control Algorithm

Primitive actions control the intelligent vehicle in order to maintain its velocity and lateral position on the road. Assuming that we know the vehicle's location relative to the center of each lane and the desired velocity,

the control system may use a PD controller as shown in Fig. 3.1.3 to reduce the difference between desired signal and the real one.

$$\text{Throtle} = K_{p_\text{T}}(V_\text{desired} - v_\text{actual}) + K_{d_\text{T}}(\triangle v_\text{actual}) \tag{3.12}$$

$$\text{Steer} = K_{p_\text{S}}(x_\text{desired} - x_\text{actual}) + K_{d_\text{S}}(\triangle x_\text{actual}) \tag{3.13}$$

where proportional K_p and derivative K_d gains have been tuned individually for each control signal in order to inhibit oscillations and perform a smooth transition from state to state.

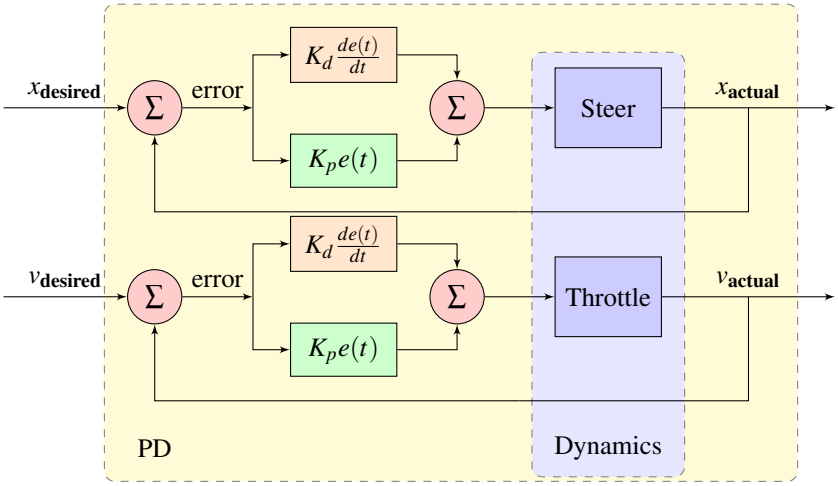

Figure 3.1: Proportional-derivative (PD) controller for low-level control of the autonomous vehicle is developed to follow the desired trajectory and velocity.

3.2 Single-Step GP-based Collision Avoidance

The task of minimizing the danger and the discomfort of travel associated with maneuvers and speed change as well can be easily solved by taking immediate action without planning the motion far ahead. This approach is as much reliable as a sensor based reactive collision avoidance, but also works with stochastic trajectory prediction, gives a better flexibility in costs and utilizes wide action space.

Single step collision avoidance is assuming to plan just one action ahead. It picks the best action minimizing the cost and assumes it will keep taking this action for all time horizon. However, this does not mean that it would use this action, because the decision can be changed very fast. Due to the fact that to find the best action the algorithm has to evaluate all possible actions just once, the solution is obtained very fast. Since it does not require to perform the optimization process in continuous world, the total number of cost function evaluations in this case is equal to number of actions.

3.3 Sequential Markov Decision Process-based Collision Avoidance

Sequential collision avoidance is a complex algorithm allowing to plan several action ahead. It does not solving the immediate problem, but also allows to find the best strategy represented by a sequence of actions. In this case, it allows to use complex actions which are unable to use by simple algorithm.

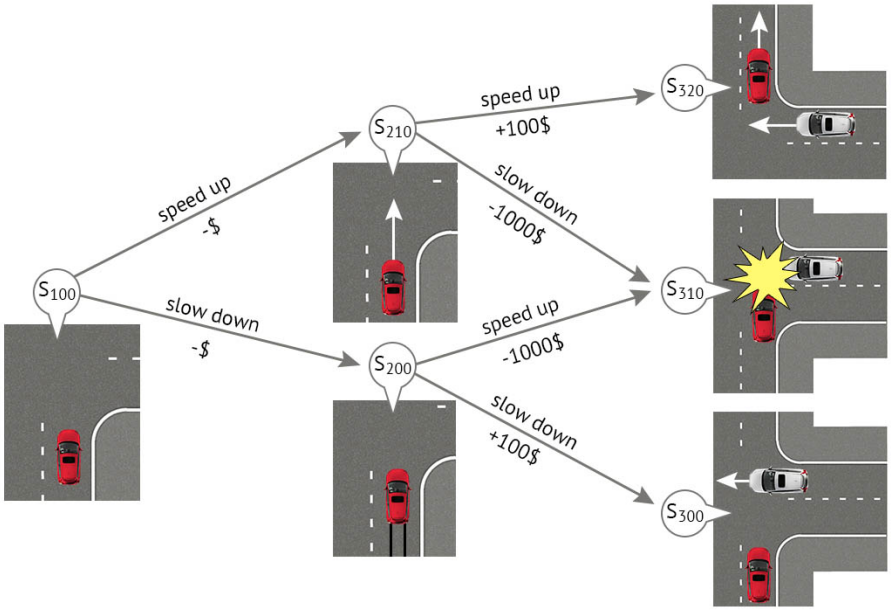

Figure 3.2: An example of MDP formulation showing that some actions lead to the collision state. These actions should be marked by highly negative reward (penalty).

In this section, we formulate the proactive decision making problem as an optimization problem. For this purpose, the autonomous collision avoidance task is posed as an MDP tuple (S,A,T,R) that captures the Markovian transition of the car in the real world [30, 31]. Here, S is the set of discrete states of the car, A is the set of desired actions, $T(s,a,s')$ is the transition model from any state $s \in S$ to any other state $s' \in S$ when the action $a \in A$ is taken, and denotes the conditional probability of transition $p(s'|a,s)$. R is the model of the reward obtained by the transition (s,a,s'). The value of each state is given by the value of the next state discounted by the discount factor γ and the cost of transition and mathematically described by the Bellman equation:

$$V(s) = \max_{a \in A} \left(\sum_{s' \in S} T(s,a,s')(R(s,a,s') + \gamma V(s')) \right) \tag{3.14}$$

The optimal policy $\pi^*(a)$ is the set of action for each state that maximizes the expected discounted reward:

$$\pi^* = \arg\max_{\pi} E\left(\sum_{s\in S}(R(s,a,s')|\pi)\right) \tag{3.15}$$

There are many approaches of solving MDPs, some of which were surveyed in the recent papers [31, 25]. The value-iteration algorithm has been chosen due to its convergence guarantees.

Data: Transition model T, Reward model R

Result: Optimal policy π^*

while $\triangle > \eta$ **do**

  for $s \in S$ **do**

    $v = V(s)$;

    $V(s) = \max_{a\in A}\left(\sum_s T(s,a,s')(R(s,a,s') + \gamma V(s'))\right)$;

    $\pi(s) = \arg\max_{\pi}\left(\sum_s T(s,a,s')(R(s,a,s') + \gamma V(s'))\right)$;

    $\triangle = \max(\triangle, |v - V(s)|)$;

  end

end

Algorithm 2: Value-iteration algorithm

Decision making Algorithm 2 for the CAS is based on the Bellman function shown in the Equation 3.14. We calculates the vector $V(s)$ of the maximum values of state s using the $T(s,s',a)$ and $R(s,s',a)$ matrices with respect to probability of the transition from this state to any resulting state and the cost of this transition. The output matrix $P(s)$ gives the best policy of actions. When the allocation of the penalty states in the matrix R is known, we have a map of actions for any state of the autonomous car, regardless of where it had really been. This policy is relevant only for the specific location of penalties or distribution of the reward at the space. We could say that, regardless of other factors, once calculated policy should fit to any similar distribution of the rewards. By that, there is no need for constantly calculating the policies on-line, they could be precomputed in advance and stored as ready-made solutions in the database what let to save the time of calculation. The frequency of the decision making algorithm has been set to 1 Hz (once every second). Therefore, after each decision the autonomous car continued to go by inertia for 1 second, until the next action is computed based on the evaluation of the environment.

3.3.1 Designing the Reward Function

The reward function is designed in a similar way to the cost function for the single-step optimization and shows the autonomous car which states should be followed. We give a large negative reward to the collisions, or to be more precise the states in which collision happens. To motivate the autonomous car move towards

the intersection, the states at the other side of the intersection get the positive reward. All other states obtain the reward according to the cost of actions shown in the Table 3.1. This formulation provides a great degree of flexibility in defining the priorities of actions and states.

Table 3.1: Action's descriptions and penalties

N_A	Description of action	Penalty
1	Keep going	0
2	Soft Speed up	0
3	Soft Slow down	0
4	Soft Merge left	0
5	Soft Merge right	0
6	Emergency stop	-100
7	Speed up	-20
8	Slow down	-20
9	Merge left	-30
10	Merge right	-30

The set of actions can be decomposed into two main subsets: so called soft actions and hard actions. The soft actions are shown in Table 3.1 with number 1 to 5. Because of their smoothness and passengers-friendliness, they were grouped as a preferred actions and defined as zero-cost actions. The firm actions with numbers 6 to 10 in Table 3.1 are rough actions which were used when the soft actions were not sufficient to prevent the collision with the costs defined accordingly to their preference. The durations of all actions were identical and defined by the time-step of the CAS algorithm equaled to 1 second.

$$R(s, s'_{\text{collision}}, a) = -10000 \tag{3.16}$$

$$R(s, s', a) = \text{Cost}(a) \tag{3.17}$$

CHAPTER 4

Simulations and Results

4.1 Introduction

Since the main purpose of the proposed collision avoidance system is a cooperation with the environment and other human-driven cars, it is highly important that the drivers behave in the similar way as a real human does. The use of real cars to prove the work of the system would be not only dangerous, but also requires a special enclosed area and approval from the authorities. On the other hand, driving a scaled car model on the experimental car testbed would not give the human driver the right feelings due to the difference in the car dynamics which makes the results not reliable. These reasons lead us to use of a computer simulation for examination of the CAS algorithms. Such a method requires to utilize an environment of the considered scenarios, create a dynamical model of a car, learn transition rules for the list of actions over dynamical simulations, learn transition rules of the car controlled by human using a steering wheel and test the real-time driving cooperation with real human drivers. Two kinds of simulations were used: an individually developed simulation algorithm and the third-company developed software. We have designed the first utilizing the Matlab computing environment as a three lane highway with an intersection where all autonomous and human-driving vehicles were involved. This algorithm gives the full flexibility in modification of the code and parameters in addition to full Matlab functionality for data analysis. The second simulation utilizes the Carnetsoft driving simulator [32]- the professional grade car simulator with a highly realistic 3D view from the cabin what gives the real feelings of driving that is especially important in the human-driving data collection.

