POLITECNICO DI MILANO
Corso di Laurea MAGISTRALE in Ingegneria dell'Automazione
Dipartimento di Elettronica, Informazione e Bioingegneria

Analysis and comparison of driver intent identification methods

Relatore: Dr. Alessandro Colombo
Correlatore: Dr. Gabriel Rodrigues de Campos

Tesi di Laurea di:
Stefano Baldan, matricola 804487

Anno Accademico 2014-2015

Sommario

Questa tesi presenta il lavoro di identificazione e analisi di metodi per la predizione del comportamento di un guidatore in un incrocio stradale. In particolare, vengono proposti i principali metodi di predizione trovati in letteratura, ovvero: Bayesian Approach, Hidden Markov Model e Interacting Multiple Model. Per ogni metodo sono presentate le caratteristiche, le equazioni e gli algoritmi che calcolano le probabilità degli intenti del guidatore, che per semplicità di calcolo si limitano a due: curvare o andare dritto. Per avere un'analisi più precisa, vengono testati gli algoritmi su diverse traiettorie in modo da confrontare i metodi sui risultati ottenuti, in particolare sul tempo di predizione e la corrispettiva probabilità. In una prima fase, le traiettorie utilizzate sono simulate da un generatore di traiettorie mentre, successivamente, sono ricavate da un dispositivo GPS, che misura la dinamica di un veicolo reale. Infine vengono presentate le analisi di efficienza, con curve di sensitività e specificità che descrivono in modo definitivo la qualità dei metodi affrontati.

Abstract

This thesis describes and analyses methods for the driver behaviours prediction in an urban intersection. The principal prediction methods found in literature are considered: Bayesian Approach, Hidden Markov Model and Interacting Multiple Model. For each method the features, equations and algorithms are discussed. The algorithms are tested on simulated and real trajectories, comparing the methods' results in terms of prediction horizon for a set intent probability. On the real trajectories a sensitivity and specificity analysis is also performed.

Contents

List of Figures

Chapter 1

Introduction

Looking at the latest statistics from ISTAT, the number of vehicle collisions in Milan between 2001 and 2014 are equal to 531978, of which 51% occurred in an crossroad. Moreover, 34% of the deaths and 52% of injuries occurred in an urban intersection. Despite the technological advances, it is commonly believed that humans will keep driving cars for the next years to come.
Systems trying to prevent (or limit) the effects of the collisions can be divided in two groups: passive and active security systems. Passive systems are protections that prevent a possible injury for the passengers of the vehicle upon an accident, such as the safety belt and the airbag. The active systems, instead, are designed to reduce the probability of a accident. As example, we can mention the Antilock Braking System (ABS) that prevents the blocking of wheels upon an hard braking, the Electronic Stability Program (EPS) controlling the dynamic stability and the Traction Control System (TCS) that restores traction if the wheels start to spin.
Within the Advanced Driver Assistant Systems (ADAS), one of the most innovative areas concerns the design of methods that are able to predict the actions of the driver and prevent, if and only if strictly necessary, the collision. The reader can refer to [2] and [3] for further information on these approaches.
In this works, the driver behaviour is assumed to be known. Therefore, an important improvement to these works is the prediction of the driver intent before the vehicle reaches the intersection, so that the active safety system has more time available to determine if an intervention is necessary to prevent a collision early in time.
This thesis aims to analyse the most important methods available in literature, which have been applied to intent estimation.
The thesis is organized as follows. Chapter 2 introduces the different meth-

ods' theoretical background, equations and algorithms used to predict the driver intent. Chapter 3 provides simulation results and performance analysis, while Chapter 4 includes experimental results considering real, measured trajectories of full scale vehicle. Finally, Chapter 5 presents some concluding remarks and perspectives for future research.

Chapter 2

Theoretical background

2.1 Aim of the project

Considering the scenario illustrated in Figure 2.1, the goal is to predict as soon as possible the driver intent using only the distance D to the crossroad and the variables describing the trajectory of the car: the longitudinal position s, i.e., the distance between the vehicle and its starting point along the longitudinal path of the car, the longitudinal speed v and the acceleration a. Information such as the steering angle or the relative distance between two cars will be disregarded in this work. The main methods of identification that can be applied to this problem are:

- Bayesian approach
- Hidden Markov Model
- Interacting Multiple Model

Despite the extensive state-of-art on collision avoidance methods, there doesn't exist a vast literature regarding our specific problem, and the advantages and disadvantages of each method are yet not clear. In this thesis our aim is to compare all the methods in order to identify the strong and week points of each method.

2.2 Preliminary definitions

Before focusing in the study of the methods, we want specify some definition. To keep the problem simple, we consider a scenario with a single vehicle with only two options:

- Turn
- Go straight

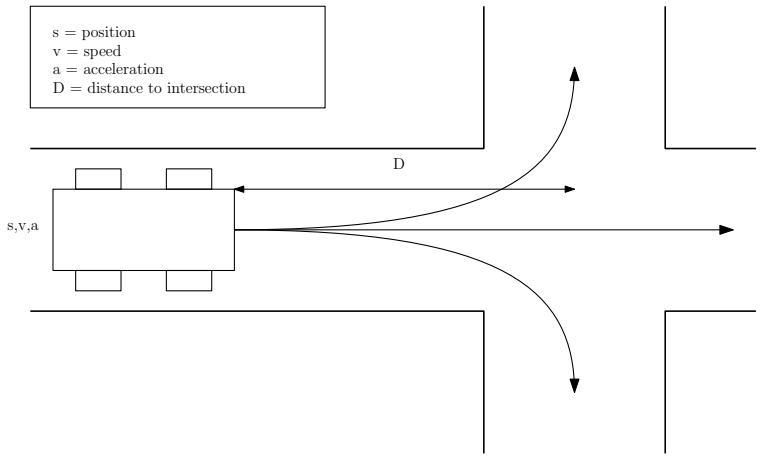


Figure 2.1: Typical scenario of the vehicle

We call them the possible *intents* of the driver. Note that the intents are only the high level description of the behaviour of the driver. The same intent may correspond to different trajectories of the car. For instance, a driver intending to go straight may keep a constant speed or slow down at the junction.
We call *hypotheses* a finite set of representative trajectories corresponding to an intent (Figure 2.2). An illustration of the system as considered in this work is presented in Figure 2.3: the vehicle trajectory (generated by simulation or measured in a real vehicle), is analysed by a classifier (based in the previously mentioned methods), that computes the probability of the driver intent.

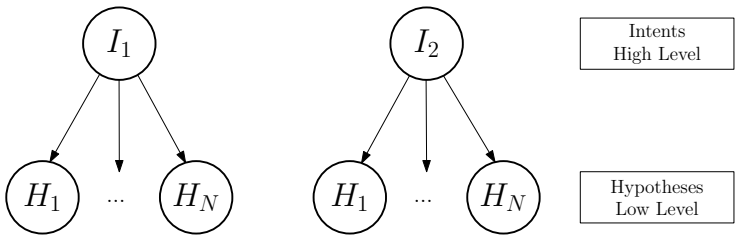


Figure 2.2: Classification of a driver behaviour

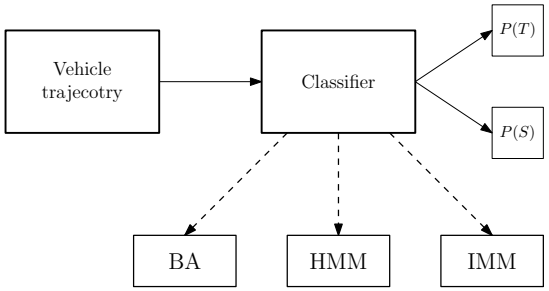


Figure 2.3: Illustration of the classification process

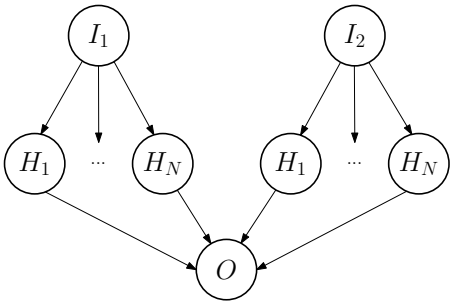


Figure 2.4: Bayesian network with the two intents I and the sets of hypotheses H that refer to the single observation O

2.3 Bayesian Approach

Our analysis of Bayesian Approach method is based on the results of [4] and [5], of Christoph Stiller and his team. We choose this work because, to the best of our knowledge, is the best method that uses the Bayesian approach showing good results in terms of prediction horizon and probability levels.
We consider several trajectories of the vehicle in different intersections. For the two possible intents I, we choose a set of hypotheses H describing the most common trajectories seen in the crossroad.
These hypotheses H, that can be seen as mean trajectories, will be compared with an observation O. This observation can be a suitable function of the measured trajectory. In particular, we will compare the values of position s, speed v and acceleration a.
The probability distribution for the driver intents I, the applicable hypotheses H and the defined observation O can be modelled by a simple Bayes net, illustrated in Figure 2.4. The probability for a particular intent I_j given the

observation O can be written as:

$$P(I_j|O) = \sum_i P(I_j|H_i)P(H_i|O) \tag{2.1}$$

where j indicates the j-th intent, i the i-th hypotheses and $P(I_j|H_i)$ is either 0 or 1 depending on the hypothesis. In fact, each mean trajectory chosen as hypothesis H_i is uniquely assigned to a single intent I_j.
The probabilities for the individual hypotheses are given by:

$$P(H_i|O) = \frac{P(O|H_i)P(H_i)}{\sum_j P(O|H_j)P(H_j)} \tag{2.2}$$

where the prior probability is equal to $P(H_i) = 0.5$ and the value $P(O|H_i)$, that we are interested in computing, represents a particular "distance" between the current observation O and the hypothesis H_i. There are different approaches that allow us to compute $P(O|H_i)$ but the most effective ones, considered throughout of the manuscript, are the following:

- Simulation Based Approach
- Comparison Based Approach

2.3.1 Simulation Based Approach

A possible way to estimate the distance $P(O|H_i)$ at time t is to create a virtual vehicle which motion is represented as:

$$\begin{cases} \ddot{\hat{s}} &= f(\hat{s}, \dot{\hat{s}}, u), \\ \dot{\hat{s}} &= \hat{v}, \end{cases} \tag{2.3}$$

where $\hat{s}$ and $\hat{v}$ are simulated values for position and speed, respectively. We give as input u the acceleration of hypothesis H_i, such that the virtual vehicle simulates the longitudinal behaviour of the car for the time interval $[t - T_S, t]$ with starting values

$$\hat{s_i}(t - T_S) := s(t - T_S) \qquad \text{and} \qquad \hat{v_i}(t - T_S) := v(t - T_S).$$

Here, T_S is the simulation time and $s(t - Ts)$, $v(t - Ts)$ are the observation of position and speed of the real vehicle at time $t - Ts$.
One possible way to compute $P(O|H_i)$ is to compare the final values of the simulation $\hat{s}(t)$ and $\hat{v}(t)$ and the actual values $s(t)$ and $v(t)$.

