Probabilistic Motion and Intention Prediction for Autonomous Vehicles

Probabilistische Bewegungs- und Intentionsvorausssge fÃijr autonome Fahrzeuge

Master-Thesis von Lina Jukonyte aus Utena, Litauen

Mai 2019

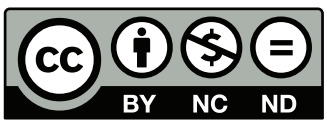
TECHNISCHE UNIVERSITÄT DARMSTADT

Probabilistic Motion and Intention Prediction for Autonomous Vehicles
Probabilistische Bewegungs- und Intentionsvorausssge fÃijr autonome Fahrzeuge

Vorgelegte Master-Thesis von Lina Jukonyte aus Utena, Litauen

1. Gutachten: Prof. Jan Peters, Ph.D., Prof. Matthias Hollick, Ph.D.
2. Gutachten: Dorothea Koert, M.Sc., Joni Pajarinen, D.Sc. (Tech.)
3. Gutachten: Dominik PÃijllen, M.Sc.

Tag der Einreichung:



Erklärung zur Master-Thesis

Hiermit versichere ich, die vorliegende Master-Thesis ohne Hilfe Dritter und nur mit den angegebenen Quellen und Hilfsmitteln angefertigt zu haben. Alle Stellen, die aus Quellen entnommen wurden, sind als solche kenntlich gemacht. Diese Arbeit hat in gleicher oder ähnlicher Form noch keiner Prüfungsbehörde vorgelegen.
In der abgegebenen Thesis stimmen die schriftliche und elektronische Fassung überein.

Darmstadt, den 5. Mai 2019

(Lina Jukonyte)

Thesis Statement

I herewith formally declare that I have written the submitted thesis independently. I did not use any outside support except for the quoted literature and other sources mentioned in the paper. I clearly marked and separately listed all of the literature and all of the other sources which I employed when producing this academic work, either literally or in content. This thesis has not been handed in or published before in the same or similar form.
In the submitted thesis the written copies and the electronic version are identical in content.

Darmstadt, May 5, 2019

(Lina Jukonyte)

Abstract

Decision-making task is one of the most determinant bonds for constructing an autonomous system. Making solid decisions by foreseeing and estimating future consequences on its own, it what makes autonomous systems intelligent. Decision making on its own is already complex task, but for vehicles, it makes more complex because of the uncertainty of the real world and continues vehicles' interaction with other vehicles and obstacles. Sensors which are using for real-world understanding and features as speed, position, other objects of traffic, etc. are noisy and very dependables from external conditions. But again, it is very hard to measure others road users' intentions due to its randomness, additionally, completely or partially visible obstacles of the road can make any received measurements and information useless. The unit responsible for decision making has to be sensible for these issues and be able to foresee the future conditions that could develop in an endless number of ways to achieve the final goal with the maximum reward or, in other words, with a minimum cost of the process.

TODO: add part about what was done in the thesis (at the very end).
Removing a driver from behind the wheel takes away more than just the physical responses. It also eliminates the complex decision-making that goes into even routine journeys – choosing whether to swerve into a neighboring lane to avoid a possible obstacle or navigating ambiguous intersections.

Zusammenfassung

Hier können Sie Ihre deutsche Zusammenfassung schreiben.

Acknowledgments

Contents

Figures and Tables

List of Figures

List of Tables

Abbreviations, Symbols and Operators

List of Abbreviations

Notation	Description
ADAS	Automotive Driver Assistance Systems
aDDa	Autonomous Driving Darmstadt for Students
BN	Bayesian Networks
CAN	Controller Area Network
DBN	Dynamic Bayesian Networks
DTW	Dynamic Time Warping
ECU	Electronic Control Units
FL	Fuzzy Logic
GMR	Gaussian Mixture Regression
GP	Gaussian Processes
GPS	Global Positioning System
HMM	Hidden Markov Model
IDS	Intrusion Detection System
KF	Kalman Filter
LCSS	Longest Common Subsequence
LiDAR	Light Detection and Ranging
LSTM	Long Short Term Memory
MAC	Message Authentication Code

OBD-II	On-Board Diagnostics-II
POMDP	Partially observable Markov decision process
RaDAR	Radio Detection and Ranging
ROS	Robot Operating System
RRT	Rapidly-exploring Random Tree
RViz	ROS visualization
SKF	Switching Kalman Filter
SVM	Support Vector Machine
TuD	University of Darmstadt
UAV	Unmanned Aerial Vehicle
V2I	Vehicle to Infrastructure
V2IoT	Vehicle to Internet of Things
V2V	Vehicle to Vehicle
VANET	Vehicular Ad Hoc Network
VGMM	Variational Gaussian Mixture Model
WHO	World Health Organization

List of Symbols

Notation	Description
$\boldsymbol{\theta}$	vector of parameters from a probability distribution

List of Operators

Notation	Description	Operator
ln	the natural logarithm	$\ln(\bullet)$

1 Introduction

People nowadays can hardly imagine their life without driving. And it is natural since traveling brings independence and freedom to human life. For example, an average American person makes 2.2 driving trips per day and spends 50.6 minutes on the road, which makes over 300 hours every year people spend in their cars [5]. Despite the joy driving brings to people, traffic safety is a major aspect which cannot be ignored. With increasing time which people spend on the cars, a number of a vehicle-related accident is far away from being perfect. The World Health Organization (WHO) annually announce a report which includes a total number of people lives which were taken away due to car accidents. The latest report was published in December of 2018 and stated that during this year there was more than 1.35 million death worldwide [6].

Recent years were full of massive developments towards autonomous driving in autonomous industry. Achievements in one area can be helpful in developing other areas, i.e. great success in image recognition and perception can allow computers to achieve super-human performance [7], etc. Unfortunately, image recognition and environmental perception alone are not enough to solve all the problems of autonomous cars. In ideal circumstances achieving full autonomy of the cars would help not only to save the environment, as well it would benefit traffic participants with more smoothly traffic and more safety on the roads. Industrial innovation experts from ARK Invest strongly believe that with fully autonomous cars accidents on the road would drop to 80% [8].

Although autonomous cars are something that engaging a wide range of engineers for some time already, this area not fully developed yet and will continue engage engineers even more in the future. At the moment big achievement which is equipped into the majority of new cars is Automotive Driver Assistance Systems (ADAS), which for the fact does not enable yet full autonomy of the cars, but it successfully assists driver while driving a car.

It is not a secret that all drivers need to interact with each other non-stop in one way or another while they are driving. This communication together with the individual behaviour of a driver is the main key to traffic safety. The behaviour of the driver can be considered as a combination of current traffic observations, short future forecast, decision making and completing actions. The driver should make decisions and actions with full safety concept for himself and other traffic participants. The same is with fully autonomous vehicles: algorithm which is running while a car is driving should ensure all passengers in the car and other traffic participants safety while making decisions and maneuvering through traffic. It is not hard to understand that one of the biggest problem with both the ADAS systems and the full autonomous cars is the human factor. To improve ADAS systems and achieve safety in fully autonomous cars, prediction methods are essential. Requirements for prediction are precision preciseness (with some time in advance), efficiency and reliability. This thesis will focus on the movement and intention prediction of humans in other cars.

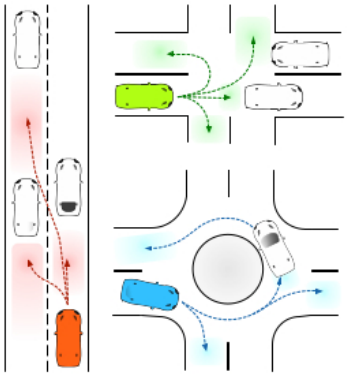


Figure 1.1.: Insertion Areas (Colored Regions) Under Different Driving Scenarios [1]

1.1 Background

Beyond trendy names like Tesla, Google, Aptiv, etc., chasing driverless cars, beyond a large number of automobile brands and other tech engineers, University of Darmstadt (TuD) has its own Autonomous Driving Darmstadt for Students (aDDa) working group [9]. aDDa initiative started on October 2017. A group of students and their supervisors from eight different departments at TuD are working for one purpose to develop fully autonomous car by themselves, here, at University. All participants of the working group closely cooperate bringing together interdisciplinary know-how experience to jointly set up and operate an autonomous vehicle. One special feature of aDDa is that the main work is done in the context of student projects (final thesis, semester work, permanent work at the team, etc.). By working together on the complex tasks of autonomous driving, participants are solving problems for tomorrow.

In not so long period of glsaDDa existing, it is made a lot of developments and improvement of existing systems. Some works to mention: "Development and Implementation of a Long-Term Dynamics Control for Automated Driving", "Collision Avoidance in Uncertain Environments for Autonomous Vehicles using POMDPs", "Conception and Design of a Camera Mounting and Calibration for Test Vehicle", "Development of an IT Security Concept for an Automated Vehicle, Pedestrian Detection", "Tracking and Intention Prediction in the Context of Autonomous Driving" and much more [9]. This thesis is also a part of aDDa project.

1.2 Purpose

Humans are very irrational and unpredictable and because of that, it is very hard to model them. Moreover, there are no two exact same people, what makes the task to model human behavior almost impossible, since every possible scenario as an endless possible outcome. When an individual is driving it is nearly always necessary to take into account surrounding cars and other traffic participants due to ensure safe, fast and energy optimized journey.

Due to the irrationality of humans and recent success and a still big interest in autonomous cars, the purpose of this thesis is to provide an initial step in a probabilistic collision prediction and decision-making system which aims at producing a risk field for the vehicle that predicts upcoming risks. This step will include creating an algorithm which will use a probabilistic approach and tries to predict future movement and intention of surrounding cars in urban areas.change pictures and maybe explain more about thesis approach in general.

The overall research question is defined as:

- How can probabilistic movement prediction using future estimations be applied for an autonomous vehicle?

This research question is then subdivided into smaller questions and task to achieve during in this research work. Three of these tasks, which are the focus of this thesis, are defined as

- Find/create a probabilistic model that learns from various demonstrations. And use this model for trajectory and intention prediction of the car in front of ego vehicle.
- Investigate if prior information about environment can improve quality of predictions.
- Investigate, how feasible is a probabilistic future movement estimation system, in terms of accuracy and computational time, for real-time applications?

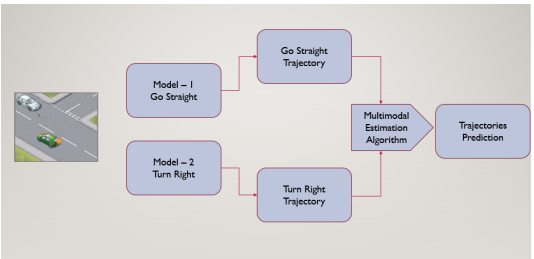


Figure 1.2.: Scheme of Simple Model Based Approach [2]

The most important thing in driving independent of the car is driverless or need a driver is safety. Safety is not possible without security, and considering this, this thesis will provide an overview of the main security and privacy issues on autonomous driving considering movement prediction. Which leads us to the forth and the final research question for this thesis:

- FINALLY DECIDE

1.2.1 Scope of the Thesis

Autonomous systems are very complex by its' nature and it is natural to make substantial limitations to obtain a reasonable scope for a master's thesis. This thesis has been decided to use Robot Operating System (ROS) environment system, RViz as its' simulation visualization tool, programming part is done in Python and C++ programming languages. Additionally, the evaluated scenarios have been chosen to be T-intersections and crossroads (four-way intersection) with a known environment, limited to only include a single vehicle in addition to the ego vehicle. This choice has been made due to the system complexity, concerning interactions, that multiple cars would introduce. The vehicles in these scenarios are considered to be cars.

Furthermore, some research areas which include image processing, object tracking, mapping and trajectory planning will not be addressed since these areas constitute research areas single-handedly. Incorporating any of these areas would make this thesis even more complex and remove attention from what should be the main focus of the thesis: the probabilistic future movement estimation.

1.3 Thesis Outline

This study focuses on developing and evaluating probabilistic based movement and intention prediction algorithm. This the algorithm uses external cues to predict surrounding vehicles movement in urban situation.

The thesis is organized as follows:

- Chapter 2. **Fundamentals and Related works** focuses foundation of the work and presents a theory behind the scope of the thesis.
- Chapter 3. **Approach** describes approaches of the thesis.
- Chapter 4. **Simulation Setup** defines how and why simulation was set in the way in was. Defines inputs for the system and experiments.
- Chapter 5. **Experiments and Results** describes experiments done during the thesis writing period and evaluate results which were received by performing various experiments.
- Chapter 6. **Security Aspects** is based on the fact that "there is no safety without security" and tries to explain the main security and privacy issues of autonomous cars related to movement predictions.
- Chapter 7. **Conclusion and Future Works** wind up this thesis with conclusions and future works based on the findings of previous chapters.

2 Fundamentals and Related Work

2.1 State of the Art

The most studies related to movement prediction on vehicles focus on lane change predictions. And there are various different methods are proposed to solve this task, the most popular are: Dynamic Bayesian Networks (DBN), Bayesian Networks (BN), Support Vector Machine (SVM), Hidden Markov Model (HMM), Mind-tracing and Fuzzy Logic (FL).

BN could be considered as a graphical representation of probability distribution. In [10, 11] DBN and BN are used to recognize actions which driver is intend to perform. Lateral movement of a vehicle is expressed using BN, having several various nodes for probability. The probability distribution for every node is determined by doing an analysis of driver behaviour in the past while driving. And ultimately, the final prediction is obtained by calculating the probability of a certain movement with respect to the possibility for every node.

In [12, 13] process of driving is described as a set of various different states while driving. When some particular actions appear in particular defined sequence, it is possible to calculate the probability that the state will change to a particular state. In these works likelihood of states shifting is designed using HMM. Authors of [14, 15] expanded their past work using even more realistic test case - they equipped a vehicle with sensors and used received graphical data to get more accurate results. Graphical models together with HMMs and its extensions were trained using the data from experimental driving, seven different driver models were created: passing, changing lanes (to the right or to the left), turning right or left, starting and stopping. The result authors received and presented was "on average, the predictive power of our models is of 1 second before the maneuver starts taking place" [15].

Author of [16] proposed new method for prediction making, which was named Mind-tracing. It is a computational framework which is able to predict possible drivers' intentions. This method is different from others because here different cognitive model versions which include a flood of a possible intention and action is used. Each action and possible intention are compared with a driver's behaviour at the same time. And the closet to the human behaviours is used for the further intentions expression.

[17] introduces FL as an alternative method for modelling behaviour of the driver. The research paper is mainly focused on the process of decision making on lanes of a highway. For getting results a triangular membership function was used, fuzzy rules were defined by observing training procedure and learning from obtained results. The model was developed using actual traffic data. The used model combines the speed and speed difference of the vehicle, the lead and lag gap distances and the remaining distance to the end of the merge lane as input variables. The precision of prediction using the model was higher than using the binary Logit model. The high prediction accuracy received using this model results in prediction accuracy made using this model overall.

[18] tested the validity and accuracy of SVM in movement prediction. After choosing proper hints for movement changing, data recorded by doing test were divided into different groups which were used for training classifier. Predictions were made using current information of the vehicle and classification hints at the current time.

Even though all methods have the same purpose, it is very hard to compare them directly. Results received having measurements in different situations and in different time steps, e.g. one research paper gives prediction two seconds in advance before a lateral position of vehicle's overlaps with the lane border, while other paper gives prediction only one second before crossing the lane. Furthermore, there is no exact definition of *lane crossing* moment, usually, it is the moment when vehicle cross edge of a lane, but in some paper, it is not clear enough.

Since it is not possible to make a clear comparison between methods due to essential differences in testing environments and different data sets, the Table 2.1 lists the best timing accuracy for each method.

As it is possible from the results shown in the table the best methods for predicting movement changes is received by using BN and SVM. These two methods are able to predict quite accurate and with a relatively short period of time, what

Table 2.1.: Different Methods Performance Comparison

Method	References	Accuracy	Time
(Dynamic) Bayesian Network	[10]	80%	1.5s before changing movement
	[11]	89%	0.5s after changing movement
Support Vector Machine	[18]	87%	0.3s after changing movement
Hidden Markov Model	[12]	89.4%	2s after changing movement
	[13]	95.2%	2s after changing movement
	[14, 15]	Unknown	0.4s before any sign of changing movement appears
Mind-tracing	[16]	82%	1.1s before changing movement
Fuzzy Logic	[17]	86.8%	Unknown

will lead in having more time in advance to decide which action to make.

For further work, any Bayesian filter/classifier could be an acceptable method for examining behaviour of the driver for various reasons:

- Bayesian-based methods can perform well while working with a very big amount of data;
- It gives results with high accuracy from a problem, containing many features. It is needful to include different physical data while modeling and examining drivers' behaviour. Traditional statistical classifiers most likely to be insufficient while processing high dimensional data.
- Bayesian filter/classifier are robust to over-fitting problem and rely on margin maximization instead of finding an edge for prediction directly from the training data.