4.2 Matlab Simulation Description

To prove the viability of the concept the computer simulation has been built to examine the driving of an autonomous vehicle when both autonomous and human-driving vehicles are involved. To consider two general-

ized cases of the problem - driving on the highway and through the intersection, the multipurpose simulation was built.

In the highway scenario, the autonomous car shown by red color rectangle in Fig. 4.1 is moving from south to north while manually controlled vehicles shown by blue color are following the same direction. This script examines the cooperation of the autonomous vehicle with others in parallel driving and checks its ability to evade from their dangerous maneuvers.

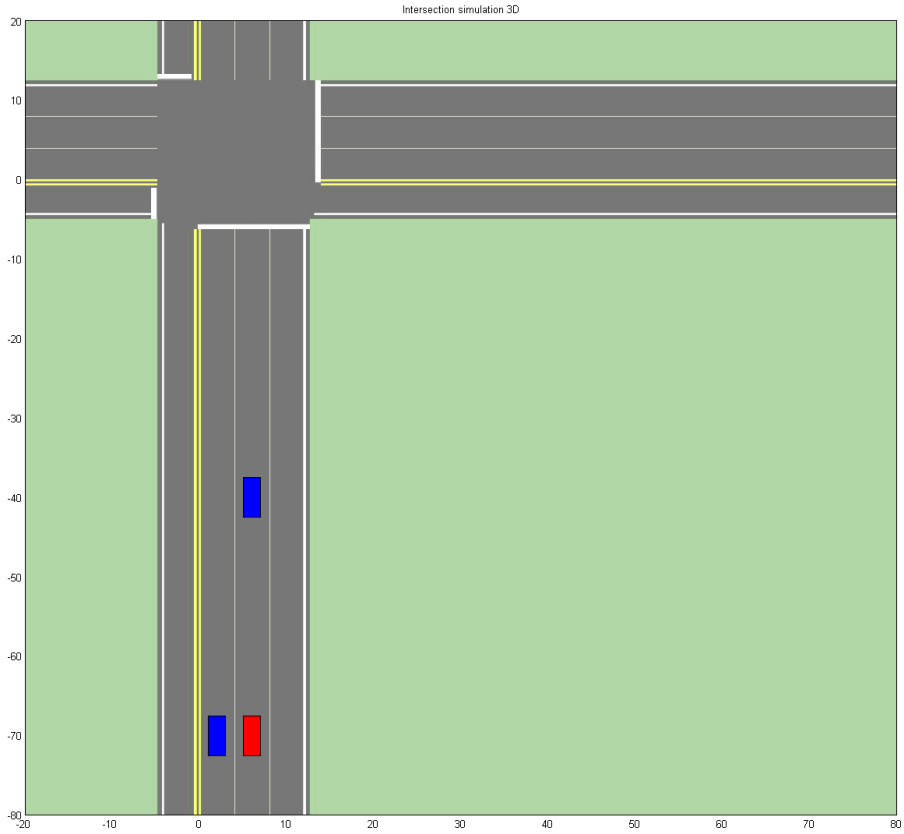

Figure 4.1: Built in Matlab simulation environment during the highway scenario. The autonomous car (red) driving on the highway in the same direction as other human-driven cars (blue).

Intersection scenario represents the special case when the autonomous car is moving in transverse direction to the general traffic as shown in Fig. 4.2. This simulation investigates the autonomously driving vehicle shown by red rectangle passing the intersection where green, blue and yellow rectangles represents the human-driving cars.

The simulation algorithm Alg. 3 shown on page 29 has been built for generalized case satisfying both scenarios to utilize the dynamical equations of all vehicles and update the vehicle's positions with a time interval of 10 ms. The short update interval guarantees the elimination of a possibility of skipping discrete states and avoids "jumping" one vehicle over another.

Data: Transition model ABM, Behavioral model HBM, Dynamic function D

Result: Result of collision

$car_n = [x_n, y_n, v_n], t = 0$;

while $y \leq y_{final} \in \boldsymbol{R}$ **do**

  $[x_n, y_n, v_n, t_n] = D_n(x_n, y_n, v_n, t_n), n = [0..N_{\text{cars}}]$;

  if $t \propto t_{CAS}$ **then**

    $S_{collision}(n) = S(Agent \perp car_n)$;

    $S_{collision} \rightarrow R$;

    if $R \neq R_{prev}$ **then**

      $\pi = CAS(x, y, v, t, T, R)$;

    end

    $a_{n=0} = \pi(s)$;

  end

  switch *Human behavior model* **do**

    case *1*

      $v_{n=1..3} = \text{Gaussian}(v_n)$;

    end

    case *2*

      $v_{n=1..3} = \text{take action } a \in A$;

    end

    case *3*

      $v_{n=1..3} = \text{load } 'HBM.model'$;

    end

  endsw

  $a_{n=0..3} \rightarrow D_n$;

  $t = t + 0.01$;

end

Algorithm 3: Simulation algorithm

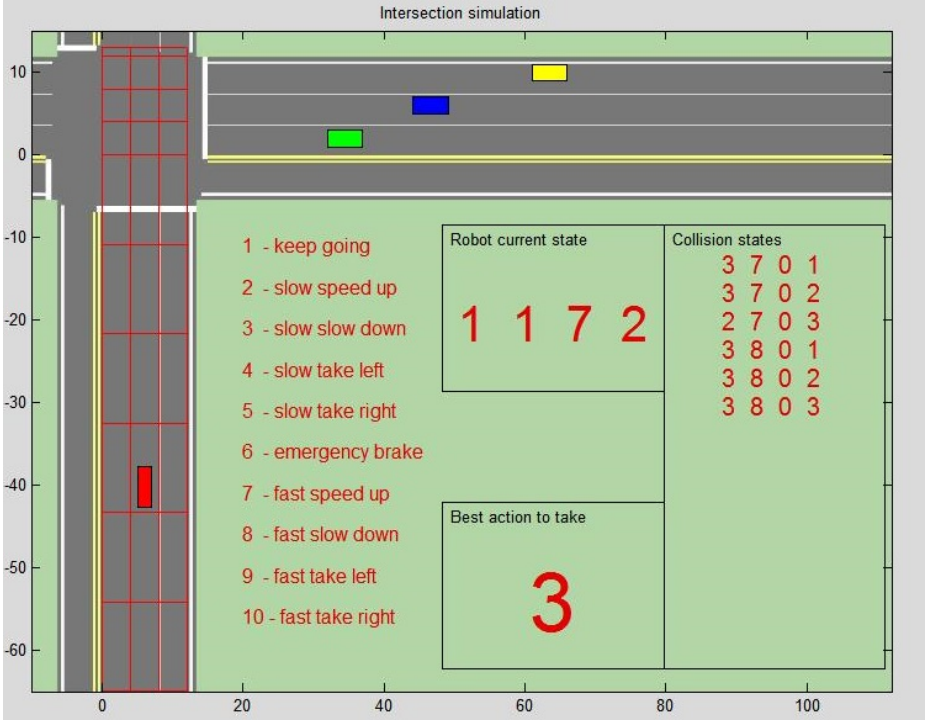

Figure 4.2: Built in Matlab simulation environment during the intersection scenario. The autonomous car (red) driving on the highway in the transverse direction to other human-driven cars (blue,yellow,green). The red grid represents discrete location states used in sequential optimization only.

4.3 Dynamic Model of a Vehicle

In order to simulate the dynamics of a car, a simplified dynamical model of the Dubin's car has been described by the equations of motion based on the dynamic vehicle model [1]. It used six parameters to describe the real vehicle and environment:

m : Mass of vehicle [kg]

a : Distance from front axle to Center of Gravity [m]

b : Distance from rear axle to Center of Gravity [m]

C_x : Longitudinal tire stiffness [N]

C_y : Lateral tire stiffness [N/rad]

C_A : Air resistance coefficient [1/m]

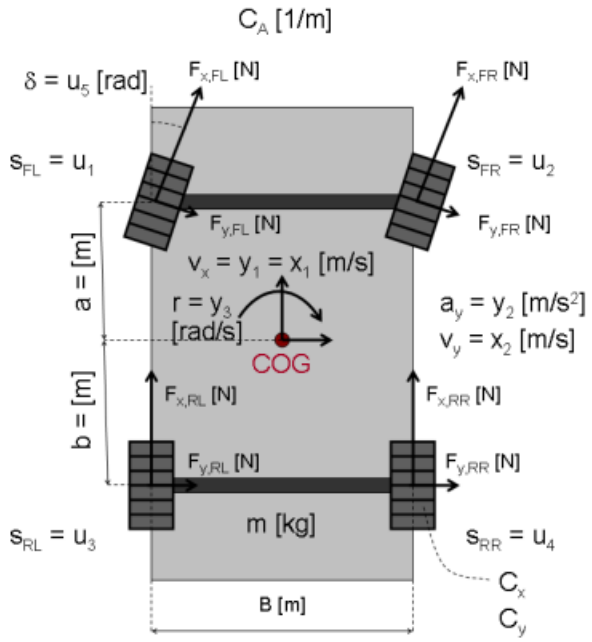

Figure 4.3: Schematic view of a vehicle dynamics system [1].