Assuming a normal distributed noise for both $s(t)$ and $v(t)$, the distance for hypothesis H_i is given by

$$P(O|H_i) = \frac{1}{2\pi\sigma_s\sigma_v} exp\left(-\frac{1}{2}e^2\right), \tag{2.4}$$

with

$$e := \sqrt{\left(\frac{s(t) - \hat{s}(t)}{\sigma_s}\right)^2 + \left(\frac{v(t) - \hat{v}(t)}{\sigma_v}\right)^2}, \tag{2.5}$$

where σ_s and σ_v are the standard deviation for the position and speed. Note that using appropriate values for σ_s and σ_v is important to choose appropriately the values of e, as they have a major influence on how easily a hypothesis will be favoured above others. In our case, we choose the two standard deviations testing the preliminary trajectories and looking how much the speed and position parameters change their values.

2.3.2 Comparison Based Approach

As an alternative to the Simulation Based Approach, the distance $P(O|H_i)$ can be estimated by comparing the actual acceleration a of the vehicle, with the acceleration $\dot{v}$ of hypothesis H_i. Similarly to the Equation (2.4), the distance $P(O|H_i)$ in this case is equal to:

$$P(O(t_j)|H_i) = \frac{1}{\sigma_a\sqrt{2\pi}} exp\left[-\frac{1}{2}\left(\frac{a(t_j) - \dot{v}_i(t_j)}{\sigma_a}\right)^2\right] \tag{2.6}$$

where i indicates the i-th hypothesis, j indicate the j-th time instant and σ_a is the standard deviation for the acceleration. A possible way to calculate the acceleration $\dot{v}$ as a function of H_i is using of a simple, continuous model give as:

$$\dot{v} = a_{MAX}\left[1 - \left(\frac{v}{u}\right)^{\delta}\right] \tag{2.7}$$

where a_{MAX} is the maximum acceleration offered by the car, v the speed of the vehicle in hypothesis H_i, u the desired speed and δ a fixed acceleration exponent.
The desired speed u depends on the legal speed limit and the individual driver. For the turn modelling, it is set dynamically since drivers slow down when approaching the intersection, as showed in this equation:

$$u = min\left(\frac{v}{\sqrt[\delta]{1 - \dot{v}/a_{MAX}}}, \bar{u}\right), \tag{2.8}$$

while in case of straight trajectory the value will be:

$$u = min\left(v, \bar{u}\right), \tag{2.9}$$

where v is the speed of the vehicle and $\bar{u}$ is the legal speed limit.
In order to avoid mistakes on the longitudinal behaviour detection and to improve the robustness of this method, it is useful to take observations over the full time interval $[t-T_s, t]$. We obtain $P(O|H_i)$ by computing the average of the individual scores from Equation (2.6):

$$P(O|H_i) = \frac{1}{N}\sum_{j=1}^{N} P(O(t_j)|H_i) \tag{2.10}$$

where N is the number of equidistant time intervals of the set $[t - T_s, t]$.

Remark: In [4] and [5] the acceleration $\dot{v}$ is calculated by a Intelligent Driver Model (IDM). In traffic flow modelling, the IDM is a continuous car following model for the simulation of highway and urban traffic that considers more than a single vehicle. The model is equal to:

$$\dot{v}_{IDM} = a_{MAX}\left[1 - \left(\frac{v}{u}\right)^{\delta} - \left(\frac{d^*(v, \Delta v)}{d}\right)^2\right]. \tag{2.11}$$

In the above equation, Δv is the difference of speed between two following vehicles, d the gap between two vehicles and d^* the desired gap. As we can see, the influence of a preceding vehicle is represented by the ratio between the effective desired gap d^* and the actual gap d. Since we only consider in this work one vehicle, such term is discarded, leading us to Equation 2.7.

2.4 Hidden Markov Model

Driver intent identification has been studied in article [1], using a Hidden Markov Model (HMM). It consists in a suitable model grammars, a training set and an evaluation process.
Firstly we create a Markov chain that describes the process.
A Markov chain is a stochastic process with the Markov property: given a present state, the future state depends only to the present state. The term "Markov chain" refers to the sequence of random variables defining a process, with the Markov property defining serial dependence only between adjacent periods (as in a "chain"). The changes of state of the system are called transitions. Moreover, the probabilities associated with various state changes are called transition probabilities, and are equal to $a_{kl} = P(\pi_i =$

$l|\pi_{i-1} = k)$ where, at the time step i, a is the probability of a transition from state k to l, and π represents a path (sequence of states) through the model. The process is characterized by a state space, a transition matrix describing the probabilities of particular transitions, and an initial state (or initial distribution) across the state space.
For each state, there are an hidden part and an observed part. In our case the hidden part is the driver intent while the observed part are the values measured, for instance, by the GPS device: the position, the speed and the acceleration of the vehicle. The probabilities associated with the emission of a certain observed value in a specific state are called emission probabilities, and are equal to $e_k(b) = P(x_i = b|\pi_i = k)$ where, in the time step i, e is the probability of emitting value b in state k, while x_i represents the i-th emitted value, and π_i the i-th state. A state without any information to emit is called silent state.
In order to describe a trajectory, we use a discrete-time Markov chain with nine states plus two silent ones: one in the beginning and another in the end of the model that represent the start and the end of the trajectory. In particular, each state describes the dynamic of the vehicle in a specific instant of time. In fact, the nine states describe the evolution of the driver trajectory along the intersection. As suggested in [1], we use only nine states, since they are enough to describe the process and allow us to create a fast algorithm in the calculation phase.
Considering that we have only a single path to follow (see Figure 2.5), the transition probabilities are set to be equal to 1.
In order to simplify the model, we use the acceleration as the unique observed value. We decide to split the acceleration in 4 levels:

1. if $a < -0.5\, m/s^2$ the state emits F
2. if $a > -0.5\, m/s^2 \wedge a \leq 0\, m/s^2$ the state emits B
3. if $a > 0\, m/s^2 \wedge a < 0.5\, m/s^2$ the state emits C
4. if $a > 0.5\, m/s^2$ the state emits A

In this way, each state will emit a value in order to create a string of letters. This sequence x of letters describes the acceleration profile of the trajectory. For the HMM method we have different phases:

- Training
- Calculation of the probability

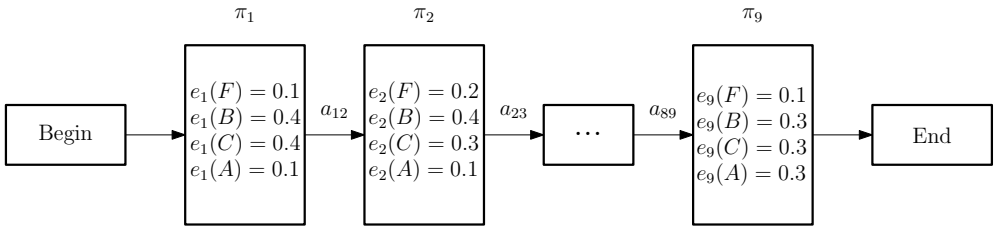


Figure 2.5: Timed Markov Chain with nine states

The phase of **training** is the preliminary part of the method. For each hypothesis H_i, associated to an intent I, we choose as observations O a set of trajectories with similar behaviour to H_i. We classify the acceleration of all the trajectories as a set of sequences x of letters (as we have seen above). Then, we use the Baum-Welch algorithm. This algorithm, also said Forward-Backward, is an algorithm that uses the observable data of the states to identify the values of the matrices of transition and emission of each hypotheses, that we call HMM parameters. In particular, the values of the matrices are estimated by counting the number of times each observable data is used across the training set of sequences.

In the **calculation of the probability** phase, the Forward algorithm computes in real time $P(O|H_i)$, i.e., the probability of a sequence x of observed values calculated through the matrices of transition and emission associated to the hypothesis H_i. The algorithm computes the probability for each intent I:

$$P(I_T) = \frac{P(O|H_T)\frac{1}{N_{H_T}}}{P(O|H_T)\frac{1}{N_{H_T}} + P(O|H_S)\frac{1}{N_{H_S}}} \tag{2.12}$$

$$P(I_S) = \frac{P(O|H_S)\frac{1}{N_{H_S}}}{P(O|H_T)\frac{1}{N_{H_T}} + P(O|H_S)\frac{1}{N_{H_S}}} \tag{2.13}$$

where N_{H_T} and N_{H_S} are the number of hypothesis for the turning and straight intent, $P(I_T)$ and $P(I_S)$ are the probability of the turning and straight intent, while $P(O|H_T)$ and $P(O|H_S)$ are the probabilities to have a sequence x conditioned to the occurrence of one of the H_T or H_S hypotheses:

$$P(O|H_T) = \sum_{i \in H_T} P(O|H_i) \tag{2.14}$$

$$P(O|H_S) = \sum_{i \in H_S} P(O|H_i) \tag{2.15}$$

In the sequel, we present in detail the mentioned algorithms.