2.2 Probabilistic Estimation Methods

Reasonable prediction and following decision-making process require considering uncertainty and objectives for the current situation. In this section, uncertainty will be represented as a probability distribution.

Uncertainty can be a result of partial information about the state of the world. In a real world trying to fulfil any given task, it is possible to meet various reasons which do not allow to finish a task without any difficulties. What means, that with information we have at hand, it is hardly possible to make a task evaluation with being completely certain.

Uncertainty can appear from practical and theoretical limitations while trying to predict future events, e.g., trying to exactly predict how a human would react in one situation or another, a decision support system would need to consider a model of the human brain. Even if the operation is known very well, it is still difficult to predict the end state and next actions which will be taken, due to spontaneous failures or other agent actions.

A robust prediction (and later decision) making system need to take into account sources of uncertainty, which exist in the current state and consider it when computing the future outcomes for events. In order to describe uncertainty computationally, it needs to have a formal representation.

2.2.1 Belief State and Probability

Solving tasks which involve uncertainty, it is very important to be able to compare the credibility of different statements. For example, if belief for action E is stronger than our belief for action T, then $E \succ T$. If E and T have the same degree of belief, then $E \sim T$.

It is also beneficial to be able to compare beliefs about statements considering some given information, e.g., we can say that likelihood for action C may happen while E condition is happening is bigger than having T, then this expression would be written $(E \mid C) \succ (T \mid C)$.

In order to make particular assumptions about the relationships of the operators $\succ$, $\prec$ and $\sim$. The assumption of *universal comparability* and *transitivity* assumptions requires to hold the same mathematical rules. Both assumptions allow representing degrees of belief by a real-valued function [19], i.e. probability function P can be expressed like that:

$$P(A|C) > P(B|C) \iff (A|C) \succ (B|C)$$

$$P(A|C) = P(B|C) \iff (A|C) \sim (B|C).$$

If new assumptions about the probability P form, then P need to satisfy the main axioms of probability: $0 \leqslant$ P (A | B) $\leqslant$ 1. If we are sure that A action will happen when B action is given then P (A | B) = 1. If A action will not happen when B action is given, then P (A | B) = 0.

Deep review about probability theory won't be provided in here, but this work relies on important probabilities properties. The first of them is a definition of *contidion probability*:

$$P(A|B) = \frac{P(A,B)}{P(B)}, \tag{2.1}$$

where P(A, B) shows the probability of A and B both being true.

Another property which is important is the *law of total probability*, which states that if β is a set of "mutually exclusive and exhaustive propositions" [19], then

$$P(A|C) = \sum_{B \epsilon \beta} P(A|B,C)P(B|C) \tag{2.2}$$

Finally, the most important rule for further work comes from the definition of *conditional probability*:

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}. \tag{2.3}$$

This equation is known as ***Bayes' rule***, and as mentioned earlier, it will be very important for the following work.

But still, after this short introduction, question what exactly belied state is, still exists. One option to answer this would be the most believable next state for an examined object, considering experience in the past, which is given. This idea can be sound and save the basis for predictions in some cases, but in general, this idea is not sufficient. Being able to operate efficiently the degree of uncertainty must be taken into account, e.g. if the main agent is confused what future state could be, it could be proper to ask directions, take a look into the map, search for reference point, etc.

Other options for belief computation would be using probability distributions over states of the world, which we have. In this case, distributions encode the subjective probability for the main agent and include information about the state of the world and give a basis for taking action under uncertainty we have. Moreover, sufficient statistical information of action made in the past and initial belief state of the agent is comprised, i.e. computed belief state for the current agent's state and additional information about its past observations and/or action made, would provide any further information about the current state of the world [20].

Computing belief states[20]:

A belief b is a probability distribution over state space $\mathcal{S}$, *b(s)* is the probability set to world state s by belief state b. The axioms for belief state is the same as for probabilities: $0 \leqslant$ b(s) $\leqslant 1$, for all s ϵ $\mathcal{S}$ and $\sum_{s \epsilon \mathcal{S}} b(s) = 1$. At every new step, new belief *b'* must be computed given old belief b, an action a and an observation o. The new belief of an new state *b'(s')* can be calculated using formula:

$$\begin{aligned} b'(s') =& Pr(s'|o,a,b) \\ &= \frac{Pr(o|s',a,b)Pr(s'|a,b)}{Pr(o|a,b)} \\ &= \frac{Pr(o|s',a)\sum_{s \epsilon \mathcal{S}} Pr(s'|a,b,s)Pr(s|a,b)}{Pr(o|a,b)} \\ &= \frac{O(s',a,o)\sum_{s \epsilon \mathcal{S}} T(s,a,s')b(s)}{Pr(o|a,b)} \end{aligned} \tag{2.4}$$

The denominator of equation (2.4), Pr(o | a, b), can be interpreted as a normalizing factor, which is independent of next state s', which causes the sum of belief of all possible next states to 1. The state estimator function SE(b, a, o), which task is to update the belief state based on the *a, o* and the previous *b*, as its output gives new belief for new state *b'*.

Please note, that this subsection and the computation of belief states description is taken directly from Partially observable Markov decision process (POMDP) steps description. In later work, belief update will act an important role, but it will be computed using different components. Detail description of belief computation related to this work will be provided in next chapters.

To have particular classifier is not enough for making accurate trajectory and movement predictions. Next chapter introduce with the most popular model for movement predictions.

2.3 Movement Prediction

Foresee future moments and trajectories for dynamical objects in traffic scenarios is vital in order to obviate risks which occur on the roads. Prediction despite of short or long term they are, must have sufficient time in advance to avoid traffic situations we, as traffic participants, don't want. In this section, relevant researches for trajectory and movement predictions are introduced.

There is numerous research made on a trajectory and movement predictions with a vehicle as interest on traffic scenarios. [3] suggesting a one way of classifying methods for motion prediction. The main three categories with an increased rate of flexibility were defined: ***physical-based, maneuver-based*** and ***interaction aware***.

- **Physics-based** motion models are the most simple of all categories. It is considered that the movement of vehicles depends only on the laws of physics. A wider description is in subsection 2.3.1.
- **Maneuver-based** motion models are more advanced than physics-based because maneuver-based motion models also consider future movements of a car which also depends on the maneuver which is intended to perform by a driver. A wider description is in subsection 2.3.2.
- **Interaction-aware** motion models take into account consideration connections between maneuvers of the car, as well as rules of the traffic. This method as not so popular as previous ones due its complicity to adapt to the real life scenarios. A wider description is in subsection 2.3.3.

Figure 2.1 summarizes motion models defined in [3].

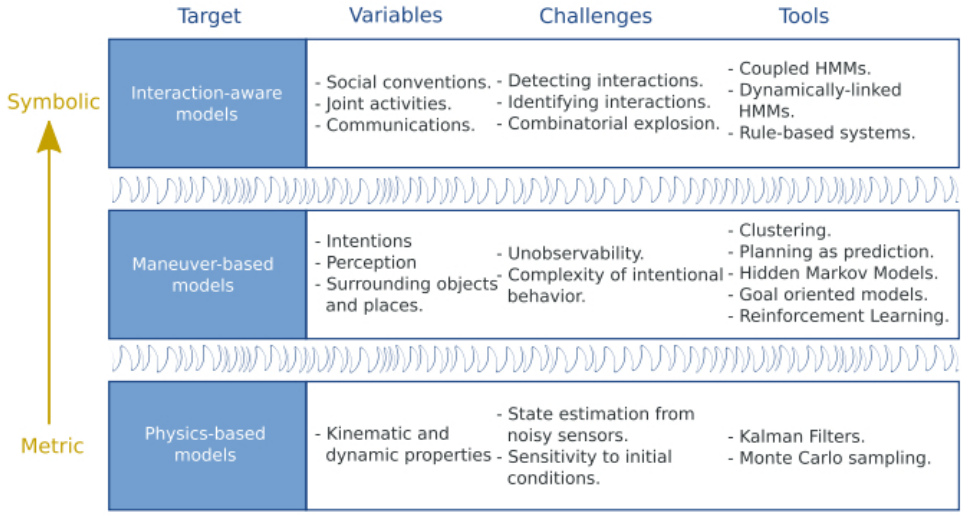


Figure 2.1.: Motion Modeling Overview [3]

Together with above-mentioned categories, authors of [21] introduced one more category to predict movements - ***data-driven based***.

- **Data-driven** based motion and trajectory prediction can be classified into clustering-based and probabilistic approaches. A wider description is in subsection 2.3.4.

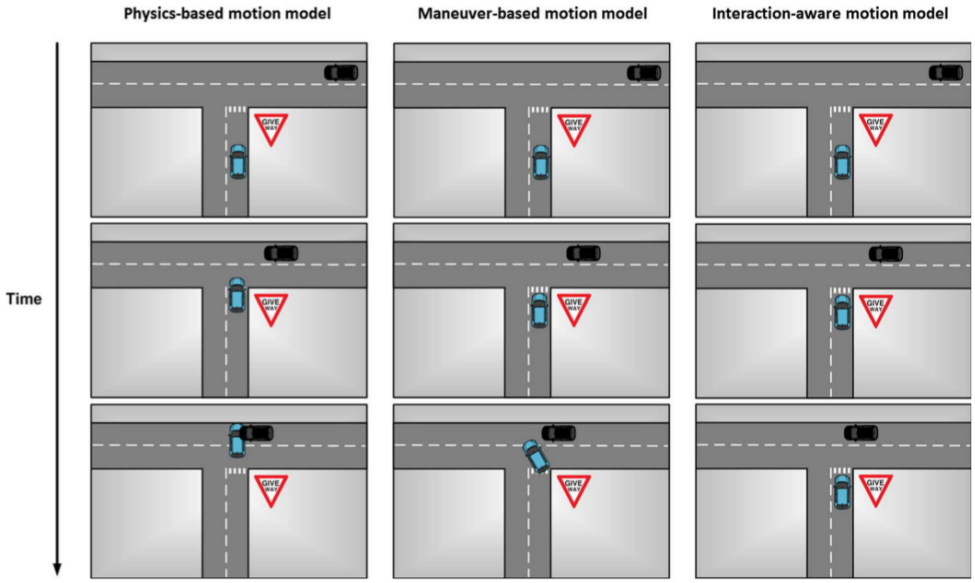


Figure 2.2.: Examples of motion prediction with the different types of motion models [3]

2.3.1 Movement Prediction Using Physics-Based Models

Physics-based movement prediction models imply vehicles as a dynamic item, controlled by the physics' laws. Movements are predicted using dynamic and kinematic models with control inputs (e.g. acceleration, deceleration, steering), properties of the car (e.g. length, weight) and some external conditions (e.g. the friction coefficient of the road surface) to the process state of the vehicle (e.g. position, speed, direction). Great work has been done using *physics-based* motion models and it still remains the most commonly used motion models for motion prediction in the context of road safety. The complexity of the models depends on a representation of the dynamics and kinematics of a vehicle, as well as, how uncertainties are handled, whether or not the road geometry is taken into account, etc.

Dynamic and kinematic models can be used for movement prediction in a lot of different ways, the main difference is how uncertainties are handled. Three main approaches will be described as follow: ***single trajectory simulation, Gaussian noise simulation*** and ***Monte Carlo simulation*** [3].

- **Single Trajectory Simulation.** The simplest method to predict future movements and trajectory of a car is to apply simple dynamic or/and kinematic models for the current state of a car while assuming that the current state of the car is determined with absolutely highest confidence and applied model (dynamic, kinematic or both) is the perfect representation for the movement of the car. This simple approach was used in [22] using dynamic and [23, 24] kinematic models. The main benefit of this straightforward approach is computational efficiency, that allows this method to be used in real time. On the other hand, predictions made by this method do not consider uncertainties of the current state and as a result, predicted movements and trajectories are not trustworthy for use in a long term ($>$ 1 sec.) predictions.

- **Gaussian Noise Simulation.** The uncertainty of the current state of a vehicle and its evolution during the time is a very important factor in movement or trajectory prediction and it can't be avoided. In [25, 26, 24] this is modelled using a normal distribution. Gaussian Noise function is very popular because of its uncertainty representation in Kalman Filter (KF), which is still a conventional method for vehicle state estimation having noisy sensors measurements in account. There are some cases where dynamic, kinematic and sensor models are linear and uncertainty is modelled using a normal distribution instead of KF Bayesian filter is used. Filtering mainly contains of two steps: prediction and update steps. In the first time step at time step t, a current state of the vehicle is is given to the dynamic or kinematic model, which gives predicted state for the next time step which has a Gaussian distribution shape. In the following step, predicted state of the next time step is combined with sensor measurements of the same time step, which is Gaussian distribution as well. Filtering is a looping of these two steps every time when new measurements are available.

  By looping the first step, it is possible to get a mean and covariance matrix for every future timestep for the vehicle state. This can be modified into a trajectory mean with linked uncertainty (i.e. normal distribution in each timestep), as showed in [27, 25]. As compared to the approached of *single trajectory simulation*, Gaussian Noise

simulation techniques have the benefit of uncertainty representation on the predicted trajectory or movements. However, there are some limitaitons as well: modelling uncertainties employing normal distribution is not quite enough to show the different possible maneuvers. A possible solution for this could be uncertainty representation using Variational Gaussian Mixture Model (VGMM). Author of [28] used Switching Kalman Filter (SKF) for this exact purpose. [26] depends on mass of KF to show possible models for movement evolution for vehicle and be able to freely change between them. [24] introduced an alternative approach: to use heuristics and change different kinematic model depending on the current situation.

- **Monte Carlo Simulation.** In generic case when no assumptions in advance are made about models linearity or uncertainty model, distribution expresion on predicted vehicle states are not clear. Monte Carlo method is the right tool for this kind of situation. The idea under the Monte Carlo method is to randomly sample the input of the dynamic or kinematic model and to generate potential future trajectories. If the road topology is taken into account, various mass can be added to the generated trajectories and movements to penalize the ones which do not respect the restriction of the road design. Kinematic and dynamic models can be used for Monte Carlo method by categorizing inputs instead of considering them as a constant. Typical inputs are categorized to acceleration, steering angle or lateral deviation. To be able to take into account eligibility of the movement, generated trajectory samples, which has a bigger acceleration than physically is allowed can be removed, as it was done in [29] or consider limitations which vehicle has (weight, length, etc.) and distribute dynamic and kinematic models in a more realistic manner and remove all impracticable trajectories from predefined trajectories list as it was done in [30]. Monte Carlo method can be used to foresee trajectory or movements for a vehicle with a very well known current state or for vehicle which has uncertainty in the current state, which was estimated by one of the filtering algorithms.

2.3.2 Movement Prediction Using Maneuver-Based Models

Maneuver-based motion models show vehicles as independent moving entities, i.e. it is assumed that the movement of a vehicle on the road match to a series of independently executed movements from the other vehicles on the same road. Oxford dictionary [31] a movement/maneuver as “a physical movement or series of moves requiring skill and care”. Term behaviour in literature often is used meaning the same meaning, e.g. in [32, 33, 34], for the sake of simplicity word "movement" or "maneuver" will be used in this work with defined meaning. Movement and trajectory prediction using maneuver-based motion models work with in advance recognized movements which driver possibly intend to perform. If an algorithm can recognize intended movement, the algorithm can assume that future actions of the driver will match the recognized movement. Due to this a *priori* information, trajectories received with this method are more relevant and reliable than the ones received using physics-based motion models. Maneuver-based motion models rely on prototype trajectories or on movement intention estimation.

Vehicle motion classification into maneuver/movement classes has been extremely widely applied not only in driver assistance systems but into natural driving studies [35, 36, 37, 38, 39, 40, 41, 42, 43, 44]. Authors of the majority of approaches are using heuristinc [38] or training classifiers like SVMs in [39], HMMs [35, 40, 41]. Long Short Term Memorys (LSTMs) in [42], Bayesian networks [43], etc., as movement-based features using speed, deceleration, acceleration, yaw rate, lane position, turn signals, distance from other vehicle and other road context information. Authors of [38] classified vehicle’s movement into class "keep lane" or "change lane" grounded on how far the closest car is and predicted future trajectory by applying quintic polynomial of the current car movement state and pre-defined ultimate movement state for each movement class, defined before.Authors of [43] used six different movement classes, which were defined before and using DBN based on multiple movements and context based features selected the potentially right future movement. Authors of [44] defined an individual Gaussian process for three movement classes and established a multi-modal distribution for possible future trajectories using each model. However, in the study, only one case-based prediction has been introduced. Authors of [35] also determined separate Gaussian processes, this time for four different movement classes, which were classified using a hierarchical HMM. This method was tested on real highway data. Authors of another study [36] used a random forest classifier for movements classification into pre-defined movement classes: left or right lane changes or keep lane. Authors used a separate Gaussian Mixture Regression (GMR) model for predicting lateral movement for vehicles using each class. Method was tested on real highway data. Similar method, but without predifined movements classes for prediction longitudinal motion for vehicles were used in [37].