In this simulation, we chose coefficients according to the Volvo V70 model as followed, $m = 1700$, $a = 1.5$, $b = 1.5$, $C_x = 150000$, $C_y = 4000$, $C_A = 0.5$. Three states of the model were taken into consideration:

$$x_1(t) = v_x(t) = \text{Longitudinal velocity [m/s]} \tag{4.1}$$

$$x_2(t) = v_y(t) = \text{Lateral velocity [m/s]} \tag{4.2}$$

$$x_3(t) = r(t) = \text{Yaw rate[rad/s]} \tag{4.3}$$

where $v_x(t)$ and $v_y(t)$ represented longitudinal and lateral velocity. $r(t)$ was the yaw rate at time t. The state-space structure of the model was illustrated by the following differential equations:

$$\begin{aligned}
&\frac{dx_1(t)}{dt} = x_2(t) \times x_3(t) \\
&+ m^{-1} \times [C_x \times (u_1(t) + u_2(t)) \times cos\,(u_5(t)) \\
&- 2 \times C_y \times \left(u_5(t) - \frac{x_2(t) + a \times x_3(t)}{x_1(t)} \right) \times sin\,(u_5(t)) \\
&+ C_x \times (u_3(t) + u_4(t)) - C_A \times x_1(t)^2]
\end{aligned} \tag{4.4}$$

$$\frac{dx_2(t)}{dt} = -x_1(t) \times x_3(t)$$
$$+ m^{-1} \times [C_x \times (u_1(t) + u_2(t)) \times sin(u_5(t))$$
$$+ 2 \times C_y \times \left(u_5(t) - \frac{x_2(t) + a \times x_3(t)}{x_1(t)} \right) \times cos(u_5(t))$$
$$+ 2 \times C_y \times \frac{b \times x_3(t) - x_2(t)}{x_1(t)} \Big] \tag{4.5}$$

$$\frac{dx_3(t)}{dt} = \frac{1}{(0.5 \times (a+b))^2 \times m} \times$$
$$\{a \times [C_x \times (u_1(t) + u_2(t)) \times sin(u_5(t))$$
$$+ 2 \times C_y \times \left(u_5(t) - \frac{x_2 + a \times x_3(t)}{x_1(t)} \right) \times cos(u_5(t)) \Big]$$
$$- 2 \times b \times C_y \times \frac{b \times x_3(t) - x_2(t)}{x_1(t)} \Big\} \tag{4.6}$$

Solving these ordinary differential equations (ODE) (Eq. 4.4 – 4.6) explicitly was difficult. However, Runge-Kutta method [33] provided a numerical solution for the state of the vehicle(velocity, acceleration and yaw rate) in every iteration.

4.4 Carnetsoft Simulation

The simulation algorithm discussed in Section 4.2 allows to control the human-driven car with the view from the top what is difficult and gives the wrong feelings of driving. Even experienced driver cannot control the car without additional training; this makes worthless all the previous driving experience obtained from the real driving. Since the general idea of this work is to utilize driving experience for behavior model training - convenience of the driver is highly important for natural driving. For that purpose, RijSchoolSimulator developed by Carnetsoft has been chosen. This simulator utilizes Logitech G27 control set, 4 monitors and software what gives the view from the cabin in front and side directions as shown in Fig. 4.4.

This driving simulator gives an excellent tool to investigate driving-related scientific questions. This software allows to prepare and perform behavioral experiments, and analyze the data. This simulator has been widely used in studies on the effects of alcohol, distraction, drowsiness on driving and driver behavior modeling studies [34]. Moreover, this software provides a database of the objects and tools to develop a new and modify the existed road maps. It samples the position of steering wheel, gas and brake pedals with 10 Hz frequency and utilizes its build-in dynamic functions to update the parameters of the cars. The graphic abilities of Carnetsoft's simulator allows to render 3D world and reproduce night driving, rain, snow effects and sound effects. Its script language is designed to create any scenario, generate traffic and manage the data.

Figure 4.4: Carnetsoft's simulator utilizes 3 monitors for realistic panoramic view from the cabin and 1 monitor for setup and simulation parameters while the steering wheel set Logitech G27 controls the human-driven vehicle. The autonomous vehicle is controlled by a separate computer using Ethernet connection and can be seen from the side only.

This realistic car simulator allows to call the instinct driving skills of the human doing this everyday task without additional training and get the results as much close to the real travel as it is possible. To make this training of the behavior models possible and control the autonomous car, Carnetsoft is connected to another computer with MATLAB algorithm via UDP connection. The data required for CAS system are streamed to the computer, transformed into variables, processed and then used for training of behavior models. These data are also used in decision making process in order to find the best corrective action what will be send back to the Carnetsoft simulator as shown in Fig. 4.5.

4.5 Training Behavior Models

In this work, to avoid the possibility of faults caused by human intention recognition algorithm, the driver expressed his intentions by using the switches on the steering wheel in the same manner as turning signals are used on the road. It was assumed that the intention was always classified correctly and did not affect on the training process of Human Behavior model (HBM). All intentions were extracted from the continuous driving on-the-fly and formed the training vector with zero initial location what followed the Markov assumption.

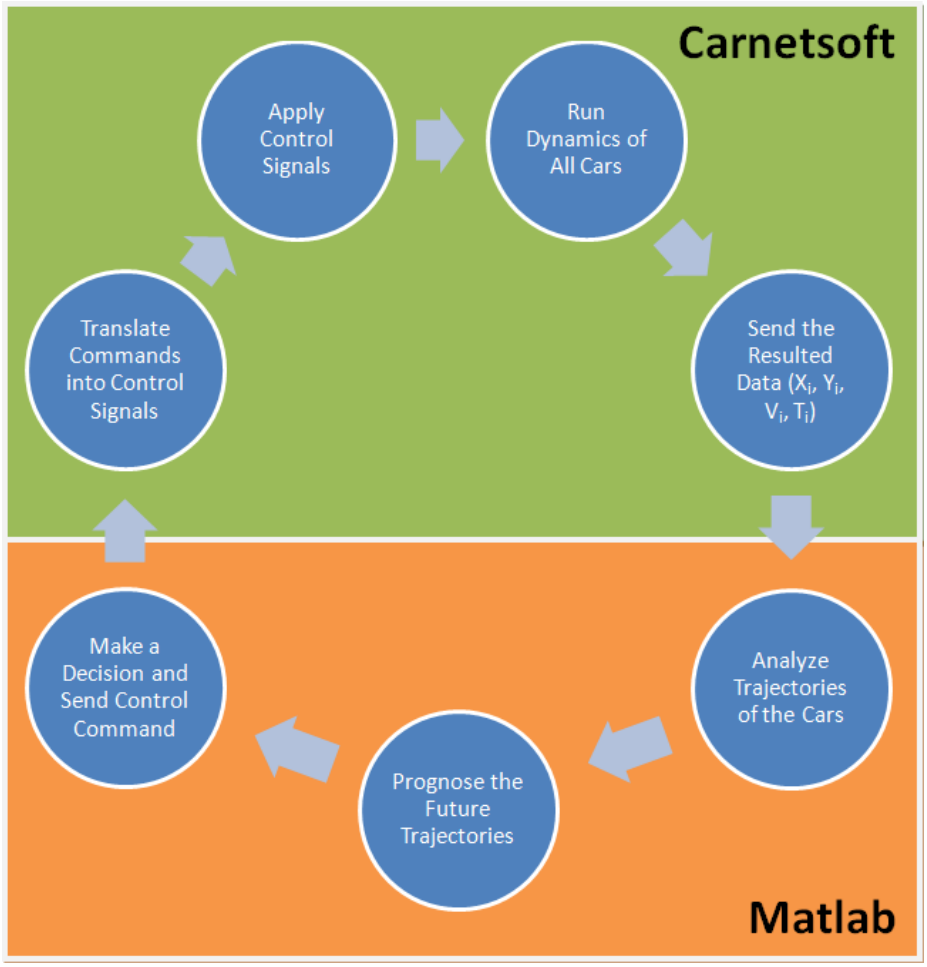

Figure 4.5: Carnetsoft's simulator applies the control to the vehicles and updates their dynamics and locations. The control signals are given by steering wheel for human-driven car and desired actions for autonomous vehicles. The proposed collision avoidance algorithm runs on the separate computer in Matlab environment. The data are transferred between computers using UDP connection.

This method allowed to update the behavior models on-line and adapt to a change of the environment. Five intentions such as changing lane to the left/right, slowing down, speeding up and keep going were collected from the training of the HBM model and shown in Fig. 4.6, where thin doted lines represented the collected location data of the car controlled by the human in Carnetsoft's simulator, while the lines of circles showed the prediction made by HBM utilizing the VGP predictor in Matlab.