Forward algorithm

The Forward algorithm is defined as follows. The value $f_k(i)$ is the probability of being in state k having observed the first i characters of x. We want to compute $f_N(L)$, the probability of being in the end state N, having observed all of $x := x_1, ..., x_L$.
The probability that we are in the beginning state and have observed 0 characters from the sequence is:

$$f_0(0) = 1 \tag{2.16}$$

while for k states that are not silent we have:

$$f_k(0) = 0 \tag{2.17}$$

The recursion for emitting states ($i = 1, \ldots, N$) is equal to:

$$f_l(i) = e_l(i) \sum_k f_k(i-1) a_{kl}, \tag{2.18}$$

where $e_l(i)$ is the probability of emitting character i in the state l, and a_{kl} is the transition probability from state k to l. Instead, the recursion for the silent states is give as:

$$f_l(i) = \sum_k f_k(i) a_{kl}. \tag{2.19}$$

The algorithm ends with this final equation:

$$P(O|H_i) = P(x) = P(x_1, ..., x_L) = f_N(L) = \sum_k f_k(L) a_{kN} \tag{2.20}$$

where $P(x)$ is the probability of having observed the entire sequence once we reach the final state.

Baum-Welch algorithm

This algorithm works as follow. Firstly, it compute the probability of generating a sequence x with the i-th symbol produced by the state k (for all x, i and k). The Forward algorithm gives us $f_k(i)$, the probability of being in state k having observed the first i values of x. The Backward algorithm gives us $b_k(i)$, the probability of observing the rest of x, given that we're in state k after i values.
The Backward algorithm is really similar to the Forward algorithm. The

initial value is $b_k(L) = a_{kN}$ for the states with a transition to the last state. The recursion is equal to $b_k(i) = \sum_l a_{kl} b_l(i)$ if l is a silent state, otherwise $b_k(i) = \sum_l a_{kl} e_l(x_{i+1}) b_l(i+1)$. At last, to end the algorithm, we have $P(x_1 \cdots x_L) = b_0(0) = \sum_l a_{0l} b_l(0)$ if the state is a silent, otherwise $P(x_1 \cdots x_L) = b_0(0) = \sum_l a_{0l} e_l(x_1) b_l(1)$.
Putting the Forward and Backward algorithms together, we can calculate the probability of the i-th symbol being produced by state k, given a sequence x:

$$P(\pi_i = k|x) = \frac{P(\pi_i = k, x)}{P(x)} = \frac{f_k(i) b_k(i)}{P(x)} = \frac{f_k(i) b_k(i)}{f_N(L)}. \tag{2.21}$$

We can calculate the expected number of times the value c is emitted by state k

$$n_{k,c} = \sum_{x^j} \left[\frac{1}{f_N^j(L)} \sum_{i|x_i^j = c} f_k^j(i) b_k^j(i) \right], \tag{2.22}$$

where j is the j-th sequence in the training set. We can also compute the expected number of times that the transition from k to l occurs:

$$n_{k \to l} = \sum_{x^j} \frac{\sum_i f_k^j(i)\, a_{kl}\, e_l(x_{i+1})\, b_l^j(i+1)}{f_N^j(L)}. \tag{2.23}$$

Finally, we can estimate the new emission and transition parameters as:

$$e_k(c) = \frac{n_{k,c}}{\sum_c n_{k,c}} \tag{2.24}$$

$$a_{kl} = \frac{n_{k \to l}}{\sum_m n_{k \to m}} \tag{2.25}$$

where $n_{k,c}$ is the expected number of emissions of c from state k for the training set, and $n_{k \to l}$ is the expected number of transitions from state k to state l. The new emission and transition parameters are the values that we will use in equations (2.18) and (2.19) for the intent identification.

2.5 Interacting Multiple Model

The last method analysed in this work is the Interacting Multiple Model and, in this case, we adopt the approach presented in [6]. The main idea of multiple model filtering is to run several filters in parallel, each one representing a different model of the system. Let x_k and P_k denote the system state's mean value and covariance matrix at time k, respectively, and Z_k represents all the observations up to k. Thus, the expected probability density

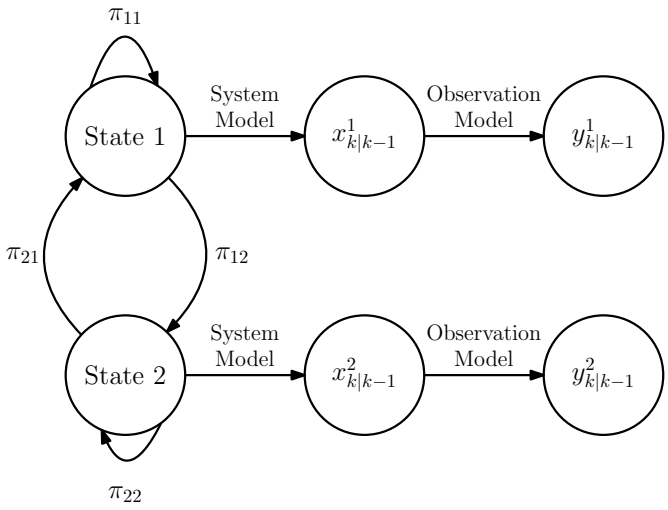


Figure 2.6: A Hidden Markov Model describing a system with two possible modes which are producing system state predictions and predicted observations by stochastic processes.

$p(x_k|Z_k)$ can be approximated as a Gaussian composition of the different model matched filters:

$$p(x_k|Z_k) \approx \sum_{i=1}^{N} \omega_k^i \mathcal{N}(x_k, x_k^i, P_k^i) \tag{2.26}$$

where the sum of the weights w_k^i is equal to 1. Depending on the real procedure used for determining the weights, it is possible to define different multiple model (MM) estimators. The static MM estimators assume that the system is constantly following one model, which is unknown to the estimator. Instead, dynamic MM estimators explicitly take into account the possibility that the system may change its model over time. Under this category, it is possible to add and remove mixture components in each estimation step.

The most popular implementation of the last approach is the Interacting Multiple Model filter (IMM). The system considers a unique state x_{k-1}^i. For each time step k, the system updates the value of state with the probability to pass from a model to another. The system uses then the transition matrix of the Hidden Markov Model (HMM) considering N states, as the number of model chosen (see Figure 2.6). The model is hidden because the model states can't be directly observed. Instead, every mode is producing an uncertain system state prediction $x_{k|k-1}^i$ by a stochastic process (described by the system model itself). In addition, every system state produces pre-

dicted observations $y^i_{k|k-1}$ by another stochastic process (described by the observation model). The purpose is to derive the state of the Markov model starting from the observations. Since the Kalman Filter is able to estimate the system state by looking at the observations, the task is reduced to determining the model from the system state vector and its covariance.
The transition probabilities of the Markov model are described by the Markovian matrix Π:

$$\Pi = \begin{pmatrix} \pi_{11} & \cdots & \pi_{1N} \\ \vdots & \ddots & \vdots \\ \pi_{N1} & \cdots & \pi_{NN} \end{pmatrix}. \tag{2.27}$$

In the Interacting Multiple Model (IMM) framework, a separate filter is running for each of the N models with a prior probability $\mu^j_{k=0} := P(m_{k=0} = H_j)$, with $j = 1...N$ is the j-th model and k the time step. Every time, before applying the Kalman filter equations ,the probability μ^j_k is updated using the transition probabilities defined in matrix (2.27):

$$\mu^{i|j}_{k-1} := P(m_k = H_i | m_{k-1} = H_j) = \frac{\pi_{ij}\mu^i_{k-1}}{\sum_{l=1}^{N} \pi_{lj}\mu^l_{k-1}} \tag{2.28}$$

where k is the time step, while i, l and j are the index of the Markovian matrix and also the specific model m of the algorithm, respectively.
In the iterative process, the probabilities are used to update the state vectors of the filters and their covariance. Thus, even if the filters are running in parallel, the probabilities (2.28) are not independent from each other. The result of the interaction process is calculated as follows:

$$x^j_{k-1} = \sum_{i=1}^{N} \mu^{i|j}_{k-1} x^i_{k-1}, \tag{2.29}$$

$$C^j_{k-1} = \sum_{i=1}^{N} \mu^{i|j}_{k-1} (C^i_{k-1} + (x^i_{k-1} - x^j_{k-1})(x^i_{k-1} - x^j_{k-1})^T), \tag{2.30}$$

where x^j is the state vector related to the j-th model and C^j its covariance. In the next step, we apply the standard Kalman filter equations to the system models in order to update the probabilities after the correction step. After this, we calculate the normalized Mahalanobis distance η^j between the predicted observation $y^j_{k|k-1}$ and the real measurement y_k:

$$\eta^j := \frac{1}{\sqrt{(2\pi)^n |S^j_k|}} e^{-\frac{1}{2}((\upsilon^j_k)^T (S^j_k)^{-1} (\upsilon^j_k))} \tag{2.31}$$

where v^j and S_j denote the model-specific innovation and its covariance. The Mahalanobis distance is able to determine the similarity of an unknown sample space compared to a known one.
Under a Gaussian assumption, the model probability $\mu_k^j := P(m_k = H_j|O)$, that is the probability of each hypothesis H, can be calculated as follows:

$$\mu_k^j = \frac{\mu_{k|k-1}^j \eta^j}{\sum_{i=1}^{N} \mu_{k|k-1}^i \eta^i} \tag{2.32}$$

where $\mu_{k|k-1}^j := P(m_{k-1} = H_j|O)$ is the probability to have the j-th hypothesis at time step $k-1$.
In order to compute the probability of the driver intent I we use the equations seen in the previous section: the equation (2.12) for the turning intent and the equation (2.13) for the straight intent.