2.3.3 Movement Prediction Using Intention Aware Models

Interaction-aware motion models introduce cars as manoeuvring items which co-operate with each other, i.e. a movement of a vehicle is considered to be affected by a movement of the other moving object in the traffic scene. Keeping into

account the dependencies between the separate moving objects leads to a much better explanation of their movement compared with **maneuver-based** motion models described in the previous subsection. As a result, it gives a better perception of the current situation.

Despite this, a relatively small amount of researches is done considering inter-moving-objects interaction in movement prediction. Authors of [45] assigned two movement classes for vehicles approaching an intersection together, applying a polynomial classifier which "punishes" cases that potentially would lead to near-collisions situations. Authors of [46] worked with a much complex scenario and assigned movement classes to multiple together interacting vehicles in a highway scenario. However, foreseen movements, trajectories of a vehicle are assumed to be given in advance. Results reported using a simulated environment. [21] in their work considered multiple interacting vehicles together with the difficulty of estimating their future motion. Authors of [36] not directly used inter-moving-objects interaction by including comparative positions and velocities of vehicles close by as features for movement and trajectory prediction.

2.3.4 Movement Prediction Using Data-Driven Model

As mentioned earlier *data-driven* movement prediction can be generally classified into clustering-based and probabilistic approaches. **Clustering-based** approaches group the training data in order to provide a set of possible prototype trajectories [47, 48]. Partially observed trajectories are checked and compared with a prototype trajectory using various distance measurements, as Dynamic Time Warping (DTW), Longest Common Subsequence (LCSS), Hausdorff distance, etc. and after matching movement trajectory with prototype trajectory, later one is used as a model for future movement. Clustering approach is quite easy, but the main disadvantage of this method is the deterministic nature of the predictions.

Probabilistic approach contrary learn probability distribution of every movement trajectory class and gives the conditional distribution for future movements, given current trajectory. This lets us avoid some degree of natural uncertainty of predicting the future.

Authors of [49, 35] for modelling trajectories and for motion prediction use Gaussian Processes which are the most popular approaches solving prediction problems so far. [36] uses GMR for prediction longitudinal movement of a vehicle, while [37] uses the same method for lateral movement prediction. [50] uses VGMMs for conditional distribution within snippets of future having snippets of movement history models. The latest approach is much easier and computationally more effective when compared to Gaussian Process Regression. Authors proved the efficiency of method predicting non-linear movements in turns at the intersection scenarios.

2.3.5 Limitations of Methods for Movement Prediction

Subsections 2.3.1, 2.3.2, 2.3.3 and 2.3.4 described movement prediction with different feature based model. This subsection will introduce limitations of all these methods.

- **Physics-based approach.** Predictions using physics-based motion models are restricted to very short-term (< 1 sec.) motion prediction due to low-level motion (dynamic and kinematic) properties this method relies on. Usually using this method it is unable to foresee any change in the vehicle movement which happens due to an execution of a particular maneuver (e.g. speed up, slow down, make a turn, etc.), or changes caused by external factors (e.g. slowing down due to traffic lights, signs, other vehicles, etc).
- **Maneuver-based approach.** For a very long time, the biggest limitation of prototype trajectories was time representation. When the movement models are showed using a finite set of trajectories it takes a very large number of prototypes to represent the large variation in the implementation of an every possible movement pattern. Handling subtle situation in traffic, as movements with waiting time at a stop line, not constant velocity caused by traffic is a very big issue for such models. For a certain extent, Gaussian Processes (GP) were introduced. They solved this kind of problem by introducing time-independent movement patterns [49]. On the other hand, GPs have some other limitations as well. First of all to be able to take into account all possible traffic scenarios, has very heavy computational time, despite that they are not considering the physical limitations of a vehicle and due to that may generate or predict unrealistic trajectories and movements. To solve these problems the best solution so far was proposed in [51]. Authors used Rapidly-exploring Random Tree (RRT) to be abe to "randomly sample points toward dynamically feasible trajectories, using as inputs the current state of the vehicle and the sample trajectories generated by the GPs" [51]. Another issue with using predefined prototype trajectories is an adaptation to a different road, i.e. for different intersections. Each movement model is defined for a specific road/intersection geometry and topology, what means that prototype models only can be used with the same or very similar topology.

Maneuver-based approach contains similar limitations which described under limitations of data-driven approach.

- **Interaction-aware approach.** Prediction using interaction-aware motion models are the most exhaustive method suggested in the literature so far. Using it, is possible to predict for a longer-term as compared to physics-based motion prediction models, and predictions are more trustworthy than using maneuver-based motion models in predictions due to taking dependencies between the surrounding cars into consideration. However this completeness has some disadvantages as well: calculation of all possible trajectories with all possible models take a lot of time and because of that, it is not very compatible with using in real-time situations. For this reason, using interaction-aware motion predictions are not so popular.

- **Data-driven approach.** The assumption that movement of vehicles do not depend on each other and other traffic participants is not accurate. All vehicles without any exceptions use a road together with other traffic participants and movement performed by one vehicle directly or indirectly effects others. Dependencies with each other are quite strong at intersection, where road rules, not only movement of other cars must be taken into account. Ignoring these reliances can lead to wrong interpretations of the situations, and affects the evaluation of the risk. A data-driven approach is quite easy, but not always it pays attention to these critical dependabilities and it is quite difficult to pre-define all possible action of other traffic participants, i.e. the main disadvantage of this method is the deterministic nature of the predictions.

3 TBD

Probably I need to restructure content, but not sure exactly where to put what

4 Setup and Implementation

ROS simulation environment was used for testing approaches. ROS is a framework where testing cases can be easily developed and tested. In a current simulator, there are few small environments imitated consisting of a static map (T and X intersection) and the car itself.
In order to visualize simulation RViz which is s a 3D visualizer for showing data and state information from ROS was used. A snapshot of the simulation environment is in Figure 4.1. The code was written in Python and C++ programming languages.

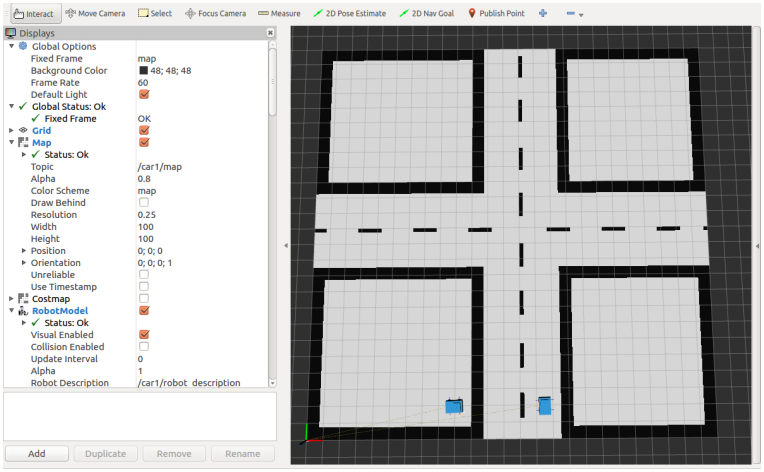


Figure 4.1.: The RViz environment that runs the simulation, having X-Intersection in mind

For this project, several testing cases were developed. Methods tested offline (with having all trajectory for testing) and online (having only current and last position). Data from every run is stored in order to analyze it and see how each method effect performance of the system. Parameters and results for each trajectory are independent and tested several times in a specific environment to be able to draw conclusions from the results we received during the testing.

4.1 Data Collection

To be able to test our approach database of various trajectories need to be used. To make our own database for possible trajectories and not to use already existing data was taken at the beginning stage of this project. Due to that data collection was one of the first steps after all simulation environment was set.

To be able to control car in ROS environment and record data joy package was used. Joy package is a ROS driver for a "generic Linux joystick".The package consists of joy_node, a node which allows communication between a generic joystick and ROS. Joy message, which contains information about the current state of each button on joystick is published by node [52], having this information command start to move, turn to any direction, stop is sent to the object we want to control in ROS environment. When an object, the car in our car is able to move in ROS using a joystick we recorded and stored a car's position every few milliseconds.

For each map type, slightly different types of trajectories were recorded. For X-intersection a various number of trajectories for movement to the right, straight and left were recorded, while for T-intersection only movement t the right and left was recorded.

4.2 Brief Algorithm Explanation

In this section a brief overview of the main code algorithm for getting test results will be explained.

4.2.1 Map Recognition

In this project, two (so far) maps, imitating X and T intersection, were used. Map recognition is necessary because the further calculation is a bit different on T and X intersection maps.
The map is recognized using .... Not exactly sure if I will leave current map recognition algorithm, where I am using simple contour recognition or will go to a more complicated one. That's why here I left this section empty.

4.2.2 Trajectories' Unification

The problem with our current database is that there are a lot of trajectories which have a different number of time steps. This happened because of different trajectory length (e.g. to go straight takes less time and it is a less distance than going to left or right), the velocity of the car may differ in each or even in the same trajectory (velocity depends on control of the joystick), etc. Having different time steps in trajectories makes some problems in further calculations, comparison of results, the calculation over thousands of steps also has a longer computation time, etc. Because of that trajectories, unification is a necessary step.

Unification made using interpolation when all trajectories transformed to have an equal number of time steps, in spite of the original number of time steps. This is achieved by interpolation, in mathematics interpolation is a method of building new data points inside the range of a discrete set of known/given data points.

The pseudo code below (Figure 4.2) shows the way of interpolation in this project.

```
1  begin
       x, y        # coordinates of trajectory which needs to be interpolated

       # Calculate the n-th discrete difference along the given x and y axis
2      xd[n] = a[n+1] - a[n]
3      yd[n] = a[n+1] - a[n]

       # Calculate Euclidean distance between xd[n] and yd[n]
4      dist = np.sqrt(xd[n] ** 2 + yd[n] ** 2)

       # Calculate cumulative sum of the elements along dist
5      u = np.cumsum(dist)

       # Stack array u in sequence column wise (horizontally)
6      u = np.hstack([[0], u])

       # Calculate evenly spaced numbers over a specified interval [start, stop]
7      t = np.linspace(0, u.max(), len_des)

       # Calculate the one-dimensional linear interpolation with given discrete data points
8      xn = np.interp(t, u, x)
9      yn = np.interp(t, u, y)
10 end
11 return xn, yn # interpolated coordinated of given trajectory
```

Figure 4.2.: Pseudo code for interpolation (need to rewrite it to make it pseudo)

Figure 4.3 shows 3 original trajectories and interpolated version of the same trajectories. Original trajectory, which goes to the right has 7,581 time steps, while straight trajectory has 13,666 and the left trajectory has 10,929 time steps after interpolation all trajectories contain 10 steps (number of time steps can be changed according to preference or test case). As can be seen, interpolation does not ruin trajectories and can be used in further calculations.

4.3 Modeling Belief for Prediction Making

A Bayesian version of a Gaussian mixture model is used for calculating the belief for each trajectory class and this section gives definitions for beliefs representing for a car over trajectory classes (going right, going straight, going left). Since

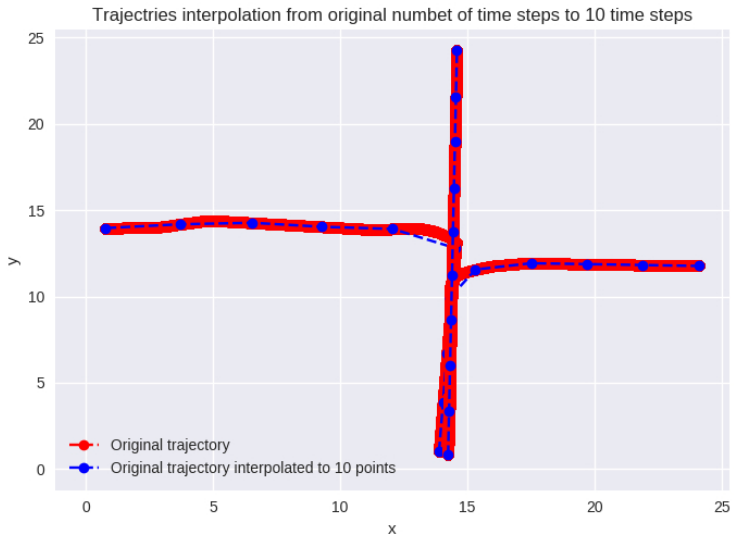


Figure 4.3.: Original and Interpolated Trajectories

we can only observe the current position of the car, which means that trajectory class is partially observable, due to that our belief is represented as a probability distribution over all trajectory classes.

For maintaining correct belief, belief update must be done every time step. Updating belief constantly is important because of sudden position change can show drivers' intentions and what his next steps could be. To be able to predict possible future movement current position information need to consider every time when belief update is calculated. The belief update is calculated using Bayes rule formula 4.1 below:

$$\begin{aligned} b_{t+1}(k) &= Pr(k|o_t, b) \\ &= \frac{Pr(o_t|k, b)Pr(k|b)}{Pr(o_t|b)} \\ &= \frac{Pr(o_t|k, b)Pr(k|b)}{\sum_k Pr(o_t|b)Pr(k|b)} \\ &= \frac{Pr(o_t|k, b)b_t(k)}{\sum_k Pr(o_t|b)Pr(k|b)} \end{aligned} \tag{4.1}$$

With given formula belief for future step $b_{t+1}(k)$ for a predefined class is calculated. $Pr(o_t|k, b)$ is the car position observation model, which returns the probability (or likelihood) of going to any direction from the current position, observed with o_t. This likelihood can be calculated using multivariable Gaussian probability distribution function f(x,μ,Σ). This probability distribution function looks the following:

$$f(x, \mu, \Sigma) = \frac{1}{\sqrt{(2\pi)^n det(\Sigma)}} exp(-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)), \tag{4.2}$$

where dimensionality n = 2 and x is a currently observed position of the car $x = (o_t^x, o_t^y)^T$, μ is a mean from all predefined trajectories in data base for the same class $\mu_k = (\mu_{t,k}^x, \mu_{t,k}^y)^T$ and Σ is a covariance matrix of all predefined trajectories in data base for the same class $\Sigma_k = (\Sigma_{t,k}^x, \Sigma_{t,k}^y)^T$.Mean for x and for y coordinates can be found using formula 4.3 and formula 4.4 respectively:

$$f(\mu_x) = \frac{\sum_{i=1}^n x_i}{n} \tag{4.3}$$

$$f(\mu_y) = \frac{\sum_{i=1}^n y_i}{n} \tag{4.4}$$

Covariance value can be calculated using formula 4.5

$$f(\Sigma_{x.y}) = \frac{\sum_{i=1}^{n}(x_i - \mu_x)(y_i - \mu_y)}{n-1} \tag{4.5}$$

The denominator of formula 4.1 is the so-called normalization factor, which sums all likelihoods of the car over all movement classes at the current time step. The final result of formula will return updated belief that car is moving towards any of classes.

Figure 4.4 shows pseudo-code how belief updates are made over time, as an input having current belief b_t at time step t, current car position x_t, y_t from the last made observation, as well we have before calculated mean and covariance values at that time, $\mu^x_{t,k}, \mu^y_{t,k}$ and $\Sigma^x_{t,k}, \Sigma^y_{t,k}$ respectively.

```
1  begin
     b_t          #current belief
     x_t, y_t     #observed current car position
     Σ_t          #mean (x and y values) of predefined trajectories in current time step
     μ_t          #covariance matrix (2x2 size) of predefined trajectories in current time step
     b_t+1 ← Ø    #updated belief (empty set)
     η ← Ø        #normalization factor (empty set)

   #normalization factor calculation for each observation probability
2    for k∈K do
3      p_k = f((x_t, y_t), μ_t, Σ_t) from N((x_t, y_t) | μ_t, Σ_t)
4      η = η + p_k
5    end

   #new goal distribution over classes calculation
6    for k∈K do
7    b_t+1(k) ← p_k b_t(k) / η
8    end
9  end
10 return b_t+1
```

update.jpg

Figure 4.4.: Pseudo Code for Updating belief

4.4 Trajectory Scaling

Computation time for prediction is one of the main things which needs to be fast. Even though more belief updates give more precise results, more computations take more time. In this project, we took various numbers of time steps and compared results to get them as precise as possible and have fewer time steps (e.g. we aimed to keep precision of trajectory of 100-time steps but have only 10-time steps trajectory).

There were a lot of various trajectories with various numbers of points, but with an original approach result still was that with more time steps prediction results were more precise. Later on, Toy Problem approach came into the sight. [53] defines it as "In scientific disciplines, a toy problem is a problem that is not of immediate scientific interest, yet is used as an expository device to illustrate a trait that may be shared by other, more complicated, instances of the problem, or as a way to explain a particular, more general, problem-solving technique".

After unsuccessfully trying different methods, the idea of raising a likelihood (formula 4.2) of the trajectory with a bigger number of time steps at every time step by $\frac{1}{\frac{biggernumberoftimestepsintrajectory}{smallernumberoftimestepsintrajectory}}$ and then compare results of matching points in each trajectories.