The ABM model, which represents a trajectory of the autonomous vehicle, has been trained by the Monte-Carlo method where all possible actions from the action set were applied to the simulated autonomous vehicle. The uncertainty in transition was caused by the flexibility in the initial location of the autonomous car and the uncertainty in control applied to the dynamics of the vehicle as well as the uncertainty in all simplified parameters such as acceleration and steering angle. For that training Matlab script was sending control actions to Carnetsoft's simulator as the CAS system would do that to control the car. After that, the

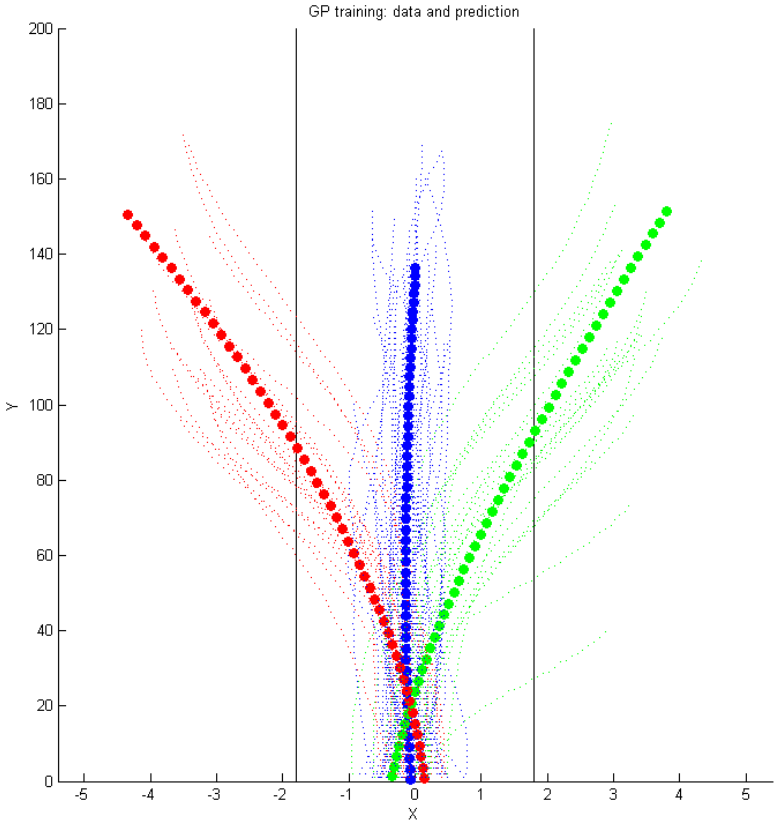

Figure 4.6: Predicted trajectory for 5 seconds ahead made by HBM shown in circle marker lines. Actual driving is shown in thin dotted lines. Color represents the intention (red - merge left, blue - keep the lane, green - merge right)

simulator sent back a vector of readings similar to the one used for training HBM. The data were collected and extracted from these vectors to train the VGPs of ABM model. The estimated trajectory given by VGP predictor is shown in Fig. 4.7. These ABM and HBM trajectory estimators were used by the CAS to define the probability of collision.

As we described in Sec. 3.3, the sequential collision avoidance algorithm utilized various compositions of actions. This required the use of transition matrices to speed up the process of the computing the cost function and made it difficult to visualize the solution as it had been done for the single-step algorithm. The transition matrix was formed using a discretization of the estimated trajectories made by the GP. This allowed to get a normal distribution of the probabilities and a smooth transition matrix without discontinuities caused by individual transitions. The example of that transition model is shown in Fig. 4.8.

The possible states of each action are shown in Figure 4.8 on page 37 in tonal gradations with respect to its probability. As can be seen, this probability was depended not only of the selected action, but the vehicle's

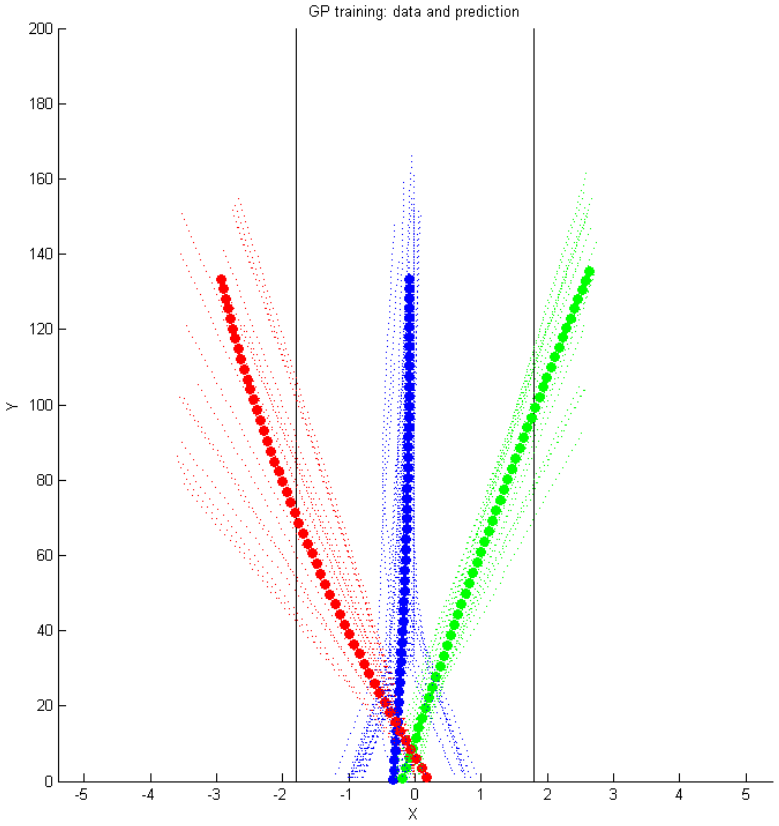

Figure 4.7: Predicted trajectory for 5 seconds ahead made by ABM shown in cross-marked lines. Actual driving is shown in thin dotted lines. Color represents the action (red - merge left, blue - keep the lane, green - merge right)

speed and location on the roadway as well. This transition represented the possible location of the vehicle during the next 1 sec. That makes the uncertainty of location higher if the velocity of the vehicle is higher, since the distance driven in one second is larger.

4.6 Collision Map

Single-step CAS algorithm, discussed in Sec. 3.2, utilized the GP predictor of the HBM to create the normal distributions of possibly occupied locations over the space. Due to the limitations of this work, we considered only one intention of the human in order to make a prediction. Future works, which cooperate the work of the intention classification algorithm, would consider a mix of intentions. This mixture would incorporate all intentions of a human with respect of their probabilities and let the autonomous vehicle to evade if the classification made with low confidence. The use of both ABM and HBM predictor gave us the two Gaussian

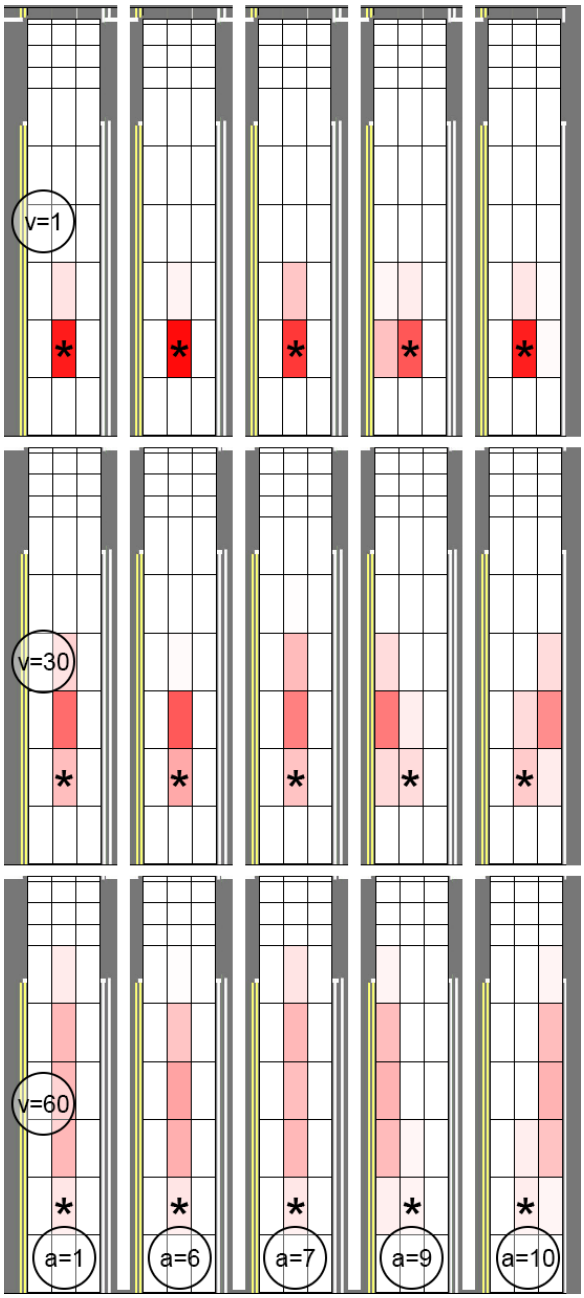

Figure 4.8: Transition model for actions: 1- keep going, 6- emergency brake, 7- speed up, 9- turn left, 10- turn right and speeds 1, 30, 60 mph. The probability of transition from the state marked by (*) is shown in gradations of red color.

distributions of possible locations of both autonomous and human-driven cars as shown in Fig. 4.9. In this figure, the prediction was made with 1 second interval for 0 to 5 seconds time-horizon. The distributions showed the probability of occupying the state $[x, y, t]$ by human-driven car (blue distribution) and autonomous car (green distribution). The intersection of two normal distributions as shown in Fig. 4.10 in red color represented the probability of collision with respect to the location and time or so called the collision map