Chapter 3

Numerical validation

In this chapter we will analyse the behaviour of the methods previously described. The methods will evaluate an arbitrary trajectory produced by a simulator and compute the probability of the driver intent. Firstly, we present the simulator and the algorithms seen in Chapter 2. Secondly, we provide the results of the classifier. Finally, we present a discussion, listing the strong and week points of each method.
We will consider just one hypothesis for intent, even if all the methods can work with different hypotheses per intent. Note that we implemented all the analysis, simulations and algorithms included in this section with Matlab R2014b.

3.1 Simulator

The purpose of the simulator is to create an arbitrary trajectory and a real-time testing routine for each method given in Chapter 2. As an output, we retrieve the probability of each intent.
In order to always have a different trajectory, we change the initial parameters (see Table 3.1) and randomly add a new values for each test. For a turning trajectory, we create a motion model $\ddot{s} = f(s, \dot{s}, u)$ (neglecting

Parameters	vmax	vin	vcurve	vafter	Tmax	s1	s2	cost	stop
Values	20 $\frac{m}{s}$	12 $\frac{m}{s}$	2.5 $\frac{m}{s}$	15 $\frac{m}{s}$	6 s	20 m	35 m	0.5	37 m

Table 3.1: Table of parameters used for the simulator: vmax is the maximum speed, vin is the initial speed, vcurve is the speed when turning, vafter is the desired speed after the braking, Tmax is the time limit of ours test, s1 and s2 are the position of braking, cost is the constant acceleration after the braking and, finally, stop is the position where the stop indicator is located.

the air drag and other fiction effects), where s and u are the position and the acceleration of the vehicle, respectively. We assume that the vehicle, approaching an intersection with a turning intent, keeps a constant speed, then brakes before the intersection, to finally accelerate and resume a constant speed after the intersection. For the straight trajectory, instead, we assume zero acceleration and constant speed. We generate the simulated trajectories using Algorithms 9 and 10 in Appendix A.

3.2 Presentation of the algorithms

3.2.1 Bayesian Approach

In this section, we present the algorithms used for the Simulation Based Approach (SBA) and Comparison Based Approach (CBA). The two hypotheses describing the driver behaviour for the turning and straight intent are given in Figure 3.1.

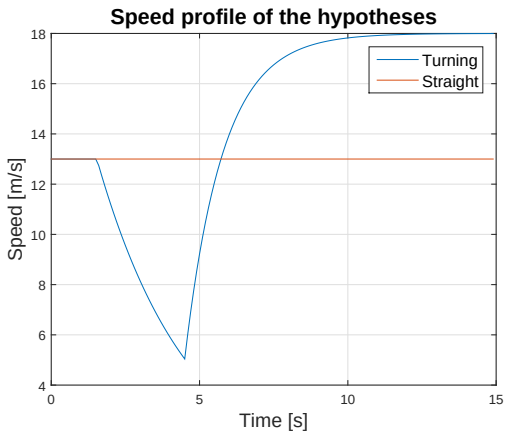


Figure 3.1: Bayesian Approach: Mean trajectory chosen as hypotheses

In order to compute the probability of the driver intent with the methods mentioned in the Section 2.3, we need the position, the speed and the acceleration of our hypotheses and those of the trajectory to be analysed. We provide an implementation of SBA and CBA in Algorithms 1 and 2, respectively.

Algorithm 1 Simulation Based Approach

Initialization of the vectors and parameters useful for the algorithm
Ts = time of simulation
$i0$ and $iEnd$ = index of the beginning and the end of the simulation
$x(\,)$ and $v(\,)$ vector of position and speed measured
$xh(\,)$ and $vh(\,)$ vector of hypothesized position and speed

for $i = 1$ to H(=Numbers of hypothesis) **do**
$s(1) = xh(i, i0)$ first value of the measured position
$vel(1) = vh(i, i0)$ first value of the measured speed
$\delta_S = xh(i, i0) - x(i0)$ difference between the measured position and the hypothesized position
$\delta_V = vh(i, i0) - v(i0)$ difference between the measured speed and the hypothesized speed
for $l = i0 + 1$ to $iEnd$ **do**
$s(l - i0 + 1) = x(l) + \delta_S$
$vel(l - i0 + 1) = v(l) + \delta_V$
▷ simulated trajectory shifted of δ
end for
$e(i) = \sqrt{\frac{s(iEnd-i0+1)-xh(i,iEnd)}{\sigma_S^2} + \frac{vel(iEnd-i0+1)-vh(i,iEnd)}{\sigma_V^2}}$
$f_{AH}(i) = \frac{1}{2\pi\sigma_S\sigma_V} e^{-\frac{1}{2}e(i)^2}$
▷ $e(i)$ is the discriminant that increases or decreases the final probability
end for
$p_{HO} = \frac{f_{AH} P_H}{\sum f_{AH} P_H}$

Algorithm 2 Comparison Based Approach

Initialization of the vectors and parameters useful for the algorithm
Ts = time of simulation
$i0$ and $iEnd$ = index of the beginning and the end of the simulation
a is the measured acceleration and a_{IDM} is the acceleration exit to the IDM

for $i = 1$ to H(=Numbers of hypothesis) **do**
for $l = i0$ to $iEnd$ **do**
$f_{AH}(l - i0 + 1) = \frac{1}{\sigma_A\sqrt{2\pi}}\, e^{-\frac{1}{2}\frac{a(l)-aIDM(i,l)}{\sigma_A^2}}$
end for
$f_{AHs}(k) = \frac{1}{\frac{Ts}{dt}} \sum f_{AH}$
end for
$p_{HO} = \frac{f_{AHs} P_H}{\sum f_{AHs} P_H}$

3.2.2 Hidden Markov Model

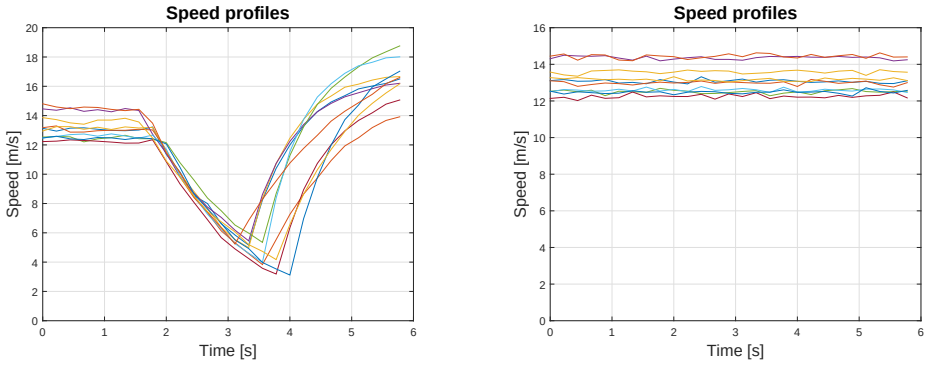
As explained before, this method is composed of two parts:

- The first part, called **training**, simulates an high number of trajectories in an intersection and, with the Baum-Welch algorithm, uses the observable state values of each trajectory to compute the HMM parameters.

- The second part, called **calculation of the probabilities**, uses the Forward algorithm to compute the probability of the driver intent.

Training

The training trajectories are generated by simulation using Algorithms 9 and 10 in Appendix A. We simulate ten training trajectories for each hypothesis, as shown in Figures 3.2(a) and 3.2(b).



(a) CBA (b) SBA

Figure 3.2: Training: speed profiles for: (a) the turning trajectory and (b) the straight trajectory

Assuming that we know in advance the training trajectory generated by the simulator and, in particular, the passed time, we can easily compute the time fraction required to have exactly nine states describing the entire trajectory. After the classification of the acceleration of each training trajectory in a set of sequence, we can use the Baum-Welch algorithm to identify the emission matrix.

Calculation of the probabilities

Once the training part is complete, we can compute the probability of the driver intent given the simulator trajectory. More precisely, we obtain the sequence x that describe the behaviour of the acceleration for the considered trajectory. For each hypothesis, we then compute the probability of the sequence x using the Forward Algorithm 4 and the probability of the driver intent with the Algorithm 5.

Algorithm 3 Baum-Welch algorithm

Initialization of the matrix and parameters useful for the algorithm
n = number of samples training
S = list of strings
nn = matrix for the calculation with psudocode
▷ Pseudocode equal to 1
L = number of state in the Markov Chain
e = emission matrix

for $j = 1$ to n **do**
  [forw,f]=Forward(e,S,j)
  [back,b]=Backward(e,S,j)
  for $i = 1$ to L **do**
    if $S(j,i) =' C'$ **then**
      $nn(1,i) = nn(1,i) + \frac{f(i+1,i+1)+b(i+1,i+1)}{forw}$
    end if
    if $S(j,i) =' B'$ **then**
      $nn(2,i) = nn(2,i) + \frac{f(i+1,i+1)+b(i+1,i+1)}{forw}$
    end if
    if $S(j,i) =' A'$ **then**
      $nn(3,i) = nn(3,i) + \frac{f(i+1,i+1)+b(i+1,i+1)}{forw}$
    end if
    if $S(j,i) =' F'$ **then**
      $nn(4,i) = nn(4,i) + \frac{f(i+1,i+1)+b(i+1,i+1)}{forw}$
    end if
  end for
end for
for $j = 1$ to L **do**
  for $i = 1$ to 4 (maximum number of emission states) **do**
    $e(j,i) = \frac{nn(i,j)}{\sum_i nn(i,j)}$
  end for
end for

Algorithm 4 Forward algorithm

Initialization of the matrices, strings and parameters useful for the algorithm
len = length of the string S
f = square matrix of dimension 11x11 (9 states + 1 state in the beginning and 1 in the end)
$f(1,1) = 1$
▷ The initial state start with probability equal to 1

for $i = 1$ to len **do**
  if $S(i) = C$ **then**
    $E(i) = e(i,1)$
  end if
  if $S(i) = B$ **then**
    $E(i) = e(i,2)$
  end if
  if $S(i) = A$ **then**
    $E(i) = e(i,3)$
  end if
  if $S(i) = F$ **then**
    $E(i) = e(i,4)$
  end if
  ▷ E is an emission vector with the correspondent values of the sequence S
end for
for $k = 2$ to $(len + 1)$ **do**
  for $i = 2$ to $(len + 1)$ **do**
    $f(k,i) = E(i-1)f(k-1,i-1)$
    ▷ we ignore the transition matrix because exist only one path
  end for
end for
$f(len+2, len+2) = f(len+1, len+1)$
$forw = f(len+2, len+2)$