To simplify all this, let's have an example: let's imagine we have the trajectory interpolated to 100-time steps and we want to see how results differ when we do belief updates all 100 times with doing belief at every 10th step (10th, 20th,

..., 90th, 100th). Without doing any changes in the original code, these results are not matching, in fact, they differ quite a lot. But if we raise likelihood (formula 4.2) of the trajectory where belief is updating 100 times by $\frac{1}{\frac{100}{10}} = \frac{1}{10}$ at every time step and then do belief updates as normal and compare them with belief updates which are calculated every 10th step with making no changes in the original code. After comparison of these results, it was easy to see that results are matching or are close enough to each other. The pseudo code of the given example is in Figure 4.5.

```
1  begin
       N            #number of time steps (e.g. 100)
       n            #number of time steps for scaling (e.g. 10)
       b_t          #current belief
       x_t, y_t     #observed current car position
       Σ_t          #mean (x and y values) of predefined trajectories in current time step
       μ_t          #covariance matrix (2x2 size) of predefined trajectories in current time step
       b_t+1 ← Ø    #updated belief (empty set)
       η ← Ø        #normalization factor (empty set)

       #belief  update calculation for N time steps
2      for i in range (1, N+1) do
           #normalization factor calculation for each observation probability

3          for k∈K do
4              p_k = f((x_t, y_t), μ_t, Σ_t)^1/(N/n) from N((x_t, y_t) | μ_t, Σ_t)^1/(N/n)
5              η = η + p_k
6          end

           #new goal distribution over classes calculation
7          for k∈K do
8          b_t+1(N)(k) ← p_k b_t(k) / η
9          end

           #take the every nth step from trajectory with N steps
10         if i % n == 0 do

11 - 17    #Belief update calculated exactly the same as in Figure 4.4

18     end
19 end
20 return b_t+1(N), b_t+1(n)
```

Figure 4.5.: Pseudo Code for Scaling Trajectory for Belief Update

Having this in mind we can make our prediction making the process faster and more precise.

4.5 Online Method for Prediction Making

The online method uses the same, previously described techniques and principles for calculation. The predefined database with trajectories exists, calculations and all preparation for a future working with them are made as well. The one difference between online and offline methods is that all testing and its results visualizations are happening in ROS and RViz. And the main difference from an offline method, the online method does not have a whole testing trajectory at the beginning, during testing only past and current steps are known. which makes all interpolation kinda complicated, and I need to discuss that with you on a meeting, maybe I am just don't know something and I am raising a problem when it doesn't exist. Results of prediction are also highlighted in real time in RViz environment.

5 Experiments and Results

In this chapter experiments, made by using different simulation setups, are described. The number of experiments for each trajectory class and for each map is done. Results were analyzed separately and then compared.

5.1 Vehicle is Moving Through X-Intersection

Every time we start belief update calculation we have to initialize the value of belief at the very first time step. When the testing map is X-intersection (Figure 5.1), from its' starting position, which is $x_0, y_0 = (14.5, 2.0)$ car can go to any direction it wants: right, straight or left. There are equal chances that car will go to any direction, so initial belief set to be equal for each direction $b_0 = (0.333, 0.333, 0.333)$

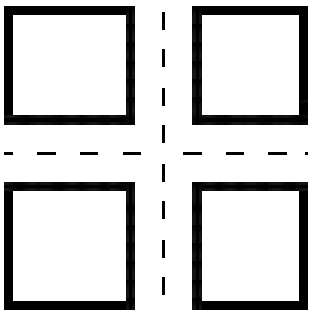


Figure 5.1.: X-intersection map

5.1.1 Moving to the Right Direction

The testing trajectory of going to right looks as it is shown as a red line in Figure 5.2.

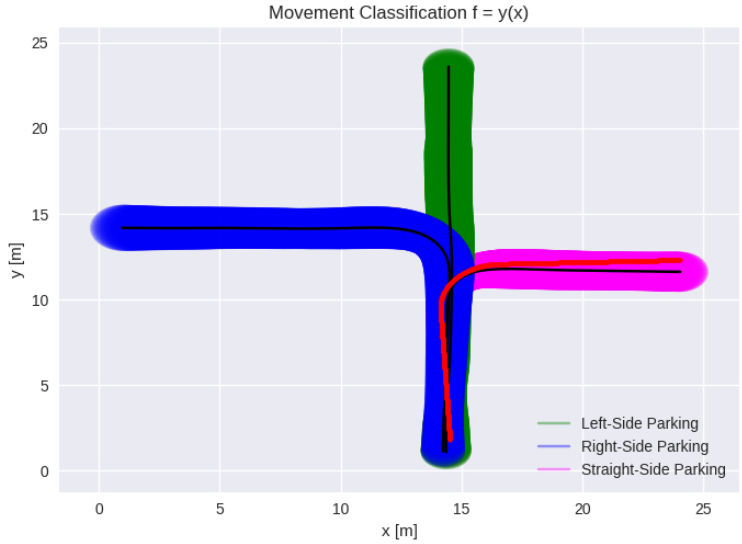


Figure 5.2.: Testing Trajectory (red) of going to the right

At first, the prediction making was checked for all trajectory, when it has 10-time steps. Results are in Figure 5.3.
From the series of plots, we can see that in the 7th step belief that direction is right equals to 1. In Figure 5.4 is shown how beliefs are changing over time. Left graph of the Figure 5.4 shows belief changes when trajectory has 10 points and

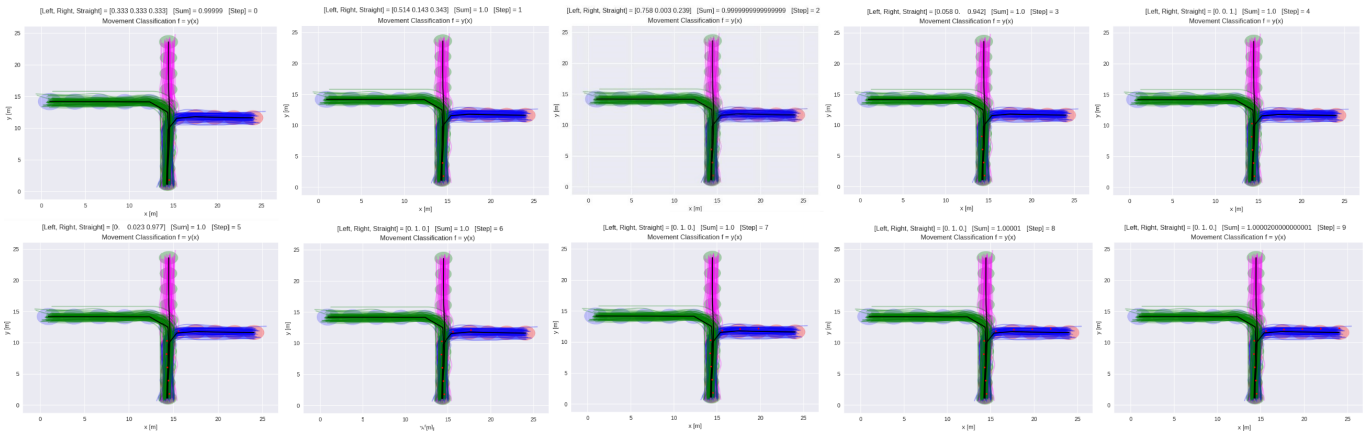


Figure 5.3.: Prediction making for trajectory which goes to the right. Trajectory has 10-time steps

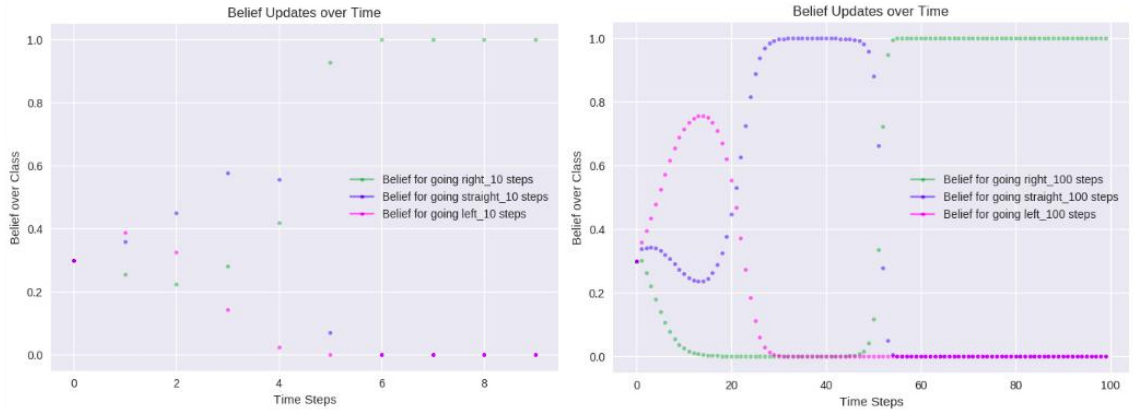


Figure 5.4.: Belief changes over time. For trajectory with 10 steps on the left, for trajectory with 100 steps on the right. Trajectory direction is right

for better visibility, there was added one more graph on the right, where is the same trajectory, just it was interpolated 100 times instead of 10.
To be able to see how belief is changing over time withing different trajectories and compare if there are the same results at the same steps in the same movement class, we plotted 10 random trajectories going to the right. Results are shown in Figure 5.5.
From the Figure 5.5 it is possible to see that some trajectories have a better time for being sure to which direction car is moving and some of them have a worse time than others. Next two figures will show trajectories which have "the best" (Figure 5.7) and "the worst" (Figure 5.6) times in the movement prediction.
By looking at Figure 5.6, "the worst" results can be explained by the position of the trajectory: it is close to all means, so it is natural that precise of prediction making can be disturbed in this case.
By looking at Figure 5.7 and "good" results we can also make an assumption that trajectory is not on the means of classes and due to that precise of prediction making is better in this case.

5.1.2 Moving Straight

The testing trajectory of going straight looks as it is shown as a red line in Figure 5.8.
At first, the prediction making was checked for all trajectory, when it has 10-time steps. Results are in Figure 5.9.
From the series of plots, we can see that in the 4th step belief that direction is straight is equal to 0.983. In Figure 5.10 is shown how beliefs are changing over time. Left graph of the Figure 5.10 shows belief changes when trajectory has 10 points and for better visibility, there was added one more graph on the right, where is the same trajectory, just it was interpolated 100 times instead of 10.
To be able to see how belief is changing over time withing different trajectories and compare if there are the same results at the same steps in the same movement class, we plotted 10 random trajectories going straight. Results are shown in Figure 5.11.

Figure 5.5.: Belief changes over time for different trajectories which belong to the same movement class. Trajectories have 100 time steps. Trajectory direction is right

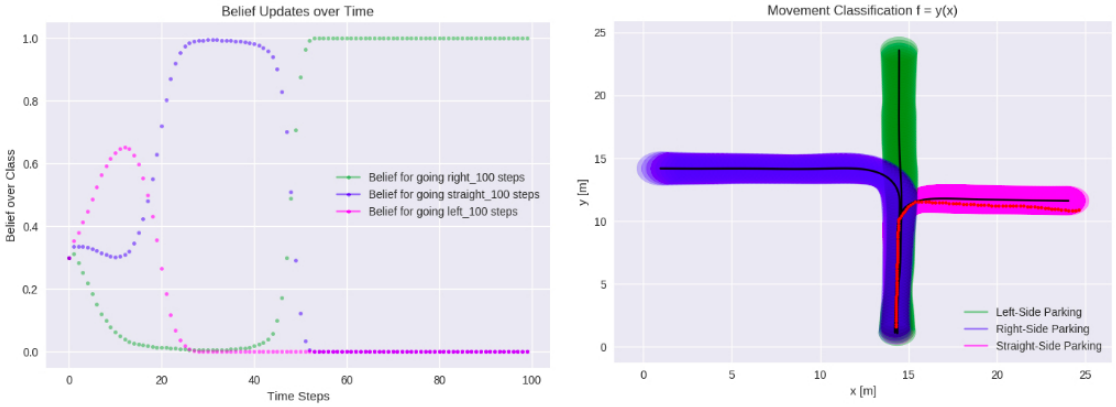


Figure 5.6.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is right

From the Figure 5.11 it is possible to see that some trajectories have a better time for being sure to which direction car is moving and some of them have a worse time than others. Next two figures will show trajectories which have "the best" (Figure 5.13) and "the worst" (Figure 5.12) times in the movement prediction.

By looking at Figure 5.12, "the worst" results can be explained by the position of the trajectory: it is closer to standard deviation values of left class mean, this could be the case, why prediction is that car is going to the left, until it is sure that it is not going to the left.

By looking at Figure 5.13 and "good" results we can also make an assumption that trajectory and coordinates at each time step is closer to mean and it is in the range of standard deviation of straight mean.

By looking into separate results of class straight one more very interesting case was noticed. Here (Figure 5.14) prediction is correct from the very first steps, but the changing of dynamics of prediction is interesting. This dynamics might be explained by car movement trajectory (it is not moving exactly straight, it is making some turning, which can be the reason of this result).

5.1.3 Moving to the Left Direction

The testing trajectory of going to the left looks as it is shown as a red line in Figure 5.15.

At first, the prediction making was checked for all trajectory, when it has 10-time steps. Results are in Figure 5.16.

From the series of plots, we can see that in the 3rd step belief that direction is left is equal to 0.935. In Figure 5.17 is shown how beliefs are changing over time. Left graph of the Figure 5.17 shows belief changes when trajectory has 10

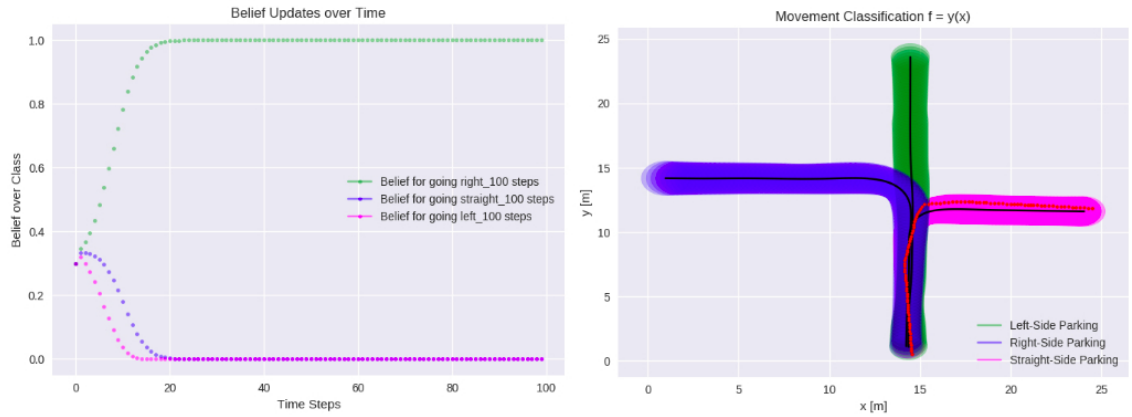


Figure 5.7.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is right

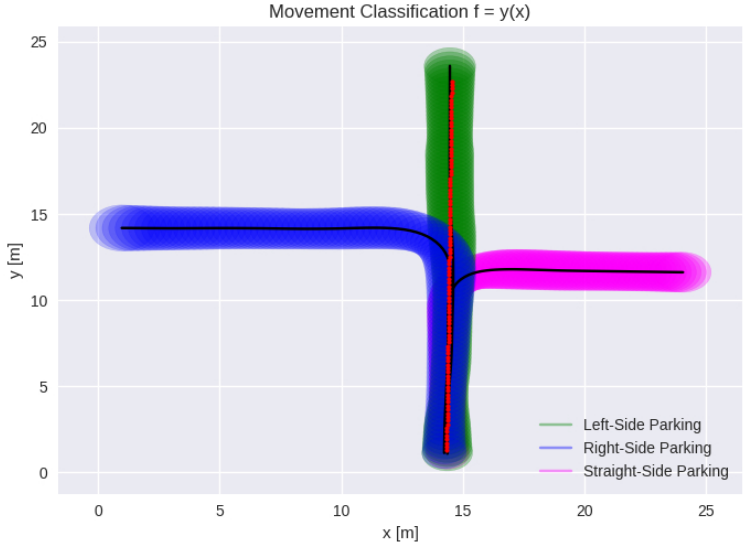


Figure 5.8.: Testing Trajectory (red) of going straight

points and for better visibility, there was added one more graph on the right, where is the same trajectory, just it was interpolated 100 times instead of 10.
To be able to see how belief is changing over time withing different trajectories and compare if there are the same results at the same steps in the same movement class, we plotted 10 random trajectories going straight. Results are shown in Figure 5.18.
From the Figure 5.18 it is possible to see that some trajectories have a better time for being sure to which direction car is moving and some of them have a worse time than others. Next two figures will show trajectories which have "the best" (Figure 5.20) and "the worst" (Figure 5.19) times in the movement prediction.
TO THINK WHY IT IS DIFFER