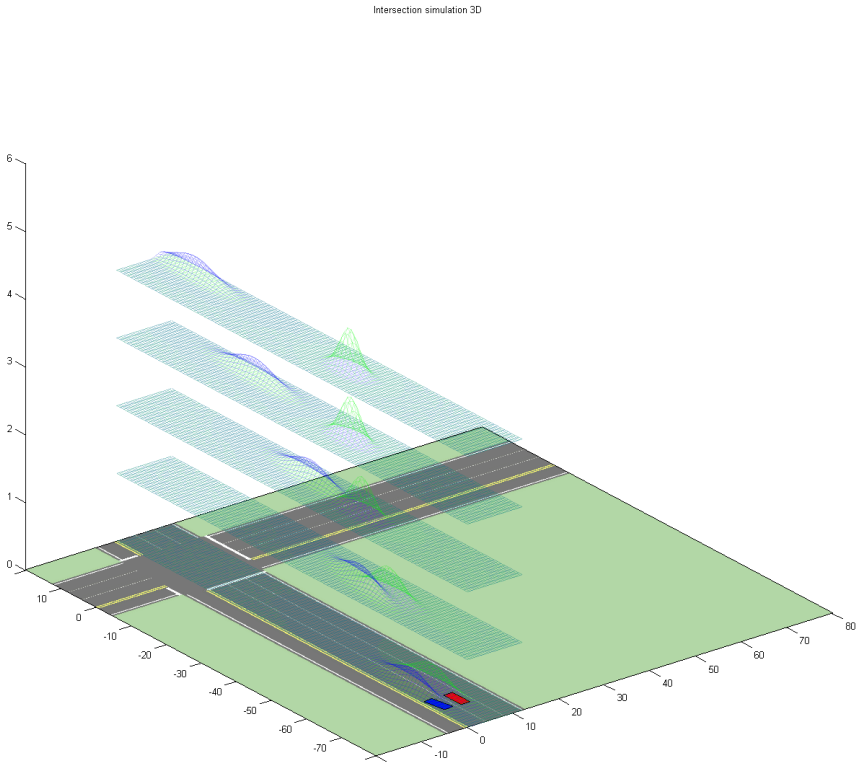

Figure 4.9: Prediction for 5 seconds ahead. Red car is autonomous, blue is human driven. Example is shown when the human intends to merge right

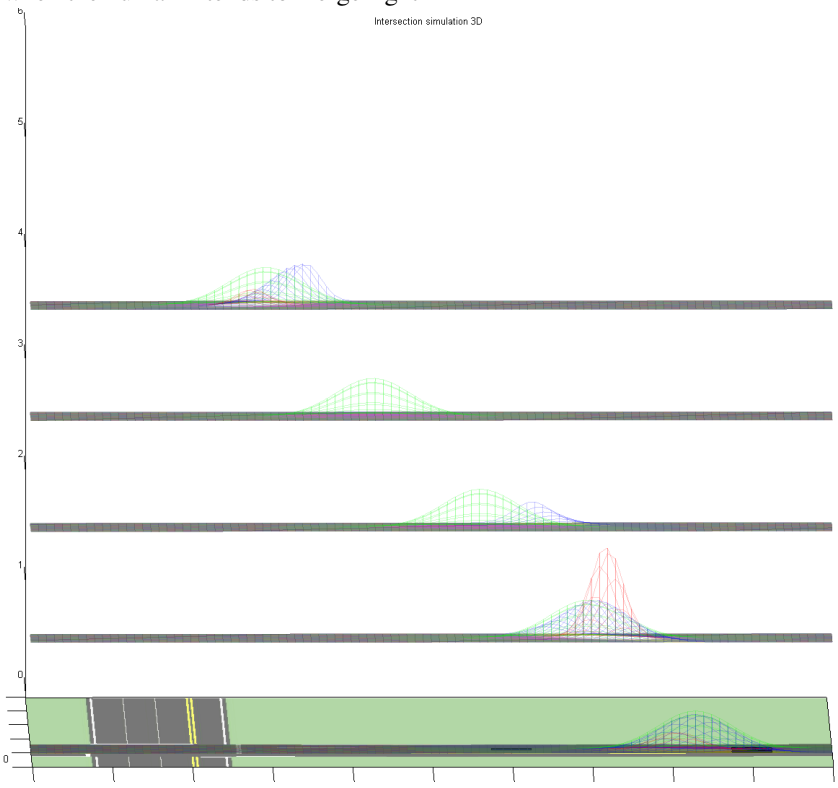

Figure 4.10: Probability of collision shown in red in $\{x, y, t\}$ plane for a unique action

required for the cost function programming.

4.7 Evaluation

To prove the work of the CAS, we designed two types of experiments: a qualitative experiment investigated the real-time behavior of the autonomous vehicle cooperating with the real human driver; a quantitative experiment reproduced the initial conditions for bunch of simulations and collected the statistical data.

In the qualitative experiment different human-drivers were driving using Carnetsoft simulator trying to reduce the distance to the autonomous car by using brake in front of it, chase it or merge to its lane. The autonomous car reacted by changing its velocity and lane. The example of such simulation is shown in Fig. 4.11 in form of trajectories resulted by the vehicles.

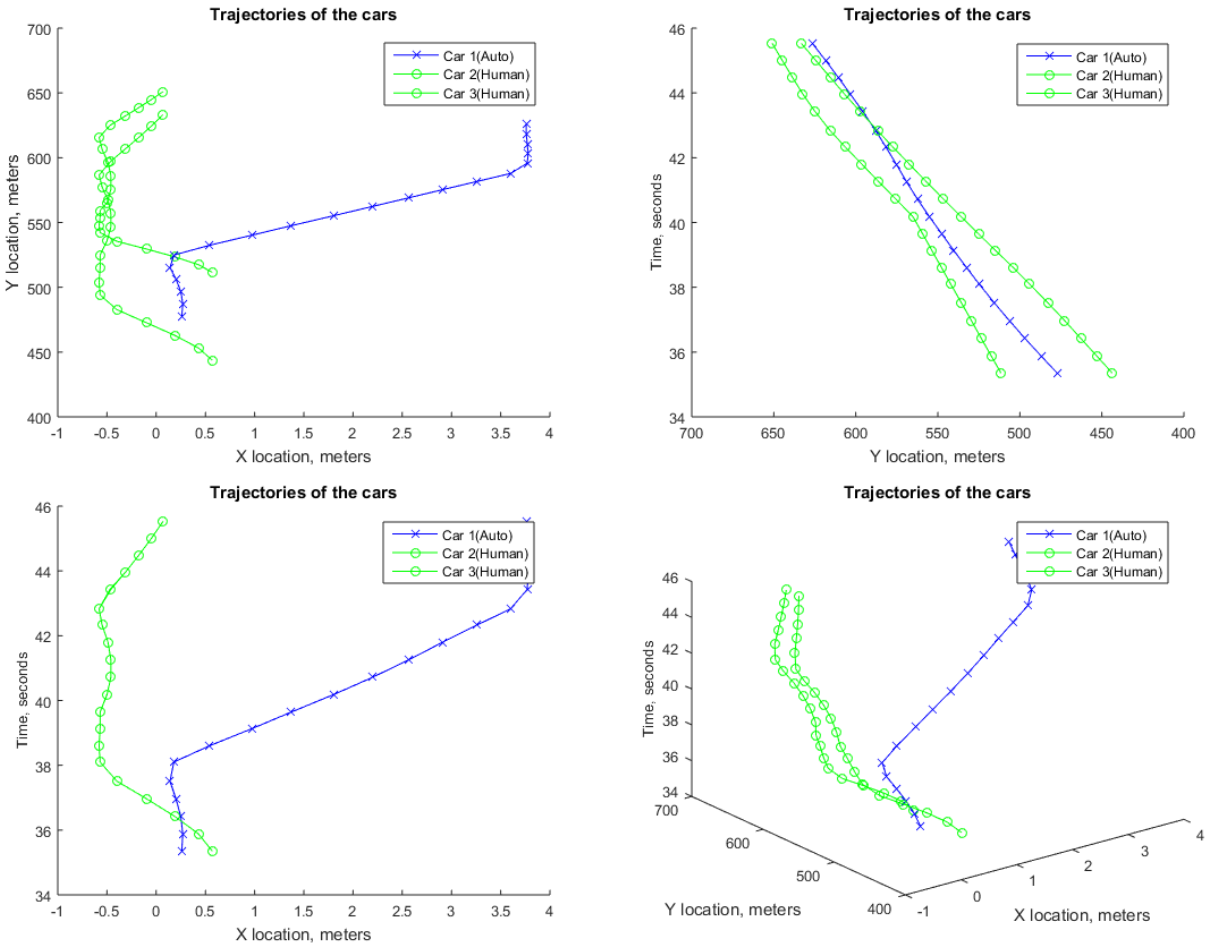

Figure 4.11: Trajectories resulted by a cooperation with a real human in the Carnetsoft simulation in the parallel driving scenario. Two human-driven cars resulted the green trajectories while the autonomous car resulted the blue one. Trajectories represented the overpass maneuver taken by the autonomous vehicle.

An investigation of two general scenarios such as parallel driving and intersection crossing required to perform quantitative simulations. For that reason the simulations were unified to some specific setup which allowed to compare the results and perform statistical analysis. The quantitative experiments were performed in the Matlab simulation algorithm described in Section 4.2. Three role-models were created to simulate a human-driving car. The first one reproduced driving with "the constant speed" by a human driver without any intention. The car was given some initial velocity while its further speed was defined according to the Gaussian probability distribution of the velocity in the previous step. The second model emulated a random selection of the behavior every second from the list of all intentions in the HBM. It reproduced an intentional action of a driver while driving. The third model was using the actual human-driving. For this purpose, the data were obtained from the driving with the use of the Logitech G27 steering wheel and pedals to control the model of the car. To avoid a possible computational delay in calculating of the solution, the human driving was not executed in real time. The pure data resulted by the human intention were saved to a data file and reproduced by steps during CAS simulation. Thus, when the CAS was calculating the solution, the manual driving vehicle stopped until the calculations got finished. This allowed us to simulate the interaction with the real drivers as close as possible. It should be noted that none of these models performed the actions in the aggressive manner aimed to commit an intentional crash.