Algorithm 5 HMM algorithm

Initialization of the matrix and string useful for the algorithm
eG = emission matrix of the turn right trajectory
eD = emission matrix of the straight trajectory
S = string of the simulator trajectory

[forwG,fG]=Forward(eG,S)
[forwD,fD]=Forward(eD,D)
$P_G = \frac{forwG}{forwG+forwD}$
$P_D = \frac{forwD}{forwG+forwD}$

3.2.3 Interacting Multiple Model

For this method, we follow the equations already seen in Section 2.5. We choose one model with straight trajectory and one with a turning trajectory. If we neglect air rag and other friction effects, we can generally describe the trajectory of a car as a linear model given as:

$$\begin{cases} \dot{x(t + dT)} &= Ax(t) + Bu(t), \\ y(t) &= Cx(t) + Du(t), \end{cases} \tag{3.1}$$

where $x = \binom{s}{v}$ is the state vector (s and v are the position and the speed of the vehicle), $u = a$ is the input of our model (a is the acceleration of the car) and $y = v$ is the output. For a curving trajectory, the matrices are given as:

$$A = \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} \frac{1}{2}dT^2 \\ dT \end{bmatrix}, C = \begin{bmatrix} 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix}, \tag{3.2}$$

while for a straight trajectory:

$$A = \begin{bmatrix} 1 & dT \\ 0 & 1 \end{bmatrix}, B = \begin{bmatrix} 0 \end{bmatrix}, C = \begin{bmatrix} 0 & 1 \end{bmatrix}, D = \begin{bmatrix} 0 \end{bmatrix}. \tag{3.3}$$

where dT is the intersection step. We initialize the parameters of the first measured value of the trajectory (state x and covariance C), the initial probability of each model $\mu^j_{k=0} := P(m_{k=0} = H_j)$ and the transition matrices of the Markov states Π.
To compute the probability of each model, we use the following algorithm.

Algorithm 6 Interacting Multiple Model algorithm

Initialization of the parameters, matrices and strings useful for the algorithm
$x = [x1, x2]$ = initial mean vectors of speed and position
$P = [co1, co2]$ = initial matrices of covariance
vel = speed of the simulated trajectory
a = acceleration of the simulated trajectory and input in our Kalman filters
E = transition matrix
m = output probability
H = number of models

for $i = 1$ to H **do**
  for $j = 1$ to H **do**
    $mij(i,j) = \frac{E(i,j)m(i)}{\sum_i^H E(i,j)m(i)}$
  end for
end for
for $j = 1$ to H **do**
  $xJ(j) = \sum_{i=1}^{H} mij(i,j)x(i,j)$
end for
for $j = 1$ to H **do**
  $PJ(j) = \sum_{i=1}^{H} mij(i,j)(P(i,j) + (x(i,j) - xJ(j))(x(i,j) - xJ(j))')$
end for
[xJ(1),PJ(1),e(1),Cy(1)]=Kalman1(PJ(1),xJ(1),co1,T,vel,a)
[xJ(2),PJ(2),e(2),Cy(2)]=Kalman2(PJ(2),xJ(2),co2,T,vel,a)
  ▷ The output of function Kalman is the mean and covariance prediction (xJ and PJ), and the error of the Kalman filter and its covariance (e and Cy)
for $j = 1$ to H **do**
  $\eta(j) = \frac{1}{\sqrt{2\pi|Cy(j)|}} e^{-\frac{1}{2}e(j)(Cy(j))^{-1}e(j)'}$
 ▷ We use the error and its covariance in order to calculate η which is the value that we use to calculate the driver intent probability
end for
for $j = 1$ to H **do**
  $m(j) = \frac{m(j)\eta(j)}{\sum_{j=1}^{H} m(j)\eta(j)}$
end for

3.3 Simulation result

In order to evaluate the efficiency of the algorithms, we will test the different methods on a generated trajectory. In order to compare the performance of each algorithm, we define here the considered criteria.

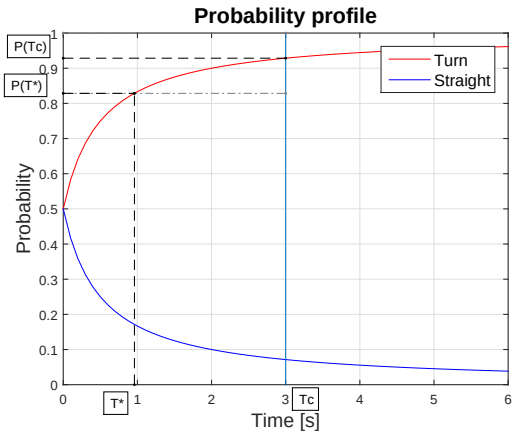


Figure 3.3: Illustration of the performance values for a probability profile

T_c is the time instant when the vehicle reaches the middle of the intersection (illustrated as a blue vertical line in all figures), while $P(T_C)$ is the intent probability returned by an identification method at time T_c. We can define now the set $P^* := [P(T_c) \pm 0.1]$, while the prediction time T^* will be considered such that:

$$T^* := min(\tau : P(t) \in [P(T_c) \pm 0.1], \forall t \in [\tau, T_c]), \tag{3.4}$$

Finally, we define, as main criterion, the prediction horizon $P(T_c) - P(T^*)$.

3.3.1 Turning trajectory result

Consider the trajectory presented in Figure 3.4, to be validated by the different algorithms.

In this simulated turning trajectory, we have $T_c = 3.7s$. As shown in Figure 3.5, a classifier based on the SBA method presents a stable, non-oscillating probability profile. We can see that the prediction horizon is 1.7 seconds, $T^* = 2s$ and $P(T^*) = 0.94$. Note that the time axis in Figure 3.5 starts at time $t = 1s$. This is because the Bayesian methods need to collect data for a period of 1 second ($T_S = 1s$) before producing a first estimation.

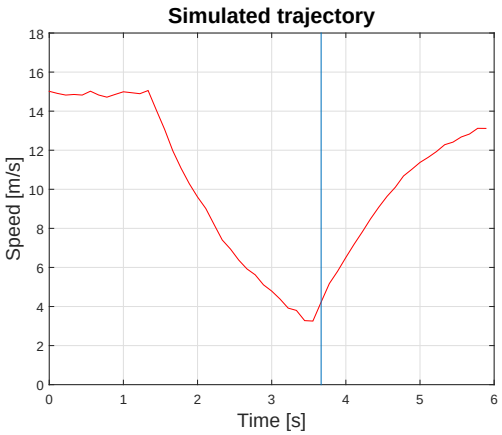
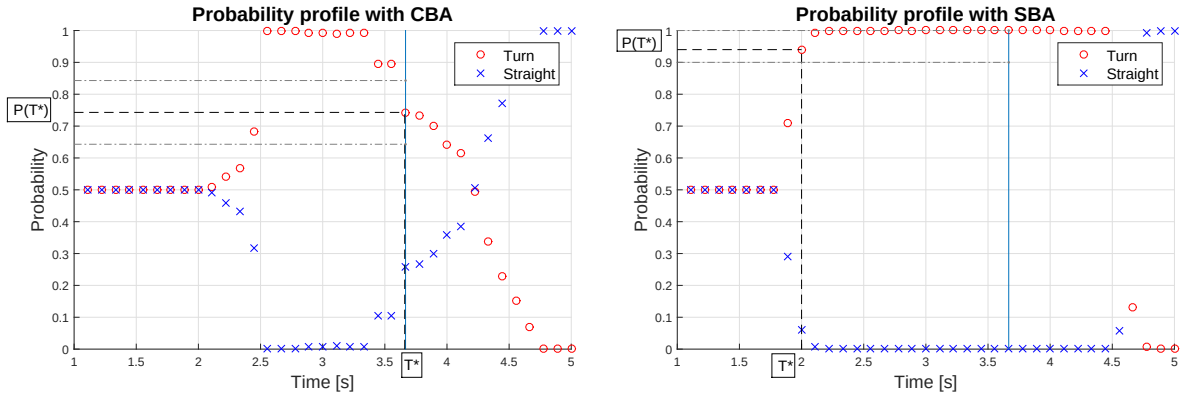


Figure 3.4: Turn: Speed profile generated by our simulator

Even if SBA and CBA are both based on the Bayesian method, the CBA presents a worse behaviour with respect to the SBA, as we can see in Figure 3.5. In particular, the prediction horizon is 0 seconds, with $T^* = 3.7s$ and $P(T^*) = 0.75$. One reason can be that the hypothesis speed profile is very similar to the simulated speed profile, while the hypothesis acceleration profile is more different from the simulated acceleration profile (Figure 3.6). As a consequence, it's more difficult for the CBA algorithm to recognize the correct intent.