5.2 Vehicle is Moving Through T-Intersection

In this section results of simulation setup for T-Intersection (map looks like it is shown in Figure 5.21) will be described. As in the case of the map with X-Intersection, starting position of the car is $x_0, y_0 = (14.5, 2.0)$, but in this setup, the car can go only to the left or right, straight direction does not exist in this case. As in the previous case, at the very beginning, initial belief must be set. Since we already know that there is no way that car is going straight (there is no straight from car starting position), we have so-called some prior information. Initial belief for going right and left is calculated by formula $\frac{\#of\,trajectories going to the right in our database}{\#of\,trajectories going to the right in our database + \#of\,trajectories going to the left in our database}$ and $\frac{\#of\,trajectories going to the left in our database}{\#of\,trajectories going to the right in our database + \#of\,trajectories going to the left in our database}$ respectively. After calculation initial belief set to be $b_0[left, right, straight] = (0.533, 0.467, 0.000)$

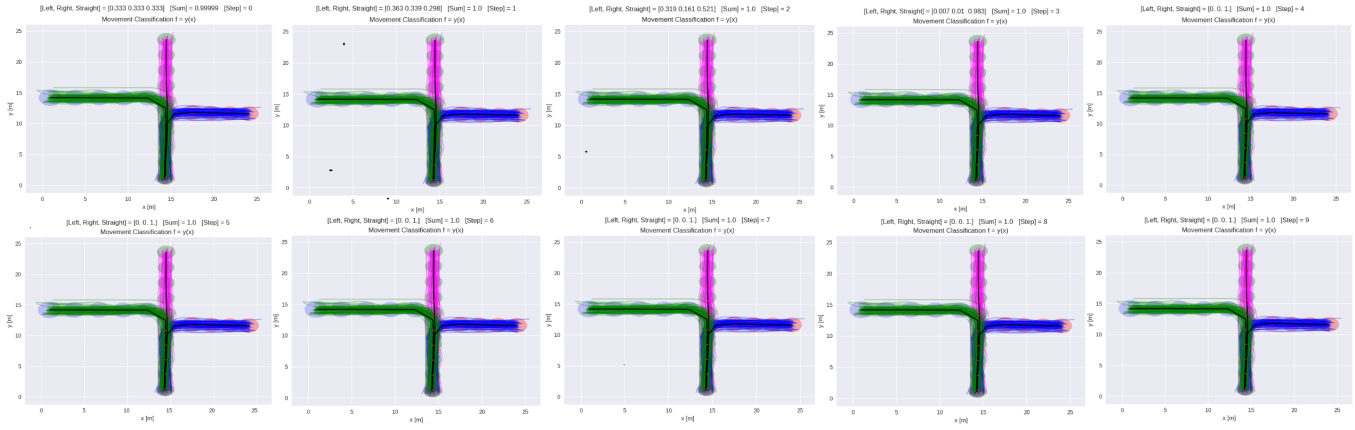


Figure 5.9.: Prediction making for trajectory which goes straight. Trajectory has 10-time steps

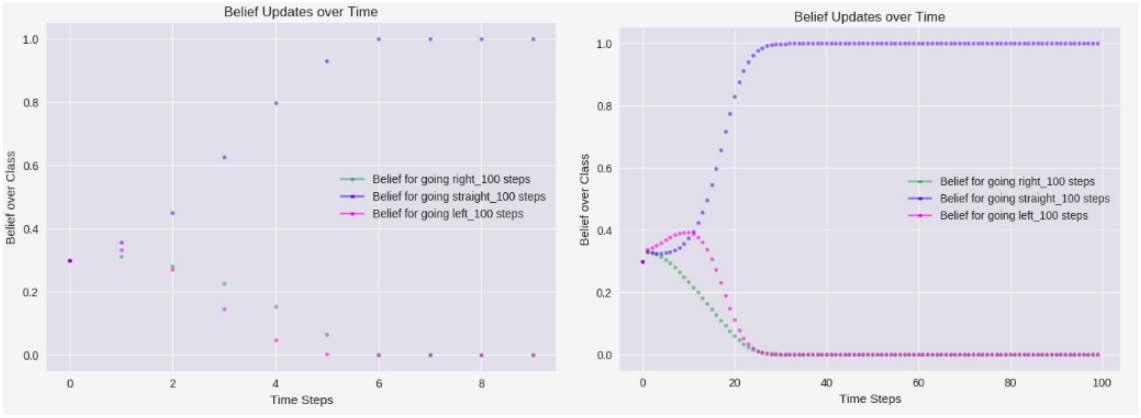


Figure 5.10.: Belief changes over time. For trajectory with 10 steps on the left, for trajectory with 100 steps on the right. Trajectory direction is straight

5.2.1 Moving to the Right Direction

The testing trajectory of going to right looks as it is shown as a red line in Figure 5.22.
In Figure 5.23 is shown how beliefs are changing over time. Left graph of the Figure 5.23 shows belief changes when trajectory has 10 points and for better visibility, there was added one more graph on the right, where is the same trajectory, just it was interpolated 100 times instead of 10.
To be able to see how belief is changing over time withing different trajectories and compare if there are the same results at the same steps in the same movement class, we plotted 10 random trajectories going to the right. Results are shown in Figure 5.24.
From the Figure 5.24 it is possible to see that correct prediction is made much faster than in case of map with X-Intersection (Figure 5.5).

5.2.2 Moving to the Left Direction

The testing trajectory of going to right looks as it is shown as a red line in Figure 5.25.
In Figure 5.26 is shown how beliefs are changing over time. Left graph of the Figure 5.26 shows belief changes when trajectory has 10 points and for better visibility, there was added one more graph on the right, where is the same trajectory, just it was interpolated 100 times instead of 10.
To be able to see how belief is changing over time withing different trajectories and compare if there are the same results at the same steps in the same movement class, we plotted 10 random trajectories going to the right. Results are shown in Figure 5.27.
As we could see from the Figure 5.24, the same results we can see from Figure 5.27: the correct prediction is made much faster than in case of map with X-Intersection. This happened because of prior knowledge.

Figure 5.11.: Belief changes over time for different trajectories which belong to the same movement class. Trajectories have 100 time steps. Trajectory direction is straight

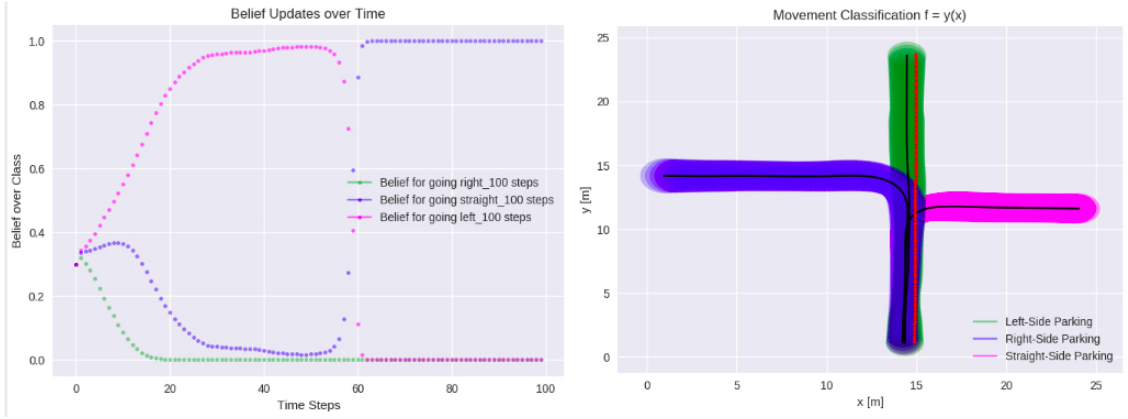


Figure 5.12.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is straight

5.3 Results Comparison Before and After Scaling

As mentioned before trajectory scaling while updating belief can allow not to make calculations so frequently, but still, keep the same precision of results as having a lot of belief updates.

One test on scaling was done having one trajectory interpolated 100 times, at every step in 100-time step trajectory likelihood (formula 4.2) was powered by 0.1 and results were compared with taking every 10th step from the same trajectory, but with calculation as normal. The scaling algorithm is described in Figure 4.5. To show the difference between scaled results and not scaled results two different graphs will be shown for one test.

5.3.1 X-Intersection

X-Intersection has three directions and scaling within all three of them will be shown.

5.3.2 T-Intersection

X-Intersection has two directions and scaling within right and left directions will be shown.

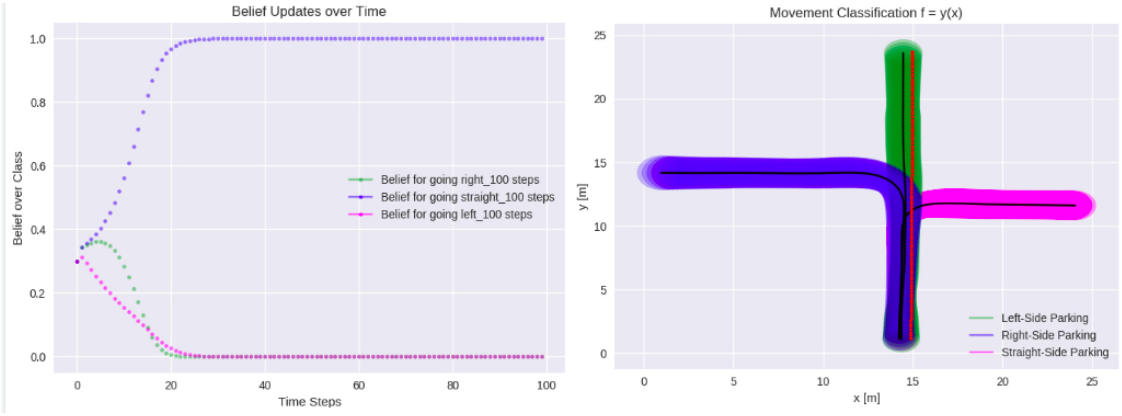


Figure 5.13.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is straight

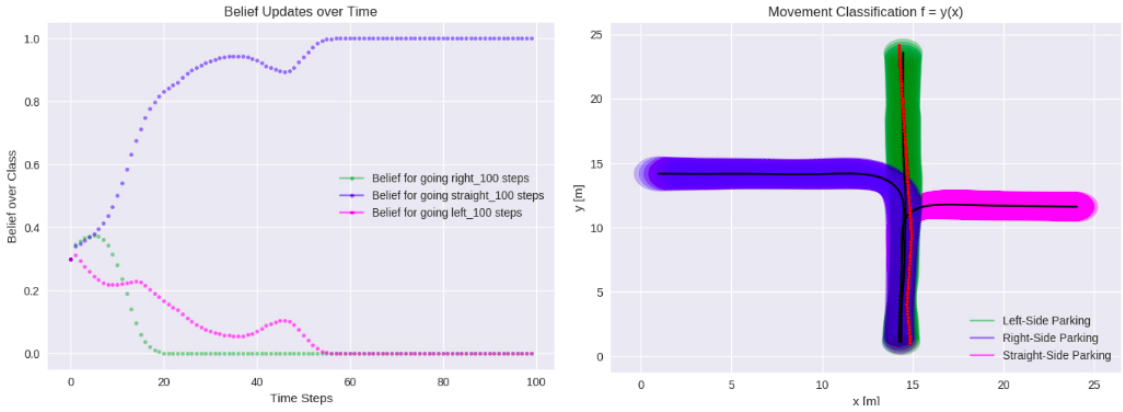


Figure 5.14.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is straight

5.4 Prediction Making in Online Method

in process

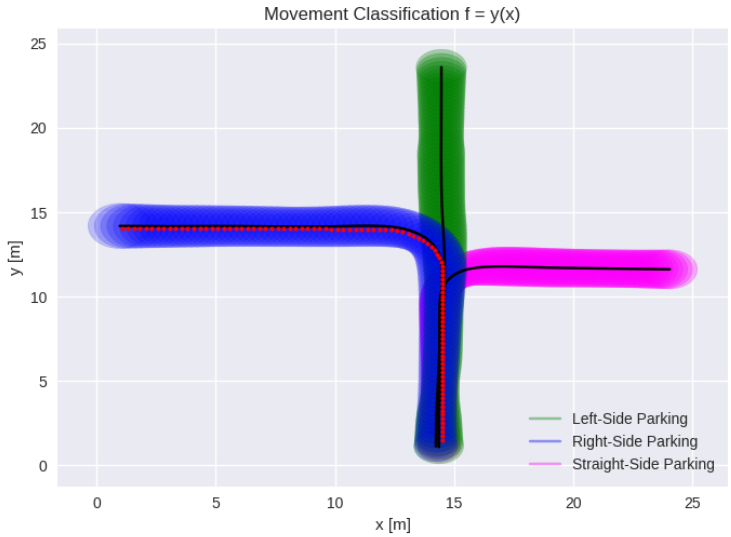


Figure 5.15.: Testing Trajectory (red) of going to the left

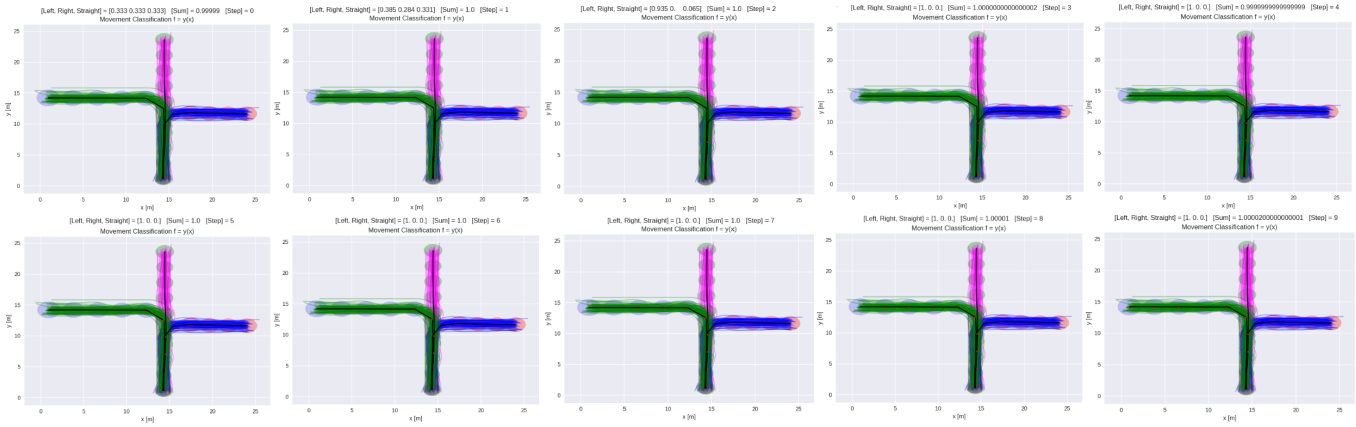


Figure 5.16.: Prediction making for trajectory which goes left. Trajectory has 10-time steps

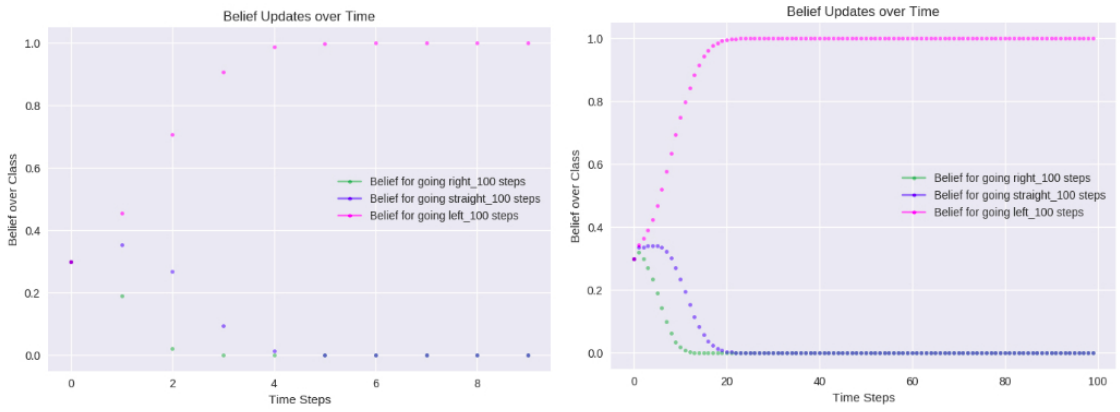


Figure 5.17.: Belief changes over time. For trajectory with 10 steps on the left, for trajectory with 100 steps on the right. Trajectory direction is left

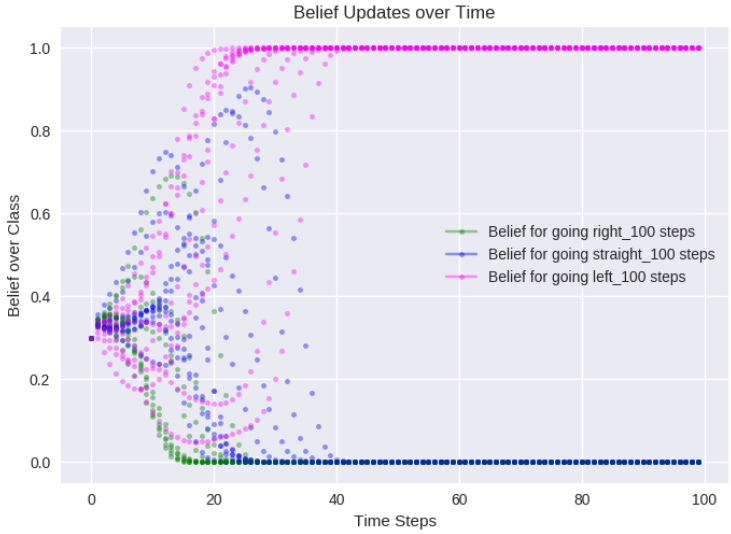


Figure 5.18.: Belief changes over time for different trajectories which belong to the same movement class. Trajectories have 100 time steps. Trajectory direction is left