4.7.1 Quantitative Results of the Intersection Scenario

In the intersection scenario the simulations provided the data sufficient to compare the work of reactive and proactive systems during 100 trials with 8 simulations each including 2 different initial velocities 30 and 60 miles per hour and the presence of one and two human-driving cars with the Gaussian distribution of the speed. This quantitative simulation did not consider random-action and real-human models due to difficulties in the comparison. In all cases, there were obtained no car collisions and the significant improvement in the travel time through the intersection in contrast to the reactive systems.

Fig. 4.12 shows the velocities of the autonomous vehicle (denoted as car1) and the human driving car(denoted as car2) moving in transverse directions. Both human-driving and autonomous cars had initial velocities 30 mph (14 mps, shown at top figure) and 60 (28 mps, shown at bottom one). As can be inferred from the figure, the time required to pass the intersection for the proactive algorithm is less(6.1 and 4.5 seconds) than for the reactive algorithm(7.1 and 9.8 seconds). The actions performed by the proactive system were smoother and required less change in the speed what gave less discomfort to the passengers. The cases considered two human-driving cars are shown in Fig. 4.13. In all simulations the travel time was less for the proactive system for 25-30% and the autonomous vehicle avoided a complete stop in most cases when the

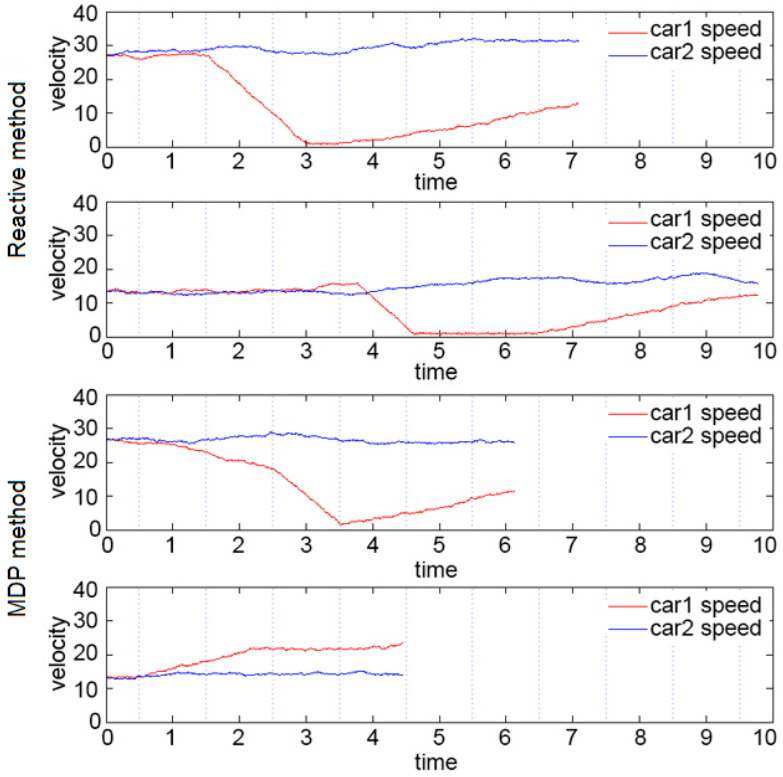

Figure 4.12: Autonomous vehicle('Car1') and human('Car2') velocities in random example, simulation stops when the autonomous vehicle pass intersection.

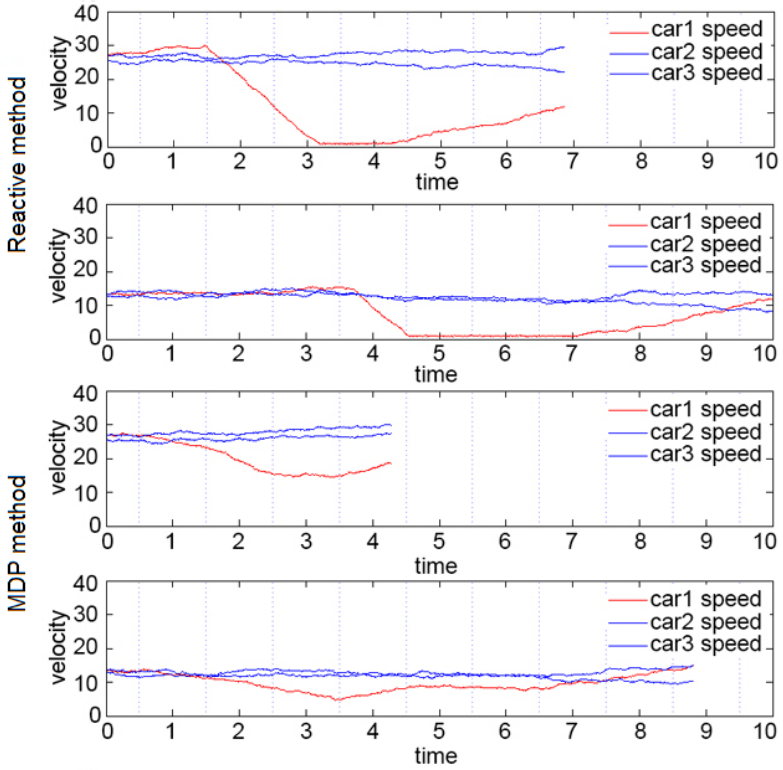

Figure 4.13: Autonomous vehicle('Car1') and human('Car2' , 'Car3') velocities in random example, simulation stops when the autonomous car pass intersection.

use of soft actions was enough.

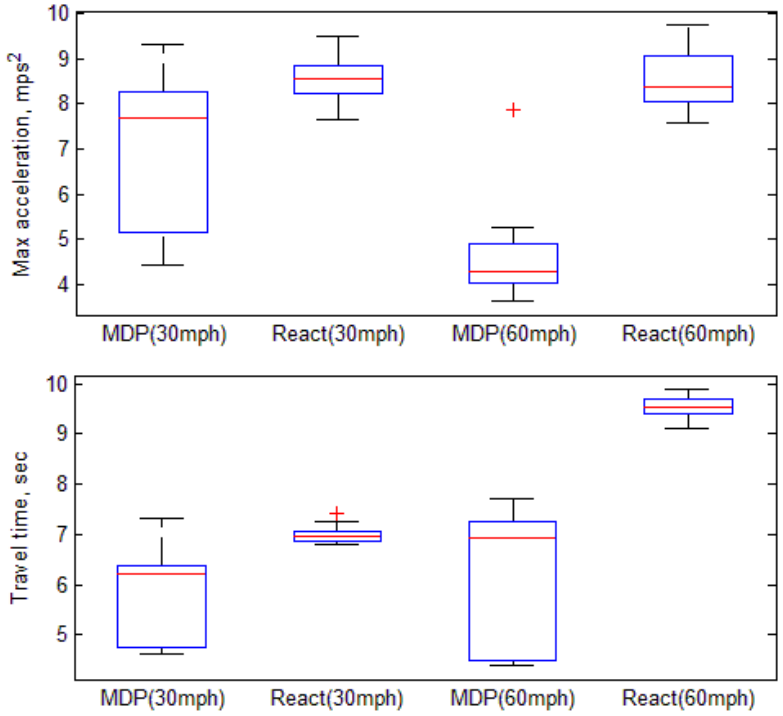

Figure 4.14: Max acceleration used and travel time comparison for MDP and reactive methods. The higher variances of MDP results are due to variety of solutions.

The statistical data over all 100 trials shown in Fig. 4.14 gave the significantly lower maximum acceleration used to avoid collision, and improvement in travel time. The wider range of travel time and accelerations were resulted by originality of each solution found by MDP for each particular allocation of the cars.

4.7.2 Quantitative Results of the Highway Scenario

The specific setup for the parallel driving considered two cars moving on a highway in different lanes as shown in Fig. 4.15. The human-driven car shown by the blue rectangle was given an initial speed twice lower that the autonomous vehicle shown by the red rectangle. The simulation considered two speed modes: high speed (60 mph) and low speed (30 mph). For the purpose of statistical analysis, each setup was run 100 iterations while the initial speed was various as $\pm$ 10% of the speed mode. Other initial conditions such as locations of the vehicles were set the same over all iterations. To be able to compare results, the behavior of the human driver was reproduced by a script and made the blue car merging left. This simulated intention created an obstacle for the autonomous vehicle.