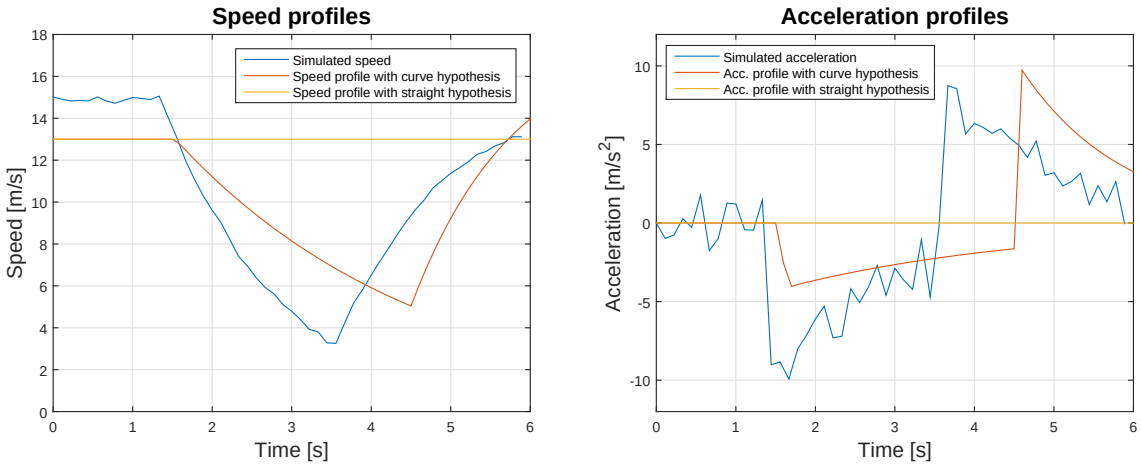
Consider the results for the classifier based on the HMM method, as shown in Figure 3.7. We can see that this method predicts the correct driver intent with prediction horizon equal to 0.9 seconds, with $T^* = 2.8s$ and probability $P(T^*) = 0.96$. The efficiency of this method is highly dependent on the training part, where we identify the HMM parameters of the training trajectories.

Finally, consider the results for the classifier based on the IMM method, as shown in Figure 3.8. We have a prediction horizon equal to 1.9 s, with probability $P(T^*) = 0.99$ and $T^* = 1.9s$. We can see that the models of the IMM are identical except for the acceleration of the vehicle. If the acceleration is low, then the state prediction is similar for the two models and it's difficult to understand which hypothesis is correct. This little difference between these two motion models explains the noisy behaviour (see Figure 3.8(b)).



(a) CBA (b) SBA

Figure 3.5: Simulation turning results - Probabilities of the two intents using: (a) the Comparison Based Approach and (b) the Simulation Based Approach



(a) Speed (b) Acceleration

Figure 3.6: Comparison of the speed and acceleration profiles in the Bayesian Approach

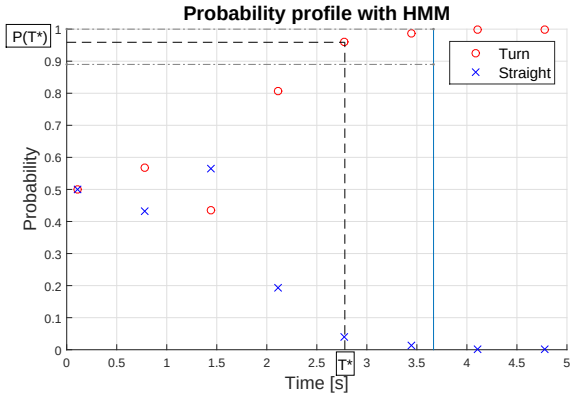
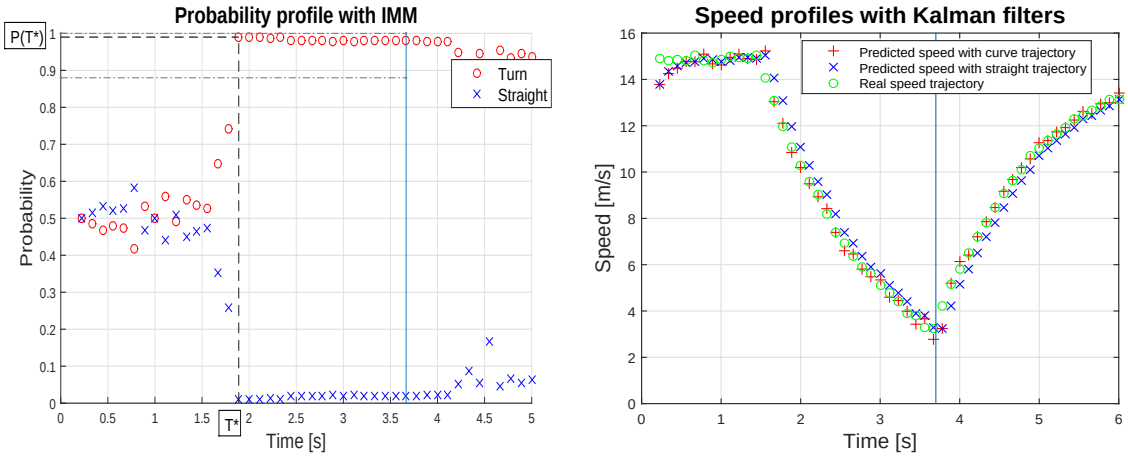


Figure 3.7: Simulation turning results - Probabilities of the two intents using the Hidden Markov Model



(a) IMM

(b) Comparison of the speed

Figure 3.8: Simulation turning results - Probabilities of the two intents using the Interacting Multiple Model, and comparison of the predicted speeds with the speed of the simulated trajectory

3.3.2 Straight trajectory result

In this case, the simulator generates a straight trajectory, as seen in Figure 3.9.

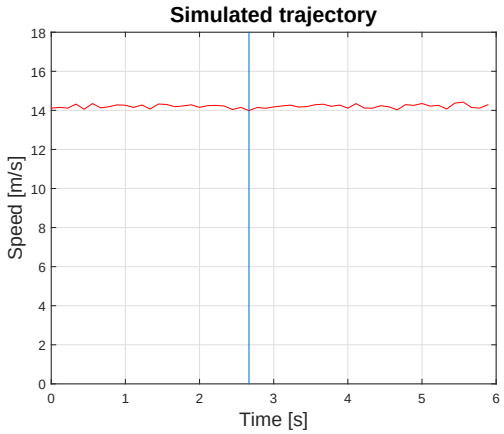
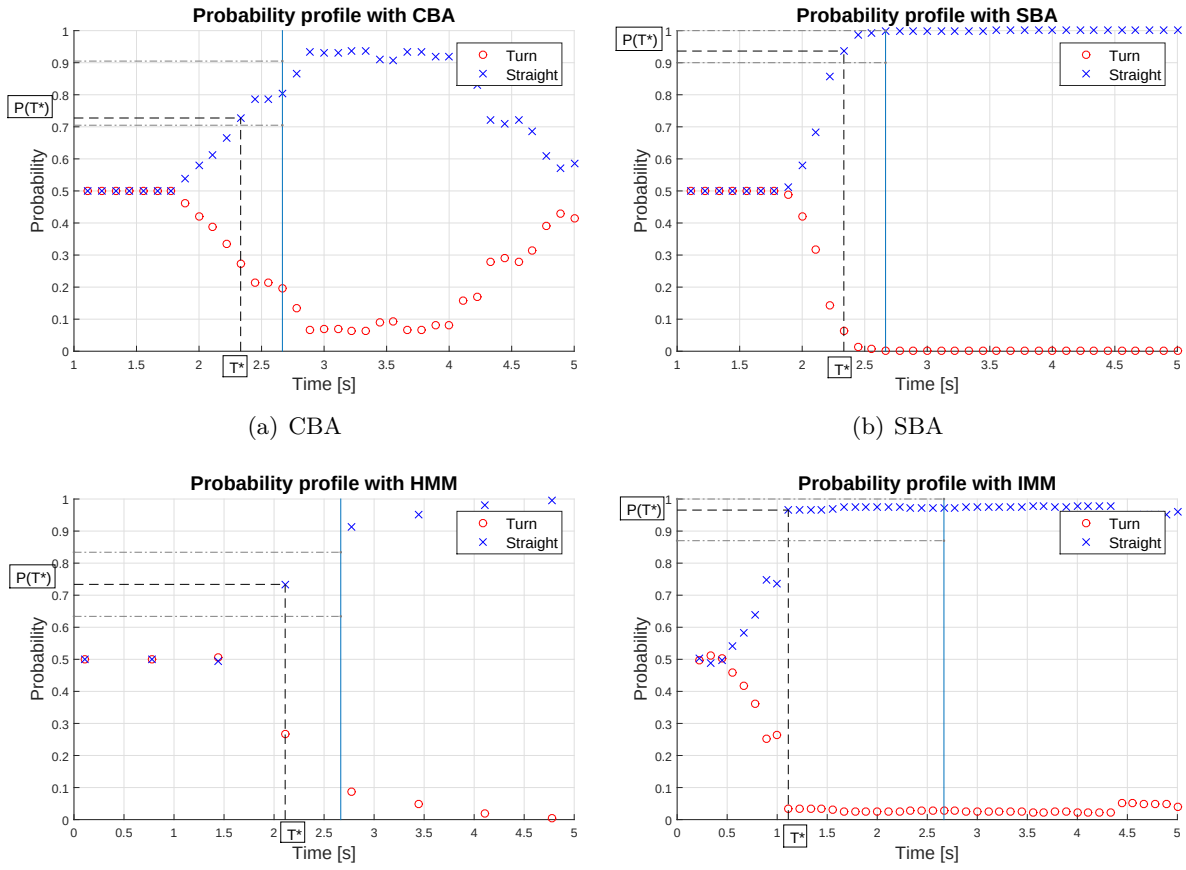


Figure 3.9: Straight: Speed profile generated by our simulator

The overall performances are very similar to the previous ones. In this case, however, all the methods recognize the correct driver intent later if compared with the results of the turning trajectory.
In this simulated straight trajectory, we have $T_c = 2.7s$. As we see in Figure 3.10(a), the SBA method predicts the driver intent with a prediction horizon of 0.3 seconds, with $T^* = 2.4s$ and the correspondent probability $P(T^*) = 0.94$. The CBA, instead, has a worse result when compared to the SBA, as we can see in Figure 3.10(a). This method predicts the correct intent with a prediction horizon of 0.3 seconds, but with $T^* = 2.4$ and low probability $P(T^*) = 0.72$.
The HMM method predicts the driver behaviour with a prediction horizon of 0.6 seconds, with $T^* = 2.1s$ and probability $P(T^*) = 0.73$ (see Figure 3.10(c)).
Finally, the IMM method has a better result with respect to the others. It predicts the correct intent with a prediction horizon of 1.6 seconds, with $T^* = 1.1s$ and the probability $P(T^*) = 0.97$ (see Figure 3.10(d)).