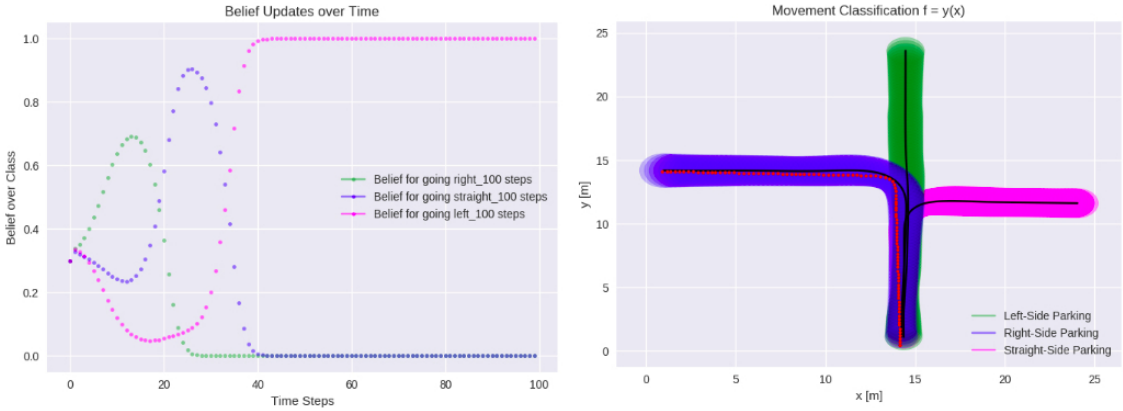


Figure 5.19.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is left

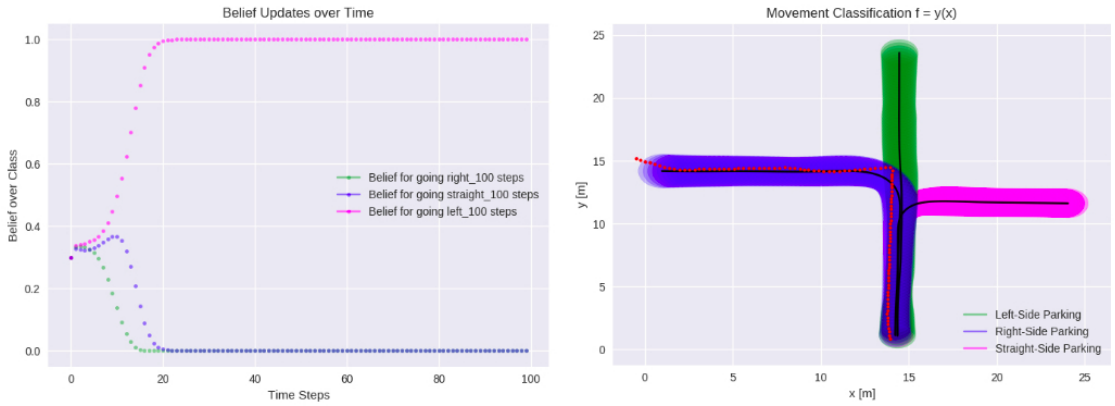


Figure 5.20.: Belief over time changing (left image) and image of testing trajectory (right image). Trajectories have 100 time steps. Trajectory direction is left

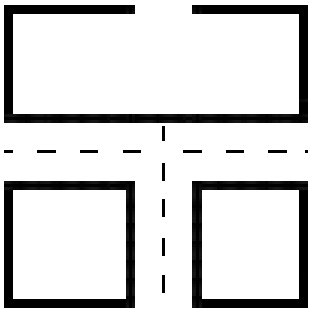


Figure 5.21.: T-intersection map

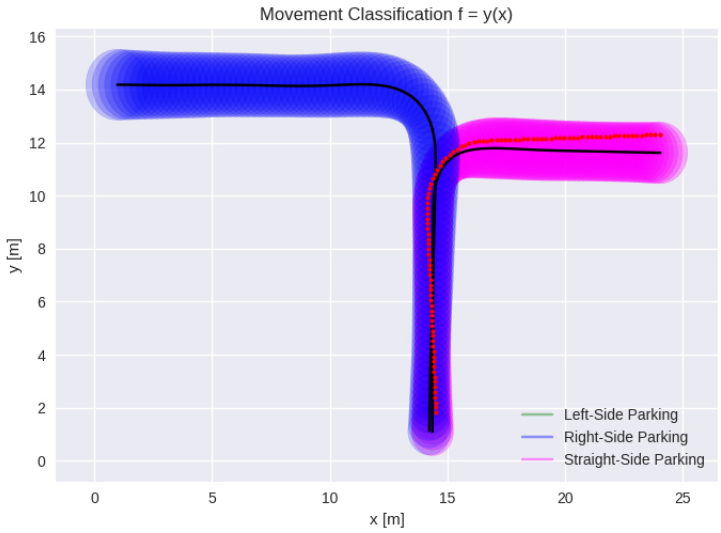


Figure 5.22.: Testing Trajectory (red) of going to right

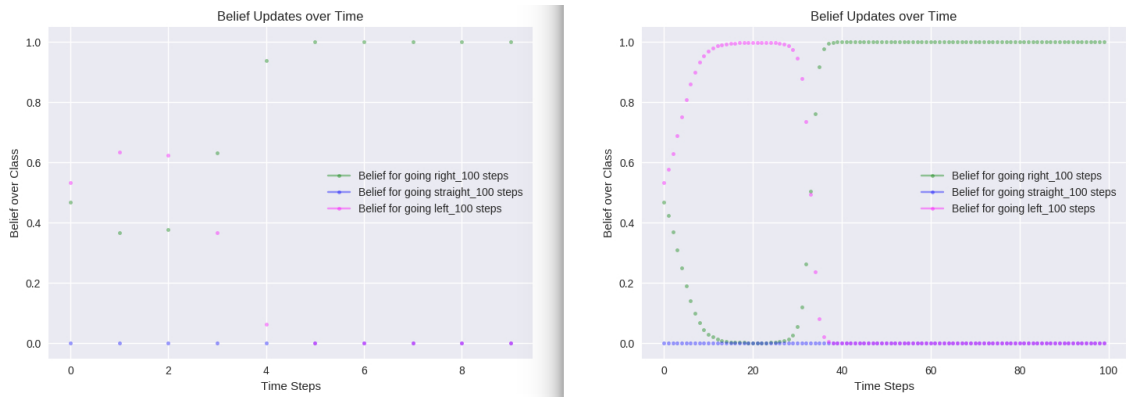


Figure 5.23.: Belief changes over time. For trajectory with 10 steps on the left, for trajectory with 100 steps on the right. Trajectory direction is right

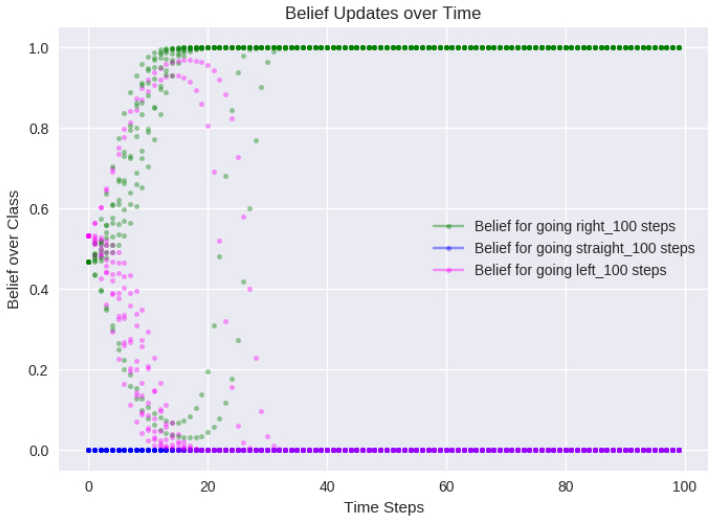


Figure 5.24.: Belief changes over time for different trajectories which belong to the same movement class. Trajectories have 100 time steps. Trajectory direction is right

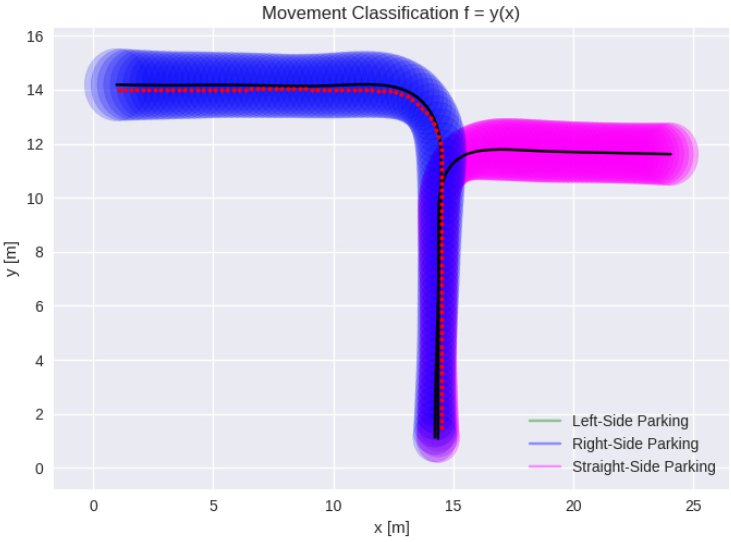


Figure 5.25.: Testing Trajectory (red) of going to the left

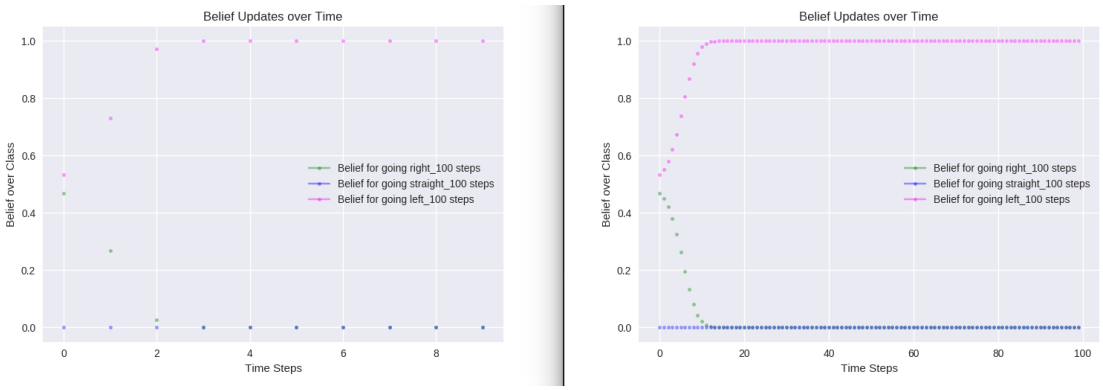


Figure 5.26.: Belief changes over time. For trajectory with 10 steps on the left, for trajectory with 100 steps on the right. Trajectory direction is left

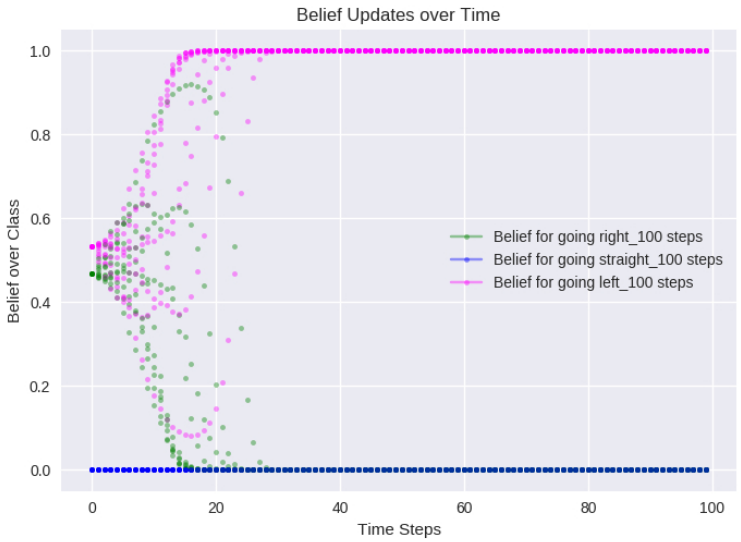


Figure 5.27.: Belief changes over time for different trajectories which belong to the same movement class. Trajectories have 100 time steps. Trajectory direction is left

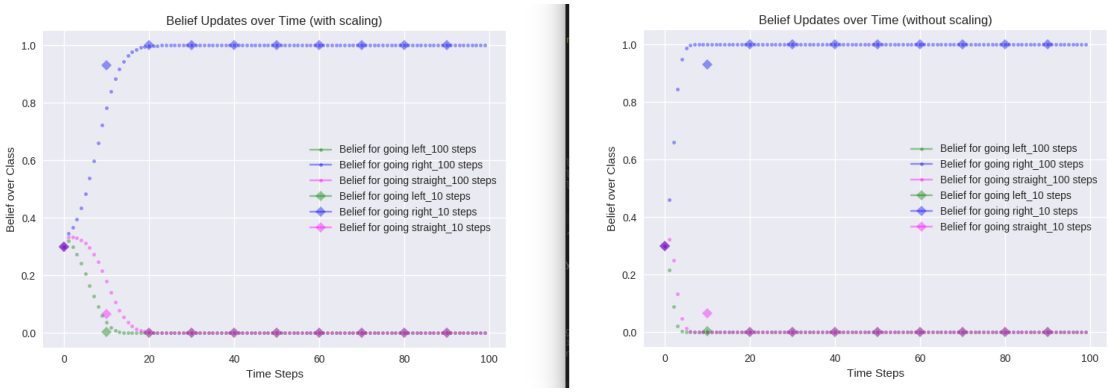


Figure 5.28.: Belief updates using scaling. Direction of the testing trajectory is right

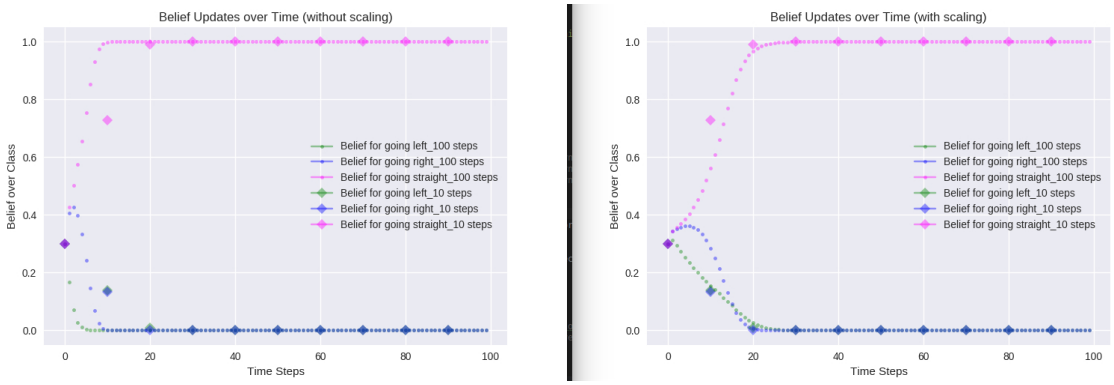


Figure 5.29.: Belief updates using scaling. Direction of the testing trajectory is straight

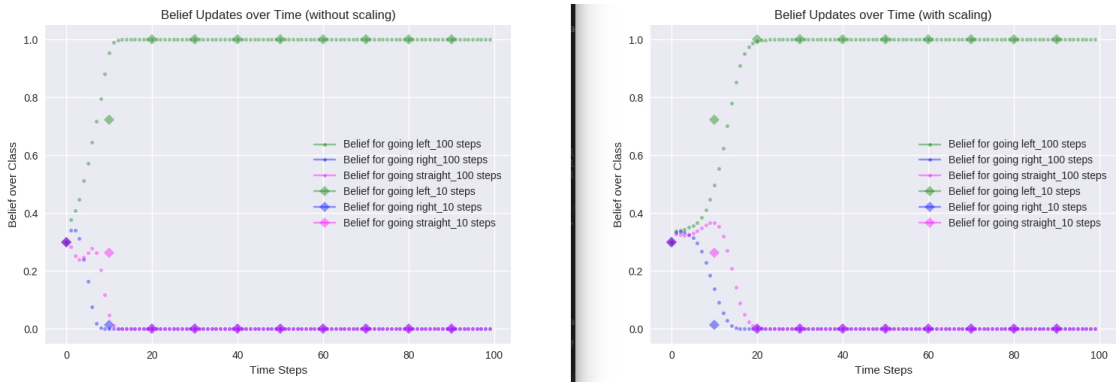


Figure 5.30.: Belief updates using scaling. Direction of the testing trajectory is left