The results over 100 iterations shown in Fig. 4.16 and 4.18 compared the sequential MDP-based, single-

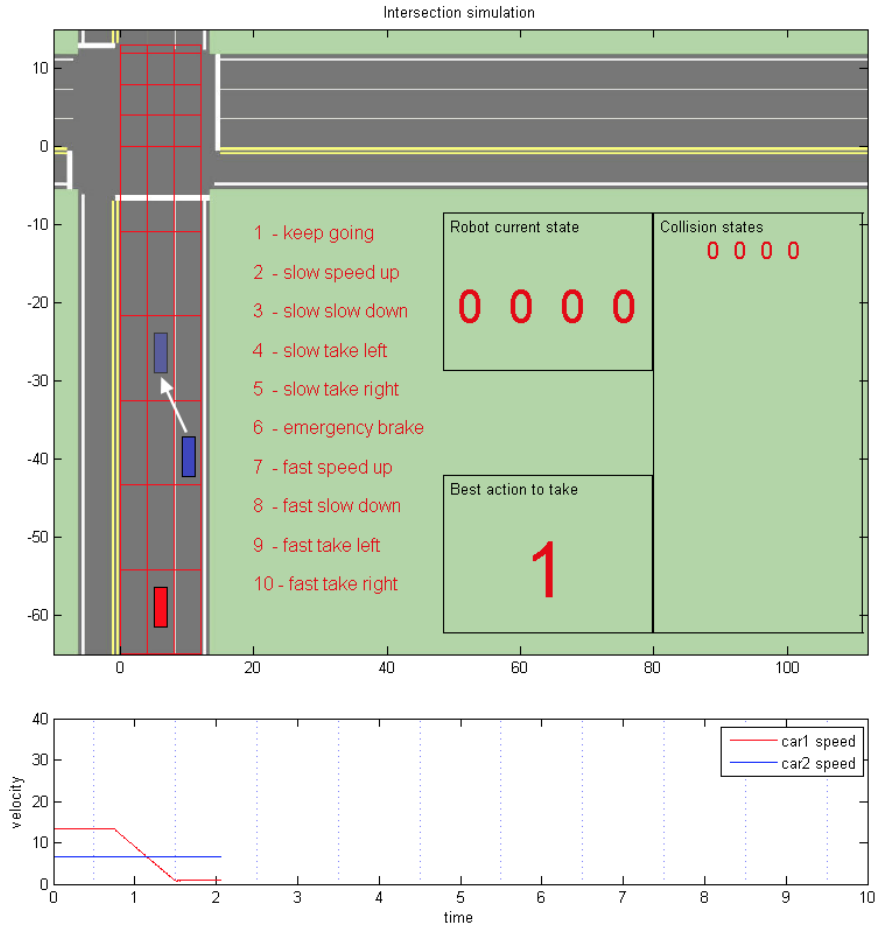

Figure 4.15: Parallel driving setup for statistical analysis. A human driven car (blue rectangle) created an obstacle by merging left to the lane used by the autonomous car (red rectangle) which has twice higher velocity.

step optimization and simple sensor-based reactive algorithms controlling the autonomous car. It should be noted that none of these algorithms got into collision in this specific setup. The sequential algorithm showed better travel time than others due to ability to increase the velocity of the vehicle after overtaking maneuver used, while the single-step algorithm performed only changing lane to the left lane to avoid collision and then keep running. In some examples the single-step CAS adjusted the speed of the vehicle before changing lane to get more time for a maneuver what led to a larger mean and variance of the travel time.

Maximum acceleration used to avoid a collision was defined as a parameter of comfort of the ride and registered the highest acceleration or deceleration (braking) taken by the autonomous car. As shown in Fig. 4.18 both sequential and single-step algorithms showed good results comparing to the reactive one which was developed as ”the last line of defense” and applied the maximum brakes in the very last moment before crash.

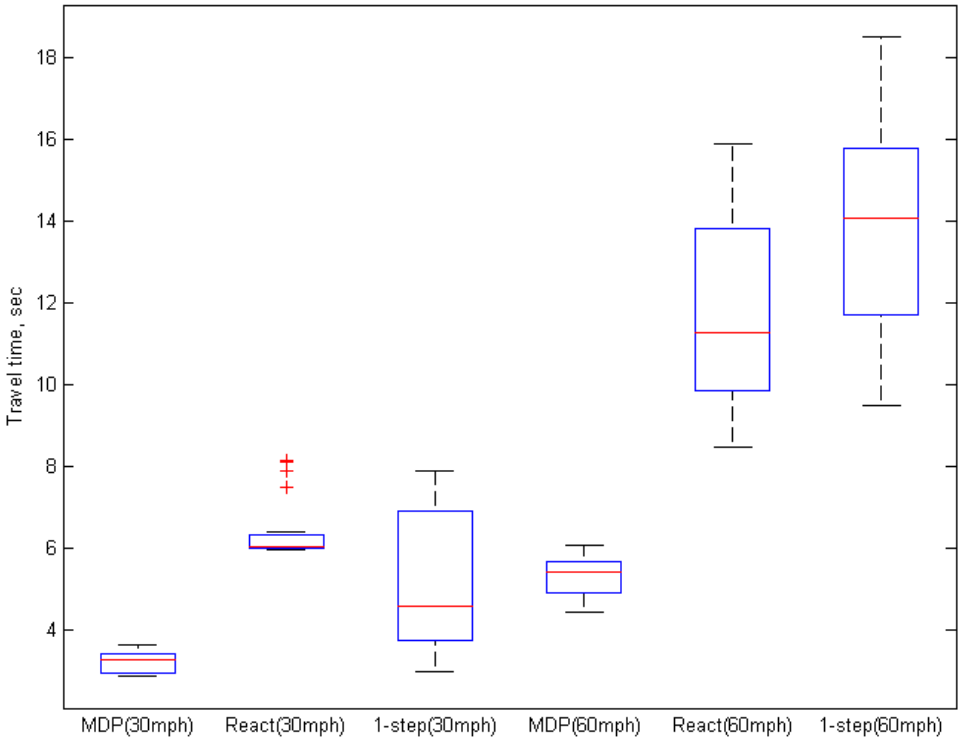

Figure 4.16: Statistics comparison for travel time in parallel driving scenario over 100 iteration.

It would be improperly to compare the reactive algorithm with others, but this comparison showed how the comfort of the passengers of the autonomous vehicle may be improved if the proactive approach is used in addition to a the reactive one. In addition, the single-step algorithm showed lower deceleration comparing to MDP-based in all speed modes due to an ability to change the action more often.

Since we developed the car control algorithm which had to operate the autonomous vehicle in real-time, another very important parameter of the CAS system was the time required for decision making. All three systems - reactive, single-step CAS and sequential CAS were evaluated when the number of neighbor human-driven cars taken into account was various from 1 to 3. Results shown in Table 4.2 revealed that the reactive system does not have any delay except the minimum step of simulation itself and it performed an emergency brake as soon as the distance to the car in front of the autonomous car was less than some threshold. The single-step optimization showed the small raise in the computation time caused by the need to make s many predictions as we took cars in the consideration. The computation time of the sequential MDP-based algorithm was larger than 10 seconds for all number of considered cars and may cause the significant difficulties in the implementation to the real vehicle. The reason for that was the iteration process which

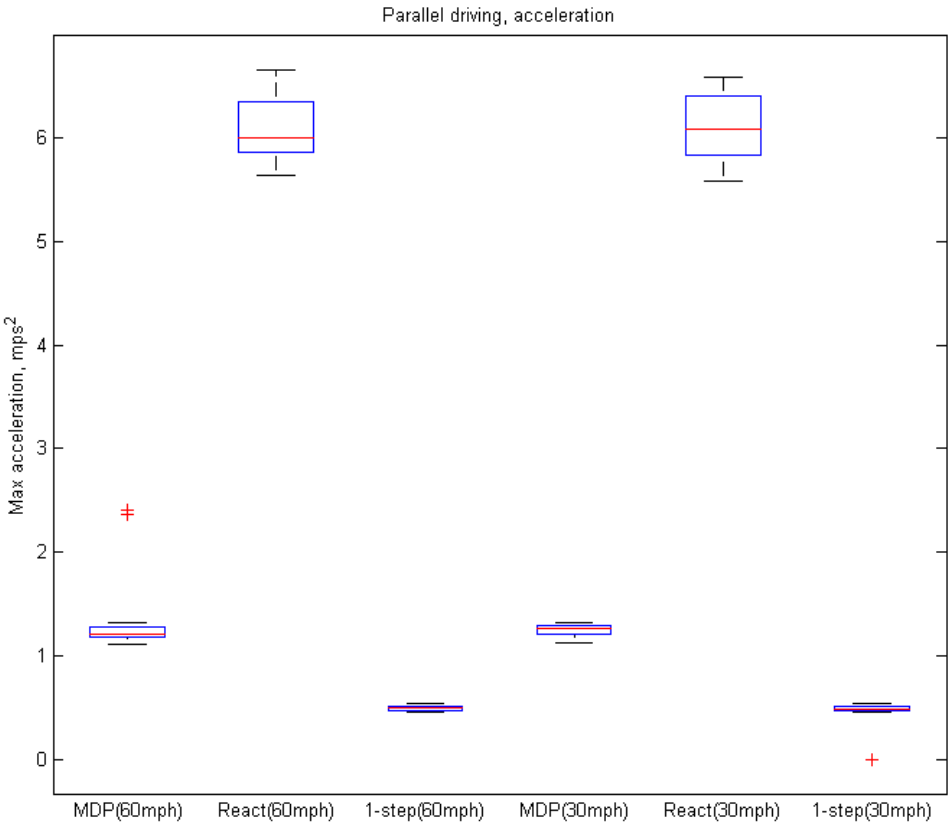

Figure 4.17: Statistics comparison for acceleration time in parallel driving scenario over 100 iteration.

operated with very large data structures. The main computation intense was caused by the multiplication of transition and reward matrices with 10^7 elements each as we mentioned in Section 2.7.1.