(a) CBA (b) SBA

(c) HMM (d) IMM

Figure 3.10: Simulation straight results - Probabilities of the two intents using: (a) the Comparison Based Approach, (b) the Simulation Based Approach, (c) the Hidden Markov Model and (d) the Interacting Multiple Model

3.4 Discussion

In addition to the two cases discussed here, we tested several other trajectories in order to have a more complete understanding of the potential of each method. Overall, the results obtained are satisfying and accurate.
The CBA is the worst method analysed. In fact, it predicts the driver intent with small prediction horizon and low probabilities. Despite this, CBA has a robust behaviour. In all the tests it predicts the correct intent with positive result.
The SBA, instead, predicts the driver intent with larger prediction horizon with respect to CBA, and also with high probabilities and robustness.
The HMM, has the same performances of SBA. It predicts the driver intent with large prediction horizon, high probabilities and robustness.
The IMM has often an oscillating behaviour. In the tests we made off line, the method often makes mistakes in the prediction of the driver intent. However, when the prediction is correct, it is the best method analysed, with high probabilities and large prediction horizon.
Considering all the results, we understand the importance of a well made training setup and data processing, in order to have always correct results. However, our aim is to find the most accurate method that is able to predict a real driver behaviour. In the next section, we will then analyse a real trajectory using a set of sensor measured data.

Chapter 4

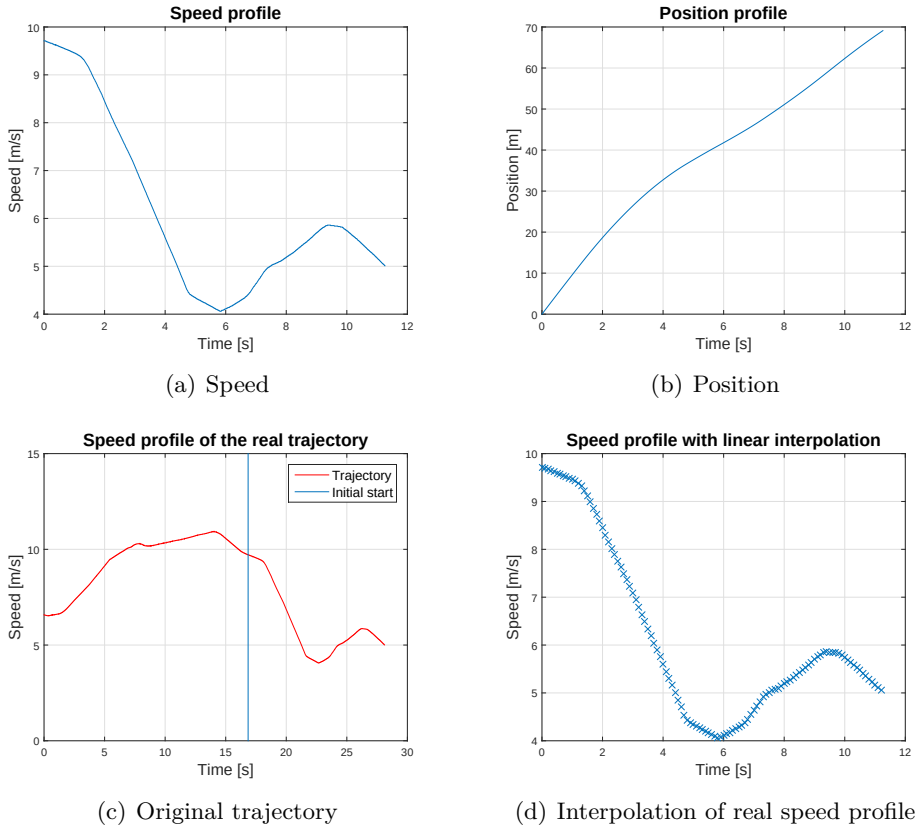
Real data validation

In this chapter, we test the algorithms using real trajectories (retrieved from a full scale car) instead of the simulated ones. We make use of a real trajectory measured with high precision GPS device.

4.1 Real data

In this analysis, we will exploit a set of real data provided by Chalmers University of Technology, where a team of researchers studied collision avoidance techniques for traffic intersections.
The data set includes several trajectories in a urban environment. The vehicle does not perform trajectories on purpose but acts, instead, like a normal car in an ordinary journey in a city road: entering a parking lot, turning at a curve, stopping at the beginning of a crossroad or turning at a round-about. For our work, we select a subset of data related to a suitable turning or a straight behaviour. As mentioned before, all the data of the vehicle trajectory has been measured by a GPS device installed inside the car. Moreover, we make use of a video recorded from the interior of the car. In particular, this video file help us to understand when the vehicle is entering the intersection.
The data set provides a huge amount of data, among which the latitude, longitude and altitude, the angle of pitch, roll and heading, the angle of slip, track, curvature and many others. We select the data related to the dynamic of the vehicle: the time, position, speed and acceleration.
In order to normalize all the trajectories, it is necessary to choose a common reference point. We consider as initial configuration the state of the vehicle when positioned 37 meters from the intersection. In particular, using the recorded video and the trajectory data, we identify the time and the

position when the car was in the middle of the intersection. Once defined that position, we subtract 37 meters in order to get the initial position of the trajectory and, by consequence, the initial time. Once found the initial position and time, we can easily define the trajectory from the starting point until the end of the data or, if necessary, until the vehicle reaches the middle of the intersection.



(a) Speed

(b) Position

(c) Original trajectory

(d) Interpolation of real speed profile

Figure 4.1: Construction of the speed and position profile of a real trajectory

In Figure 4.1(a) and 4.1(b), we can see the speed and the position profiles of one of the real trajectories extracted from the original one as we can see in Figure 4.1(c).
Note that the data we have available was not uniformly sampled in time. In order to obtain a trajectory vector with uniform time sampling, we interpolate linearly the original data with a time step of $0.1s$, as shown in Figure 4.1(d).

4.2 Improvement of the methods

In order to improve the performance of the algorithms, we decide to consider the information learned from the numerical tests in Chapter 3 to change, when possible, the properties of each method.

4.2.1 Bayesian approach

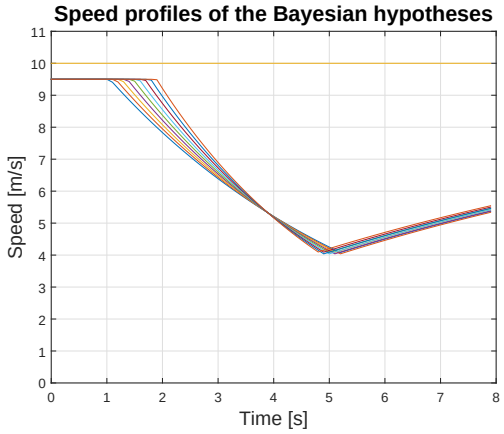


Figure 4.2: Illustration of the ten hypotheses of the Bayesian approach: nine for the turn case and only one for the straight case

As mentioned before, the outcomes of the simulation was already satisfactory for both methods (SBA and CBA). But, in order to improve the accuracy of the classifier, we decide to create more hypotheses for the turning case with different breaking times (Figure 4.2).
By adjusting the initial algorithm in such a way, we expect to increase the accuracy on the driver intent estimation, which should in practise increase the ahead-time prediction.

4.2.2 Hidden Markov Model

One possible improvement for this algorithm concerns the training phase. In our case, we don't have enough real trajectories for the training phase. For this reason, we train the HMM with simulated trajectories. This is, however, an handicap since the real trajectories have a more noisy behaviour when compared to the simulated ones. Bearing that in mind, we try to create the simulated trajectories as similar as possible to the real ones.
In order to have a more specific behaviour of the trajectories and a more

accurate emission matrix with asymptotic values, we also decide to increase the number of the training trajectories to 20 (rather than the previous ten) in the preliminary settings of training, as we said in the Section 3.2.2.
The most important improvement is related to the number of states in the Markov chain. If nine states are used to define the complete trajectory (before and after the intersection), it is then really hard to estimate accurately the driver intent before the vehicle reaches the intersection. In fact, in the Section 3.3, we showed that for each trajectory sample approximately 3 states are necessary to describe the trajectory after the crossroad, even if it is irrelevant for our analysis. For this reason, we decide to use all the 9 states to describe the behaviour of the car before it reaches the crossroad.

4.2.3 Interacting Multiple Model

In the simulation validation part, the IMM method revealed many issues in the prediction of the driver intent. In order to improve its behaviour, we decide to change two important settings.
Firstly, as done for the Bayesian approach, we increase the number of the hypotheses for turning intent in order to predict more accurately and with a higher anticipation the behaviour of the driver.
Secondly, we modify the general model used for describing the trajectory of a vehicle seen in Section 3.1. Rather than considering the acceleration as an input of each model (equal to the observed acceleration), we now consider it as a time-dependent parameter (equal to the acceleration of the relative hypothesis). With this improvement, we expect to obtain a bigger difference between the state predictions in output from the filters.

4.3 Results

From the original data set, we extract eight useful turning trajectories and five straight trajectories. For the sake of brevity, however, we will only analyse in this chapter a sub-set of these trajectories. We will compare the performance of the different methods by comparing the prediction horizon $P(T_c) - P(T^*)$ and the overall behaviour of the classifier.

4.3.1 Turn trajectory results

Turn 1

The first example considers a simple turn in a crossroad, with a speed profile shown in Figure 4.3.