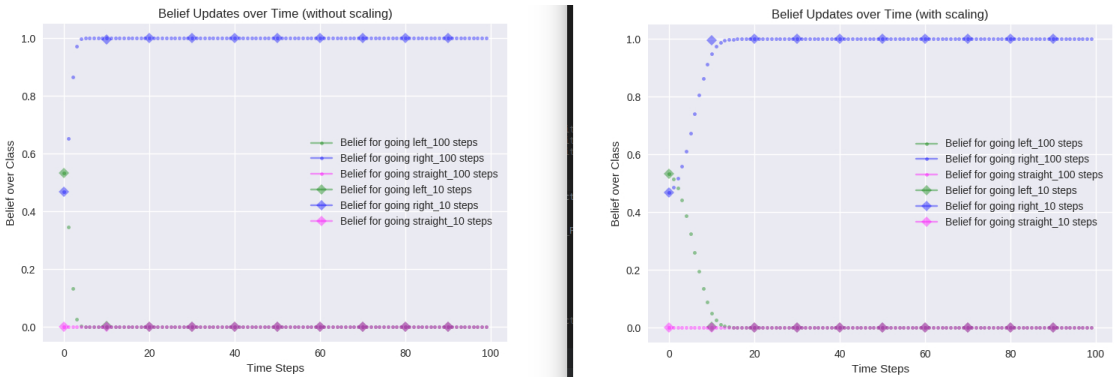


Figure 5.31.: Belief updates using scaling. Direction of the testing trajectory is right

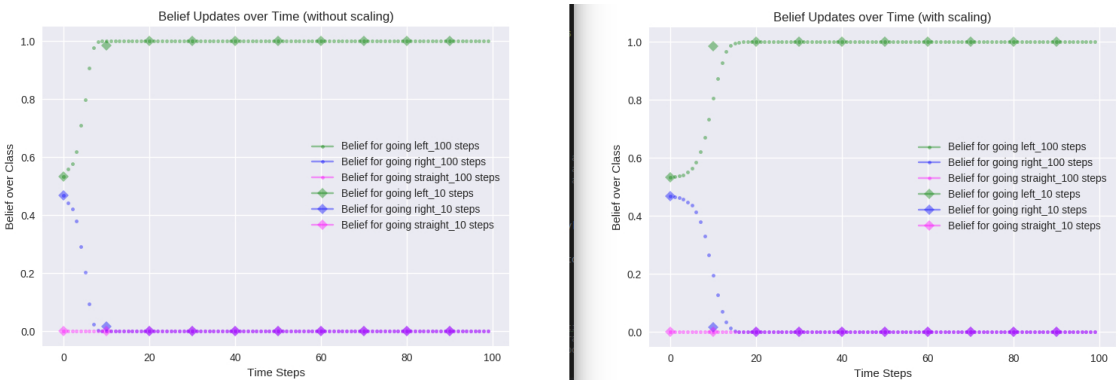


Figure 5.32.: Belief updates using scaling. Direction of the testing trajectory is left

6 Security Aspects

Fully autonomous cars are not quite ready to go into the roads yet, but autonomous vehicles are getting closer to reality more than ever before. More than 60 cities in the world have testing programs for autonomous cars or in preparation for it. Around 80$ billion is already invested in the technology, and almost every modern vehicle manufacturer has dedicated some resources for more and more automation in the newly launched car [??]. According to statistics [??] around 130k cars with partial automation are sold yearly and with current predictions until 2020 around 98k fully automated vehicles will be sold. Until 2040 this number is expected to be more than 96 million, which represents 95% of all vehicles sold.

Without a big novelty in technology, autonomous cars are promising enhanced safety and improved convenience, however, it still has a darker side. Autonomous cars essentially can be called Internet cars and being high-tech vehicles they have vulnerabilities and a lot of security issues. This chapter aim is to get to the bottom of potential security risks and defense strategies.

6.1 Background

As autonomous vehicles are getting closer to reality as private and public transit vehicles, it is natural to be interested in safety and security which these new technologies are promising. One report made by FBI brought out a number of security vulnerabilities and concerns related to autonomous vehicles, stating that autonomous technologies can become a potentially deadly weapon in the future [??]. But greater concern than terrorism attacks on autonomous vehicles are the risk of controls systems being hacked in and taking control of a driving or other essential systems and in this way to put into a dangerous situation driver or passengers of the car. Overtaking a control of the car can be used to consciously cause an accident or to drive a car into all new unplanned location and potentially steal a car using technologies. It also can allow lock passengers inside against their will. Or to track the car user and make a profile of the person, who is driving/using a car. In fact, autonomous cars technology are still quite new and it is still hard to see the full scope for security risks of autonomous vehicles.

This can sound too non-realistic, but white hat hackers for years already have been showing security issues related to connected cars, demonstrating how easy is to take control and do harm over a lot of various systems even in non-automated vehicles. All these vulnerabilities connected to the Internet are open to various kind of attacks. Even advanced autopilot system, designed by Tesla can be tricked quite easily. One of Chinese security company demonstrated that it is very easy to spoof sensors of a vehicle, causing them to sense "ghost" objects or fail to detect a real object at all [??].
Here natural question can arise, so what makes autonomous vehicles so vulnerable against malicious attacks as compared with non- or only partial autonomous cars? Authors of [4] suggesting two main reasons:

1. **The increased interaction between autonomous cars and environment.** At the moment the most communications between vehicles on the road occurs via Vehicular Ad Hoc Networks (VANETs). This type of communication allows sharing fast changing surrounding information with vehicles which are nearby communicating vehicle. This allows for other cars/drivers in the cars to be aware of what is the road situation nearby them [54]. Since technologies are improving and fully autonomous cars will be on the roads in the very near future, Vehicle to Infrastructure (V2I) and Vehicle to Internet of Things (V2IoT) communication will be more common on the roads. Due to connectivity in one network, only one "infected" vehicle can compromise the entire network if the network is not properly secured.

2. **The increased interaction between components of the system inside, so-called intra vehicular communication.** Autonomous cars have a lot of different Electronic Control Unitss (ECUs) which are connected with each other using Controller Area Network (CAN) bus. One of the biggest advantages of using CAN bus is that it is like a central unit into which a lot of different modules can be added or removed without changing the wiring architecture in the car. CAN has three main parts: **Data link layer**, which is responsible for data transferring. **High speed** physical layer and **Low speed** physical layer which is also responsible for fault toleration. In most cases, the most important control units (which has a direct impact on safety, e.g. brake or engine control module) are connected to high speed layer and others, not so security sensitive are connected to low speed layer. Not so common situation, but it is possible to have a "gateway bridge" which opens the route for selected packages from low to high (or vice versa) layers. So it is a possibility that that malicious packets are connected to the CAN bus using low speed layer

and without any further checking or suspicion it can be transferred into high speed layer causing to serious consequences. Control architecture in autonomous vehicles is working in this way what all nodes get packages from CAN. Any malicious component which is connected to the internal vehicle network can snoop all communications or it can infect all other elements. In order to protect the network, every path for package moving should be protected to ensure the vehicle security. Unfortunately, it is impossible to predict all possible attacks and to foresee all vulnerable places, because there always be new strategies which will threaten the security of autonomous cars. "The development and improvement of one will always counteract and necessitate the development of another" [4].

Another cause of CAN vulnerabilities is that all CAN packages are not authenticated before using them for communication within the system. Any element of a network can send infected element further if former did not do any validation of the package before accepting it [55]. One way to protect network infrastructure is to use packet-level authentication method. This method allows authenticating a package without having trust association of the package sender [56].

6.2 Attack Taxonomy

This section will introduce the potential threats, vulnerabilities and attacks of autonomous vehicles can face. For categorization of attacks, we use way proposed in [4]. Each attack is classified based on: **Type** (or source) of the attacker, **Attack vector** (path and method which was taken to get access to the vulnerable place), **Target**, **Reason/objective/motive** of the attack and **Potential outcome**.

- **Attacker** – a source of the attack. Usually, when system face an attack it tries to mitigate that immediately. One of the steps of mitigation should be an identification of attack source. Having this information it is possible not only to prevent future attacks, but also understand why the attack was implemented in the first place.
- **Attack Vector** – is a way how the attacker got access to the system he is targeting. It is also an enabler for an adversary to exploit the targeted system. Attack vectors can briefly be categorized to *physical* and *remote* access.

    1. **Physical Access**: these attacks are classified into invasive and non-invasive attacks.
        a) *Non-invasive Attacks*: these attacks have no physical contact with the car, usually, embedded devices are used for attack implementation. However, being relatively close to the targeted vehicle is necessary in order attack to work.
            i. Side-channel attacks: this attack usually ends with leakage of useful information about transmitted data within the system or internal working paths are mixed to work in alternative ways. An attack can leak information such as power consumption, timing information, signal analysis, etc. The most common defense against this attack is employing asynchronous information processing units or/and "shielding" mechanisms.
        b) *Invasive Attacks*: the main difference as compared to non-invasive attacks is that invasive attacks include a physical connection to the targeted system. The result of this attack can be the network security and ECUs can be compromised. Potential way how arversaries can connect to the car and gain access to its ECUs is On-Board Diagnostics-II (OBD-II) port which is usually used for car diagnostic. Another way to reach a car is wireless remote access, this is possible when an autonomous car is connected to the critical infrastructure, e.g. someone in the car connects a smartphone for entertainment reasons, and in this way, all internal system can be exposed to external people or networks. There are different types of invasive attacks which are discussed below.
            i. Code Modification:this attack may happen when attacker change the code with malicious modifications in order to compromise the system. This may be achieved by connecting OBD-II scanner to the car. As mentioned before this is a tool for vehicle diagnostic, meaning that it is widely available for anyone who wants to buy it. Advanced models of OBD-II has integrated feature for chip tunning which is used for extraction (and modification) of ECUs codes. This attack can be avoided by ensuring that all connection to the car is password protected, which enable only certified people to connect and modify codes of the car.
            ii. Code Injection: this is a very similar attack to the previous one. In this case, after connecting to the car, the attacker can inject new (most likely malicious) code instead of modifying the old one. Code injections can be made not only by an attacker, the owner of the vehicle or person who is checking

a car can also inject a new code, hoping to improve the performance of a vehicle. When injected codes are non-compliant with car's components or when new codes are not proved by authorities, problems can appear. One of the ways to avoid code injections is a usage of the intrusion detection system or/and usage of privileged access, which allows only authorized people to connect to the car, owner excluded.

iii. Packet Sniffing: also known as package analyzing. For this attack computer program or hardware, called sniffer (or analyzer) is used. A packet sniffer can see details between any communication node. Again, the tool is very useful when network related problems need to be diagnosed. But the tool can be used for malicious purposes as well: an attacker can use sniffers for collecting unprotected information, for eavesdropping or capturing packages for following replay attack. Defenses for this attack can be various encryption techniques for confidentiality in packages protection, as well as usage of real-time packages send/received update messages.

iv. Packet Fuzzing: this technique usually is used during security testing procedures, when invalid data is sent to the system/element, expecting to get receive some error or fault conditions, which allows exploiting weak places in the system and security loopholes. While testing the system, fuzzing helps to detect problems and utilize them in a further stage. When this technique is used by adversaries, the received information is used to get into the system and do any possible harm. Protection against fuzzing is to fix all errors and security loopholes immediately after its exposure. And system updates need to be verified and authenticated before presenting it to the public and uploading to the car.

v. In-Vehicle Spoofing: is a situation in which an attacker, using some software tools, pretending to be another person by falsifying data. In this way, adversary gains an illegitimate advantage. In order to attack be successful adversary need to overcome security mechanisms (if exists) to replace original elements with spoofing devices. Usual defenses for this is into autonomous car network include reply attack resistance techniques and fingerprinting module to be able to differentiate between original and current module in the vehicle system [57].

2. **Remote Access**: since wireless connection to external sencors, such as cameras, Light Detection and Ranging (LiDAR), Radio Detection and Ranging (RaDAR), Global Positioning System (GPS) are getting more and more popular, attackers can also use remote access as method to attack systems of autonomous vehicles.

   a) *External Signal Spoofing*: one example for this type of attack is GPS spoofing. This attack is possible because GPS using a wireless connection. During the attack GPS receiver is deceived by sending the wrong signal to GPS from another device. The incorrect signal may be similar to real GPS signal or can be captured before and replayed at a certain time. An attacker can trick GPS receiver to accept and recognize only fake signal by gradually increasing power strength of the wrong signal until it eventually replaces the original signal. As soon as the attacker gains control of GPS device in an autonomous car, he can send false GPS information and lead car in the wrong direction and destination. This attack was not tested on autonomous cars, but it was successfully tested on GPS devices in Unmanned Aerial Vehicle (UAV) and yachts [??, ??]. GPS devices are not the only target in autonomous cars. Vehicles contain numerous sensors, which can be attacked with spoofing. One of these devices could be visual sensors like cameras, LiDARs. Similar to the previously described attack was made on LiDAR sensor in [??]. Sending fake signals to the device in a range between 20 and 250 meters from a car (sensor) a lot of non-existing obstacles can be detected as well as existing obstacles on the road can be missed. The way to protect the system against spoofing attacks is to ensure more security, not to accept any signals and information without checking the authenticity and integrity of a signal sending device. To ensure that correct information is coming to LiDAR additional information sources can be used and perform information checking between two different devices. If this cross-checks matches and succeeds information is more likely to be correct than in a case when information is not matching between different sources.

   b) *Jamming*: these attacks are against wireless or external sensors for vision and due to that, the authorized communication might be destroyed. The most sensitive devices for jamming attacks are LiDAR, RaDAR, various cameras. Jamming devices can block sensors for receiving correct data. Authors of [??] used jamming to blind cameras of autonomous vehicles to hide objects on the road and make map "cleaner" as it is. There are ways to protect sensors against this attack using removable near infrared-cut filter to the camera, however, this method is working only in the day time. Another measure is photo-chromic cameras' lenses, which can filter out specific types of light.

- **Attack Target** – the target component of the car is usually depends on motive/object of attack and attacker's intention. If the attacker wants to track a path which car is moving, the target element can very likely be camera and/or RaDAR or LiDAR , since they are vision elements of the car and having information from these sensors it is easy to see and follow the path, car took. If attacker targeting traffic optimization and/or passengers safety, as a target VANET can be used, ect.

- **Attack Motive** – various motives for attacks are possible, better we understand it, it is more likely to protect all control systems and ensure safe driving. One of possible reasons for attacking can be deception - this is a spreading a false information, hoping to effect behavior of other, this attack may lead to hazardous situations on the traffic/roads. One of motives can be economical - to get a financial gain. The another very serious motive, due the any reasons, can be to cause severe harm to passengers of the car, etc.

- **(Potential) Consequences** – various consensuses are possible if any of these attacks against autonomous vehicles will be successful, such as some (important) control functions might to fail or be sabotaged, data leakage (e.g. vehicle movement information) is very possible outcome [12??]. The "health" of the system is very likely will be affected after successful attacks, which can lead to passengers of the car health issues.

Figure 6.1 summaries attack taxonomy described earlier in this section.

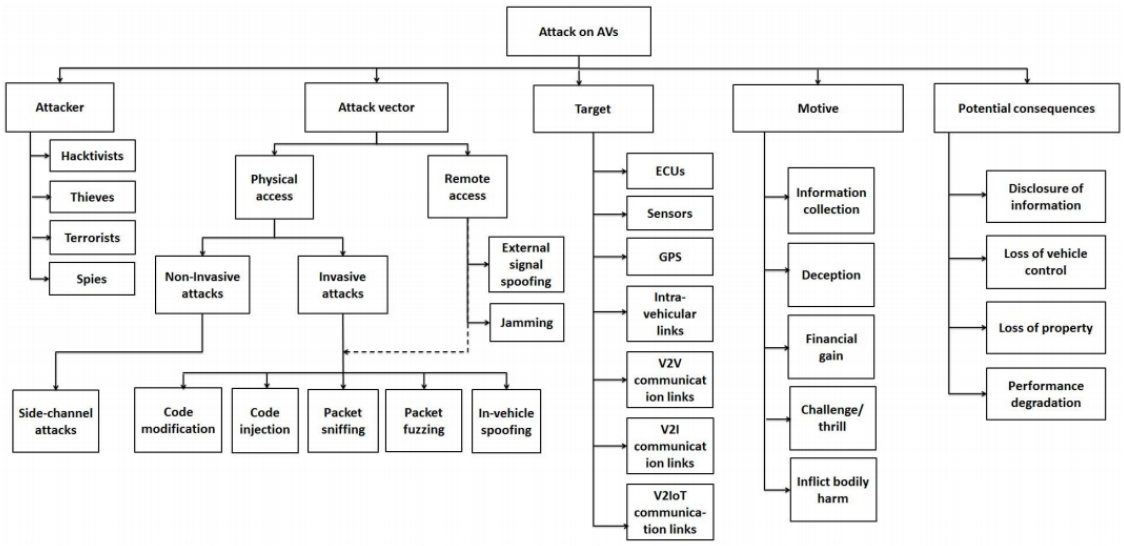


Figure 6.1.: Autonomous Vehicle Attack Taxonomy [4]

Another important problem which must to me mentioned while talking about safety and security issues on autonomous cars and is not mentioned in attack taxonomy yet is **privacy**. An attacker can arrange an attack where he follows car movement, times and makes a very detailed profile about car owner and with this information can do various things: from robbing the house of the victim while he is not at home, to selling this information to someone else, combine different attacks to do the biggest damage attacker want (or is able to) arrange.