Table 4.2: Computation time required for decision making in seconds

N_{cars}	Reactive	Single-Step	Sequential
1	0.05	0.2	13
2	0.05	0.35	13
3	0.05	0.45	14

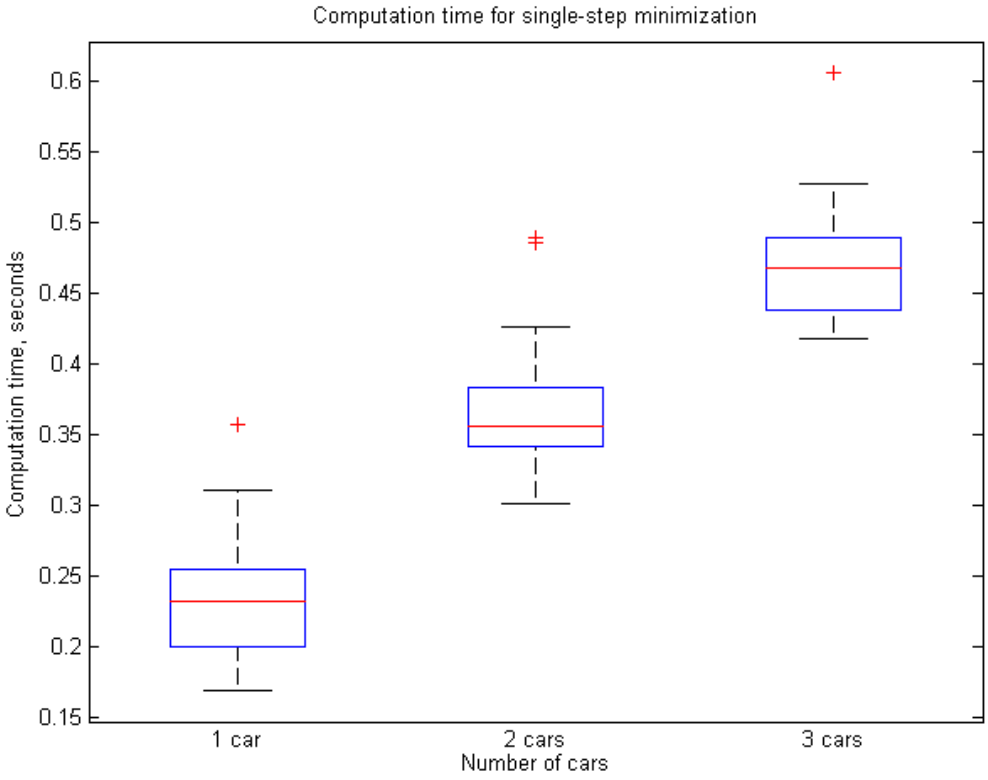

Figure 4.18: Statistics of computation time required by the single-step CAS algorithm to build predicted trajectories and solve optimization task with respect to the number of neighbor human-driven cars taken into account.

CHAPTER 5

Conclusion

5.1 Summary

This work proved the importance of taking an intention of other drivers into account and possibility of the modeling the human behavior with Gaussian models using two-step Gaussian Process. These models represented the prior knowledge of the human behavior and estimated the most probable trajectories based on this knowledge by giving the mean and variance of the predicted location for finite time horizon. Various simulations proved the sufficiency of an early change in the velocity or trajectory of the agent in order to avoid a collision between cars. This little change successfully reduced the probability of a collision in both scenarios considered driving on a highway and at an intersection. Proposed proactive algorithms showed significant improvement in maximum acceleration used and travel time with respect to the sensor based reactive approach. However, the single-step CAS showed the better results than sequential MDP-based CAS due to the higher frequency of updating the solution and lower computation time the same time. On other hand, the proof of this single-step decision making algorithm was limited to a simple driving task required a primitive action to success. Such condition made the sequential decision making excessive and ineffective. It can be assumed that in some specific allocation of the human-driven vehicles, the driving task may take a form of a puzzle where taking a primitive action won't be enough to success. In that case, the MDP algorithm would show the better results what should be evaluated in future work.

Significant advantages over the reactive methods using a full-stop algorithm programed with "if-else" were reached in the travel time and acceleration. Simulations showed that the delay was reduced by 25 - 50% for the case of the cross-traffic. The car has performed a full stop only when there was not enough distance to maintain the lower speed while other cars were passing through.

In sequential decision making using MDP, the calculation of the optimal policy carried out on-line significantly delayed the CAS algorithm and cannot be implemented as an on-line process on a real car. The only

way to reduce this delay is to increase the scale of problem and make the sequential algorithm be working only for defining the general strategy on the road, while the single-step algorithm does a routine collision avoidance task.

5.2 Future Work

This work posed new problems for further research. Recommendations for future work could include:

- Consider a cooperation of the MDP and the single step algorithms. This option could get benefits of both algorithms: run the single-step algorithm with a short time horizon to avoid collision with the nearest cars, while the sequential CAS may be used for long-term estimation with a large time step to define the lanes and modes of driving which safer and more comfortable.
- Consider more intentions and even more important consider the mix of intentions to evaluate a risk of misclassification of the behavior.
- Reduce number of limitations caused by the limitation of models used and algorithms.

References

[1] MathWorks, "Modeling a vehicle dynamics system."

[2] U. DOT, "Table 1-11: Number of us aircraft, vehicles, vessels, and other conveyances. may," 2013.

[3] E. Z. L.J. Blincoe, T.R. Miller and B. Lawrence, "The economic and societal impact of motor vehicle crashes, 2010."

[4] N. T. S. Board, "Special investigation report. the use of forward collision avoidance systems to prevent and mitigate rear-end crashes," May 2015. [Online; posted 19-May-2015].

[5] G. Leen and D. Heffernan, "Expanding automotive electronic systems," *Computer*, vol. 35, no. 1, pp. 88–93, 2002.

[6] J. Levinson *et al.*, "Towards fully autonomous driving: Systems and algorithms," in *Intelligent Vehicles Symposium (IV), 2011 IEEE*, pp. 163–168, IEEE, 2011.

[7] A. Davies, "Self-driving cars will make us want fewer cars," March 2015. [Online; posted 09-March-2015].

[8] R. H. Paul Gao and A. Zielke, "A road map to the future for the auto industry," October 2014. [Online; posted 01-October-2014].

[9] H. King, "Google: Human drivers are the problem," May 2015. [Online; posted 12-May-2015].

[10] U. C. Bureau, "Statistical abstract of the united states: 2012," *U.S. Census Bureau*, 2011.

[11] D. L. Strayer and W. A. Johnston, "Driven to distraction: Dual-task studies of simulated driving and conversing on a cellular telephone," *Psychological science*, vol. 12, no. 6, pp. 462–466, 2001.

[12] S. Sivaraman and M. M. Trivedi, "Looking at vehicles on the road: A survey of vision-based vehicle detection, tracking, and behavior analysis," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 14, no. 4, pp. 1773–1795, 2013.

[13] W. Liu, X. Wen, B. Duan, H. Yuan, and N. Wang, "Rear vehicle detection and tracking for lane change assist," in *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 252–257, IEEE, 2007.

[14] T. Li, S.-J. Chang, and Y.-X. Chen, "Implementation of human-like driving skills by autonomous fuzzy behavior control on an fpga-based car-like mobile robot," *Industrial Electronics, IEEE Transactions on*, vol. 50, no. 5, pp. 867–880, 2003.

[15] R. Sukthankar, "Raccoon: A real-time autonomous car chaser operating optimally at night," tech. rep., DTIC Document, 1992.

[16] D. A. Pomerleau, "Alvinn: An autonomous land vehicle in a neural network," tech. rep., DTIC Document, 1989.

[17] G. W. Goodrich, "Collision avoidance using optical pattern growth rate," march 1981. US Patent 4,257,703.

[18] J. K. Aggarwal and M. S. Ryoo, "Human activity analysis: A review," *ACM Computing Surveys (CSUR)*, vol. 43, no. 3, p. 16, 2011.

[19] W. Sheng, Y. Ou, D. Tran, E. Tadesse, M. Liu, and G. Yan, "An integrated manual and autonomous driving framework based on driver drowsiness detection," in *Intelligent Robots and Systems (IROS), 2013 IEEE/RSJ International Conference on*, (Tokyo, Japan), pp. 4376 – 4381, Nov. 3–8, 2013.

[20] J. Hardy, F. Havlak, and M. Campbell, "Multiple-step prediction using a two stage gaussian process model," in *American Control Conference (ACC), 2014*, pp. 3443–3449, IEEE, 2014.

[21] M. Garcia Ortiz, *Prediction of driver behavior*. PhD thesis, Bielefeld University, march 2014.

[22] S. M. LaValle, "Rapidly-exploring random trees a new tool for path planning," 1998.

[23] T. Brandt, T. Sattel, and M. Bohm, "Combining haptic human-machine interaction with predictive path planning for lane-keeping and collision avoidance systems," in *Intelligent Vehicles Symposium, 2007 IEEE*, pp. 582–587, IEEE, 2007.

[24] T. Bandyopadhyay *et al.*, "Intention-aware pedestrian avoidance," in *Experimental Robotics*, pp. 963–977, Springer, 2013.

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

[26] J. Santa, A. F. Gomez-Skarmeta, and M. Sanchez-Artigas, "Architecture and evaluation of a unified v2v and v2i communication system based on cellular networks," *Computer Communications*, vol. 31, no. 12, pp. 2850–2861, 2008.

[27] F. F. M Menouar, Massimiliano Lenardi, "A movement prediction based routing protocol for vehicle-to-vehicle communications," *Communications*, vol. 21, pp. 07–2005, 2005.

[28] G. Chowdhary, H. Kingravi, J. P. How, P. Vela, *et al.*, "Bayesian nonparametric adaptive control of time-varying systems using gaussian processes," in *American Control Conference (ACC), 2013*, pp. 2655–2661, IEEE, 2013.

[29] L. Csató and M. Opper, "Sparse on-line gaussian processes," *Neural computation*, vol. 14, no. 3, pp. 641–668, 2002.

[30] R. Bellman, "A markovian decision process," tech. rep., DTIC Document, 1957.

[31] A. Geramifard *et al.*, "A tutorial on linear function approximators for dynamic programming and reinforcement learning," 2013.

[32] Carnetsoft, "Driving simulator for training and research," May 2015.

[33] C. L. E. Hairer and M. Roche, *The numerical solution of differential-algebraic systems by Runge-Kutta methods.* Springer, 1989.

[34] W. Van Winsum, D. de Waard, and K. A. Brookhuis, "Lane change manoeuvres and safety margins," *Transportation Research Part F: Traffic Psychology and Behaviour*, vol. 2, no. 3, pp. 139–149, 1999.