6.3 Defense against Attacks Taxonomy

As mentioned earlier it is very hard to predict what is going to happen to prevent yourself against attacks. However, with current knowledge about system and known potential vulnerabilities, it is possible to make some kind of predictions and develop network architectures and working protocols which are not so vulnerable. Various literature surveys propose the main 4 types of defenses for autonomous vehicles: **Preventive**, **Passive**, **Active** and **Collaborative** defenses.

This classification and different ways of protection ensure that the system is secure and resilient for different attacks.

1. **Preventive Defense**: this type of defense mechanisms mainly focuses on protecting a system from attack before it starts or finishes with success. Preventive defense is mainly focusing on normal working conditions and not during the attack and does not solve any issues with "after attack" scenarios.

a) *Secure Communication*: for secure communication in any case, from simple chatting to serious control commands, data encryption is basic and crucial. Using encryption content and confidentiality of messages are assured. Encryption scheme can also help with the identification of sender/received of sent data. If the encryption scheme does not include identification, integrity and identification of sender/receiver can be assured, using Message Authentication Code (MAC) algorithms. To know the integrity and authentication of another side of communication is very important for secure communication.

b) *In-Vehicle Device Authentication*: controllers used in the car can be completely trusted if they have a manufacturer certificate which gives information such us controller identifier, public key, information about authorized carryout, etc. If information, provided in certificate matches with information which car itself contains, then authentication process is successful and car safely can use all information coming from that particular controller.

c) *User Authentication*: sometimes, in order to have more protection, user authentication is used. To make sure that the right person has access to the car (e.g. doors opening, starting the car, etc.) additional biometric information can be used.

d) *Firewall*: it is an additional tool, which always can be used. Firewalls check all incoming and outgoing data traffic based on rules which user/authenticated people can define. Firewalls also can be very helpful in communication with a trusted and not-trusted environment: this is very important while communicating with different objects in vehicles' networks.

2. **Passive Defense**: this type of defense is similar to earlier described preventive defense. Passive defense measures are taken to minimize damage caused by an attacker without having the intention of taking initiative. A passive defense can be an additional level of protection (not the main one). As compared to active defense, the passive defense does not require any analysis from human.

   a) *Attack Detection*:

      i. Intrusion Detection: To detect physical threats to the car is much easier than to see attacks against system operations. However, there are various models for Intrusion Detection System (IDS) which can be used for autonomous vehicles. Authors of [??, ??] proposed and tested IDS models using various computational simulation scenarios. Even though there are models which have quite good results, research and development of IDS should not stop in order to achieve higher accuracy in attack detections.

      ii. Anti-Malware: these solutions are used in all usual computer network (or only computer) systems. Anti-malware systems need to be able to protect from harmful attempts to penetrate into the main system. Since malware for autonomous cars is a relatively new thing, it might be not possible to find numerous malware "available", but they still should be taken into account. Research communities, who specialize in autonomous cars, trying to predict possible attack models containing new malware to be prepared for these attacks.

   b) *Attack Response*:

      i. Nullification: when an attack is recognized by system nullification can be used. This defense mechanism can neutralize an attack using cyber/electronic capabilities, e.g. GPS signal anti-jamming technologies [??]. These technologies suppressing signal from malicious jamming devices.

      ii. Isolation: it helps vehicles to isolate themselves from other cars during an attack. Self-isolation also prevents ECUs re-programming while the car is running. Self-isolation should happen in a few levels: the autonomous vehicle network system should isolate itself from other vehicles and the affected layer should be isolated from other levels in the same networking system in order not to affect the healthy vehicle behaviour. When a car is attacked ideally it not only should isolate itself but also inform vehicles around about attack in order other cars could take some actions to defend itself against attack.

   c) *Attack Recovery*:

      i. Availability: this feature one of the most important in all types of systems. In the context of autonomous cars, availability is very important when talking about safety inside and outside of the vehicle. In order to ensure safety and have good fault toleration within the system in autonomous cars and to ensure quick recovery after attacks, availability must be ensured in the system.

3. **Active Defense**: is one of the advanced and determined defense techniques. Different approached described below.

a) *Continuous Security Monitoring*: autonomous vehicles belong to critical infrastructure and it is essential that security of these systems should not be compromised. It must have (near) real-time solutions for checking and/or restoring healthy driving conditions in the car. Non-stop and continuous monitoring and "snapshots" of all running systems are required to be available for security checking at any time. It is also important to ensure that all critical devices and interfaces are not blindsided at any time while the car engine is running.

b) *Adaptive Security*: Nowadays the most critical systems are fast changing or refer to infrastructure with a fast pace, hence old and static defense mechanisms are not sufficient anymore. It is necessary to model and use defense mechanisms which are dynamic. So-called adaptive reconfigurations for target can be used for ensuring better control and balance during attack. In order to prepare it self for future attacks defense mechanisms should be able to analyze past and current attacks and use self0learning mechanisms to predict what may happen in the future and adapt defense mechanisms for new forms of attack in the future.

4. **Collaborative Defense**:

a) *Cloud Computing*: As mentioned above, if cars in the same network over the VANET will share information about potential threats, it is possible that overall security will be improved. When all communication and communication history will be transferred to the clouds, it will become one of the main targets for attackers. Since it will be a "hot spot" for adversaries, security specialist must investigate infrastructure very well, which will give more knowledge and information about potential threats and will let to develop an adaptive defense for better protection of autonomous cars and critical infrastructure.

This section provided a brief summary of vulnerabilities, attacks and defense mechanism in term of security of autonomous vehicles. One of the most important things n security is not to stop looking for potential vulnerabilities and ways to protect the car and all network in case of attack and/or to recover as fast as possible if an attack happened. Regardless of the effectiveness of current methods, it is necessary to think about new defense mechanisms for real-time operations of the vehicle.
Figure 6.2 summaries defense against attacks taxonomy described earlier in this section.

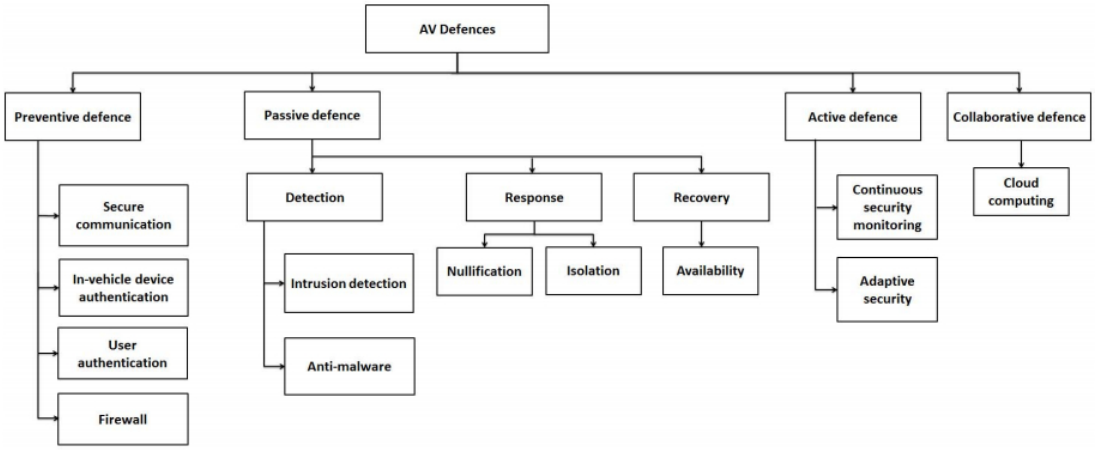


Figure 6.2.: Autonomous Vehicle Defense Taxonomy [4]

6.4 The most important attacks

As mentioned before there are different types of attacks and various ways how to use them. Further in this section will be discussed the most common and dangerous security attacks while making movement prediction:

- **Visual sensors attacks**. It is important to make sure that the map of car path is correct and there are no hidden or not real obstacles on the road. (Thinking to write more about External Signal Spoofing on LiDAR ???) Another serious attack on visual sensors can be used for a wrong understanding of traffic signs. How easy is to "confuse" STOP sign with Giving Way on Oncoming Traffic sign or to confuse 90km/h with 50km/h, etc.

- **Privacy issue**. Privacy protection might look not important and not relevant while talking about autonomous cars, but it is. (Want to give more insights on how attacks can be arranged and used ???)

7 Conclusions and Future Works

TODO

Bibliography

[1] W. Z. Yeping Hu and M. Tomizuka, "Probabilistic Prediction of Vehicle Semantic Intention and Motion," 2018.

[2] A. Singh, "PREDICTION in Autonomous Vehicle - All You Need To Know."

[3] C. L. Stéphanie Lefèvre, Dizan Vasquez, "A survey on motion prediction and risk assessment for intelligent vehicles," *ROBOMECH Journal*, 2014.

[4] V. L. Thing and J. Wu, "Autonomous Vehicle Security: A Taxonomy of Attacks and Defences," 2016.

[5] A. F. for Traffic Safety., "American Driving Survey, 2015 – 2016."

[6] WHO, "Global Status Report On Road Safety 2018."

[7] S. R. Kaiming He, Xiangyu Zhang and J. Sun, "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification," 2015.

[8] T. Keeney, "Mobility-As-A-Servive: Why Self-Driving Cars Could Change Everything," 2017.

[9] TuD, "aDDa for Students."

[10] G. B. Ismail Dagli, Michael Brost, "Action recognition and prediction for driver assistance systems using dynamic belief networks," p. 179–194, 2002.

[11] K. T. Shigeki Tezuka, Hitoshi Soma, "A study of driver behavior inference model at time of lane change using bayesian networks," p. 2308–2313, 2006.

[12] A. P. Andrew Liu, "Towards real-time recognition of driver intentions," p. 236–241, 1997.

[13] A. P. Andrew Liu, "Modeling and prediction of human behavior," p. 229–242, 1999.

[14] A. P. Nuria Oliver, "Driver behavior recognition and prediction in a smartcar,," p. 280–290, 2000.

[15] A. P. Nuria Oliver, "Graphical models for driver behavior recognition in a smartcar," p. 7–12, 2000.

[16] D. D. Salvucci, "Inferring driver intent: A case study in lane-change detection," p. 2228–2231, 2004.

[17] C. S. Yi Hou, Praveen Edara, "A genetic fuzzy system for modeling mandatory lane changing," 2012.

[18] D. D. S. Hiren M. Mandalia, "Using support vector machines for lane change detection," 2005.

[19] M. J. Kochenderfer, "Decision Making Under Uncertainty: Theory and Application," p. 11–57, 2015.

[20] A. R. C. Leslie Pack Kaelbling, Michael L. Littman, "Planning and acting in partially observable stochastic domains," p. 99–134, 1998.

[21] A. R. Nachiket Deo and M. M. Trived, "How Would Surround Vehicles Move? A Unified Framework for Maneuver Classification and Motion Prediction," 2018.

[22] J. S. Mattias Brannstrom, Erik Coelingh, "Model-Based Threat Assessment for Avoiding Arbitrary Vehicle Collisions," p. 658–669, 2010.

[23] K. K. Jrg Hillenbrand, Andreas M. Spieker, "A multilevel collision mitigation approach: situation assessment, decision making, and performance tradeoffs," p. 528–540, 2006.

[24] A. J. A. L. A. Aris Polychronopoulos, Manolis Tsogas, "Sensor fusion for predicting vehicles' path for collision avoidance systems," p. 549–562, 2007.

[25] F. N. Samer Ammoun, "Real time trajectory prediction for collision risk estimation between vehicles ," p. 417–422, 2009.

[26] M. S. Nico Kaempchen, Kilian Weiß, “IMM object tracking for high dynamic driving maneuvers,” p. 825–830, 2004.

[27] J. B. Thomas Batz, Kym Watson, “Recognition of dangerous situations within a cooperative group of vehicles,” p. 907–912, 2009.

[28] K. P. Murphy, “Dynamic Bayesian networks: representation, inference and learning,” 2009.

[29] S. B. Adrian Broadhurst and T. Kanade, “Monte Carlo road safety reasoning,” p. 319–324, 2005.

[30] A. M. Matthias Althoff, “Comparison of Markov chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars.,” p. 1237–1247, 2011.

[31] D. from Wordreference, “Definition of maneuver.”

[32] S. B. Tobias Gindele and R. Dillmann, “A probabilistic model for estimating driver behaviors and vehicle trajectories in traffic environments ,” p. 1625–1631, 2010.

[33] D. R. Ismail Dagli, “Motivation-based approach to behavior prediction,” p. 227–233, 2002.

[34] L. H. S. J. P. H. Georges S. Aoude, Vishnu R. Desaraju, “Driver behavior classification at intersections and validation on large naturalistic dataset,” p. 724–736, 2012.

[35] C. Laugier *et al.*, “Probabilistic analysis of dynamic scenes and collision risks assessment to improve driving safety,” p. 4–19, 2011.

[36] A. W. G. B. J. Schlechtriemen, F. Wirthmueller and K.-D. Kuhnert, “When will it change the lane? A probabilistic regression approach for rarely occurring events,” p. 1373–1379.

[37] G. B. J. Schlechtriemen, A. Wedel and K.-D. Kuhnert, “A probabilistic long term prediction approach for highway scenarios,” p. 732–738, 2014.

[38] V. C. Y. W. Adam Houenou, Philippe Bonnifait, “Vehicle Trajectory Prediction based on Motion Model and Maneuver Recognition,” p. 4363–4369, 2013.

[39] K. K. H. L. D. S. L. J. P. H. Georges S. Aoude, Brandon D. Luders, “Threat assessment design for driver assistance system at intersections,” p. 1855–1862, 2010.

[40] M. M. T. Brendan Tran Morris, “Trajectory learning for activity understanding: Unsupervised, multilevel, and long-term adaptive approach,” p. 2287–2301, 2011.

[41] K. D. Holger Berndt, Jorg Emmert, “Continuous driver intention recognition with hidden Markov models,” p. 1189–1194, 2008.

[42] M. M. T. Aida Khosroshahi, Eshed Ohn Bar, “Surround vehicles trajectory analysis with recurrent neural networks,” p. 2267–2272, 2016.

[43] J. A. Matthias Schreier, Volker Willert, “Bayesian, maneuver-based, longterm trajectory prediction and criticality assessment for driver assistance systems,” p. 334–341, 2014.

[44] J. F. Quan Tran, “Online maneuver recognition and multimodal trajectory prediction for intersection assistance using non-parametric regression,” p. 918–923, 2014.

[45] C. W. H. R. E. Kfer, C. Hermes and F. Kummert, “Recognition of situation classes at road intersections,” pp. 3960–3965, 2010.

[46] C. F. P. G. T. D. W. A. Lawitzky, D. Althoff and M. Buss, “Interactive scene prediction for automotive applications,” pp. 1028–1033, 2013.

[47] K. S. C. Hermes, C. Wohler and F. Kummert, “Long-term vehicle motion prediction,” p. 652–657, 2009.

[48] D. Vasquez and T. Fraichard, “Motion prediction for moving objects: a statistical approach,” p. 3931–3936, 2004.

[49] F. D.-V. J. M. Joseph and N. Roy, “A Bayesian nonparametric approach to modeling mobility patterns,” p. 1587–1593, 2010.

[50] U. K. J. Wiest, M. Hffken and K. Dietmayer, “Probabilistic trajectory prediction with Gaussian mixture models,” p. 141–146, 2012.

[51] N. R. J. P. H. Georges S. Aoude, Joshua Joseph, “Mobile agent trajectory prediction using Bayesian nonparametric reachability trees,” p. 1587–1593, 2011.

[52] ROS, “ROS joy package summary.”

[53] S. Scholar, “Toy Problem.”

[54] N. A. S. G. D. K. Swarun Kumar, Lixin Shi and D. Rus, “CarSpeak: A Content-Centric Network for Autonomous Driving,” 2012.

[55] F. R. S. P. Karl Koscher, Alexei Czeskis and T. Kohno, “Experimental Security Analysis of a Modern Automobile,” 2010.

[56] D. Lagutin, “Packet Level Authentication Overview,” 2010.

[57] K.-T. Cho and K. G. Shin, “Fingerprinting electronic control units for vehicle intrusion detection,” 2016.

[58] A. M. Matthias Althoff, “Comparison of Markov chain abstraction and Monte Carlo simulation for the safety assessment of autonomous cars,” p. 1237–1247, 2011.

A Some Appendix

Use letters instead of numbers for the chapters.