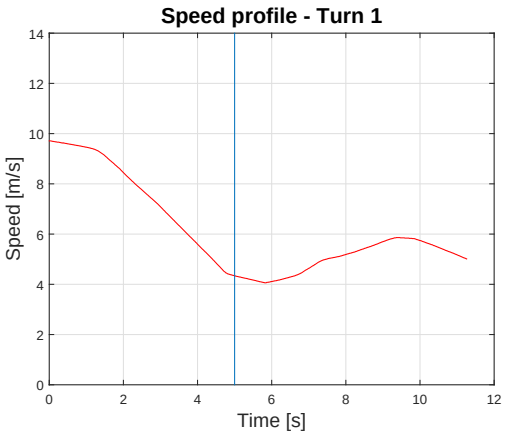
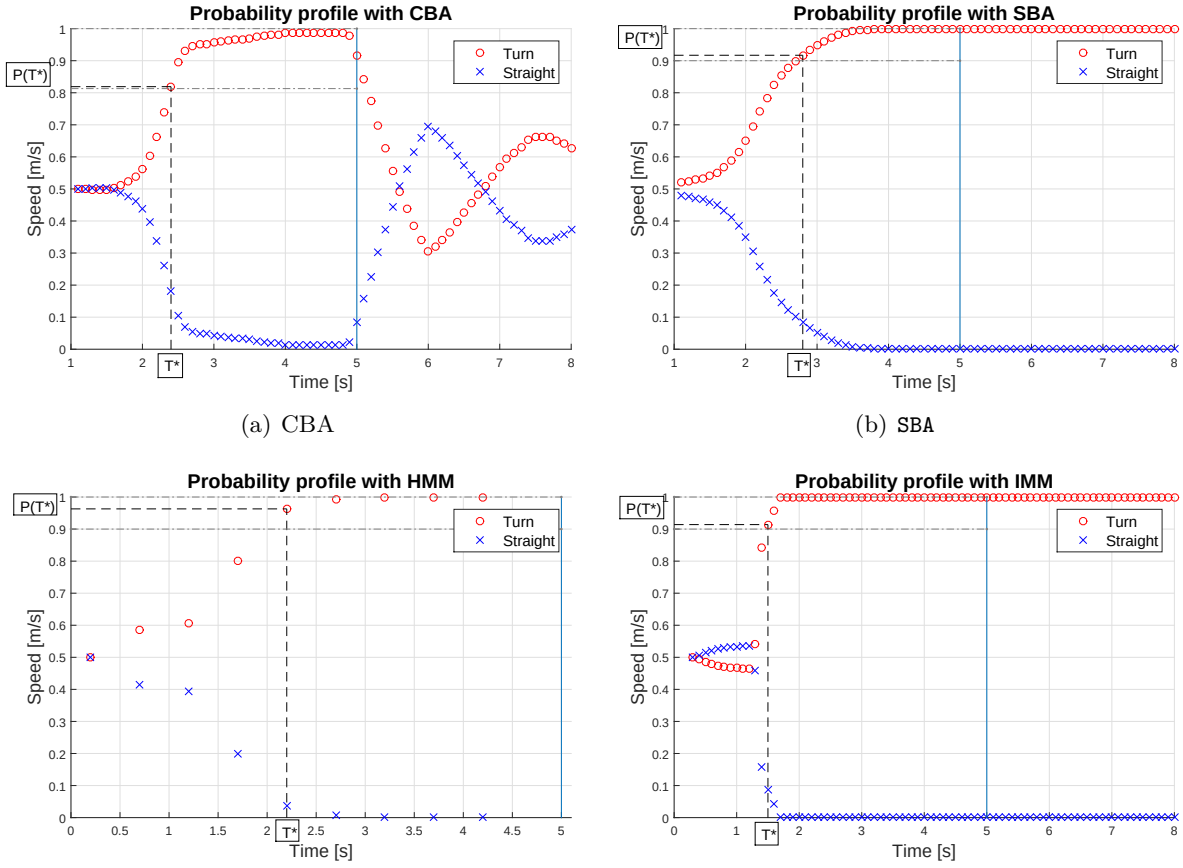


Figure 4.3: Turn 1 - Speed profile of a turning trajectory

The results of the first test are really good when compared to the simulated ones, as we can see in Figure 4.4. For all the methods we have a good reaction with respect to the braking of the vehicle.
In this turning trajectory we have $T_c = 5s$. For the SBA we have a prediction horizon of 2.2 seconds ($T^* = 2.8s$) with $P(T^*) = 0.91$. For the CBA we obtain a prediction horizon of 2.6 seconds, with $T^* = 2.4s$ and $P(T^*) = 0.82$. While for the HMM we have a prediction horizon of 2.8 seconds, with $T^* = 2.2s$ and $P(T^*) = 0.96$. In the end, the IMM predicts the correct intent with 3.5 seconds of prediction horizon, $T^* = 1.5s$ and $P(T^*) = 0.91$.



(a) CBA (b) SBA

(c) HMM (d) IMM

Figure 4.4: Turn 1 - Probability profile with: (a) Comparison Based Approach,(b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model

Turn 2

The second example considers a speed profile as shown in Figure 4.5. In this case, we have a turning trajectory with a smooth braking when compare to Turn 1. Naturally, this means that the vehicle will exhibit a lower deceleration, which allows us to expect that the methods where the acceleration is a decision parameter will fail or struggle to identify the driver intent.

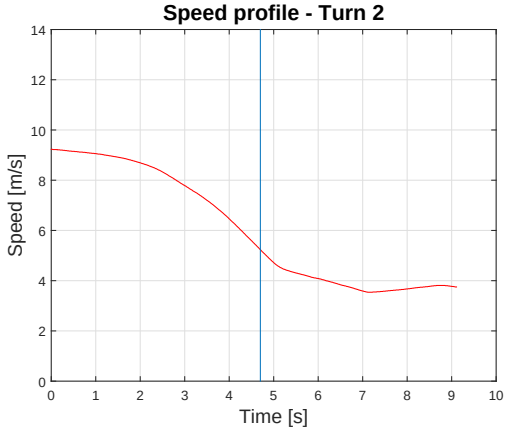
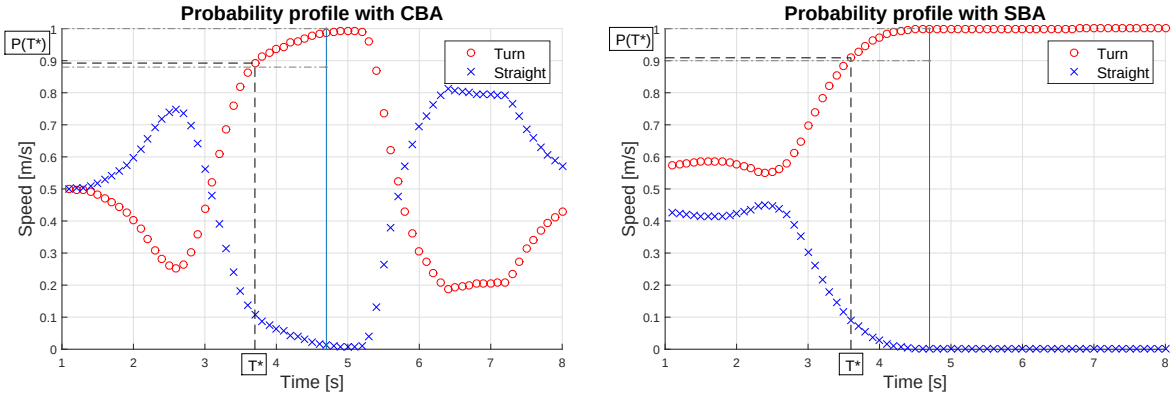


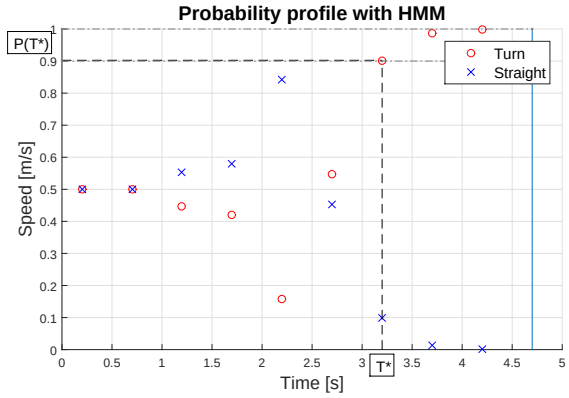
Figure 4.5: Turn 2 - Speed profile of a turning trajectory

As we expected, the SBA, HMM and IMM methods struggle to find the correct intent. As a result, we have an oscillating behaviour for the probability profiles. In this turning trajectory we have $T_c = 4.8s$. The SBA predicts the correct intent with 1.2 seconds of prediction horizon, with $T^* = 3.6s$ and $P(T^*) = 0.91$. Instead, the CBA predicts the intent with 1 second of prediction horizon, with $T^* = 3.8s$ and probability $P(T^*) = 0.9$. The HMM with 1.6 seconds of prediction horizon, with $T^* = 3.2s$ and the probability $P(T^*) = 0.9$. Finally, the IMM with 1.4 seconds of prediction horizon, with $T^* = 3.4s$ and $P(T^*) = 0.95$.

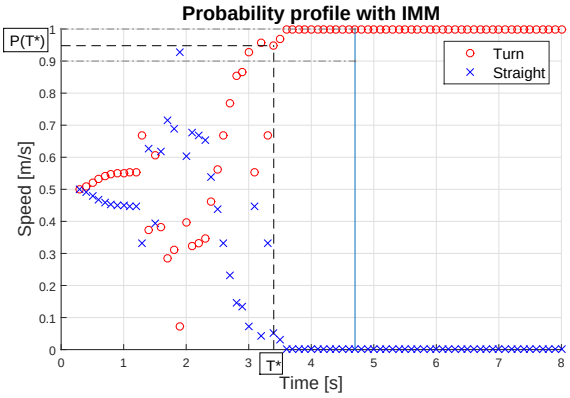


(a) CBA

(b) SBA



(c) HMM



(d) IMM

Figure 4.6: Turn 2 - Probability profile with: (a) Comparison Based Approach, (b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model

Turn 3

In this case, we have a turning trajectory with a lower initial speed and a smooth deceleration, as we can see in Figure 4.7.

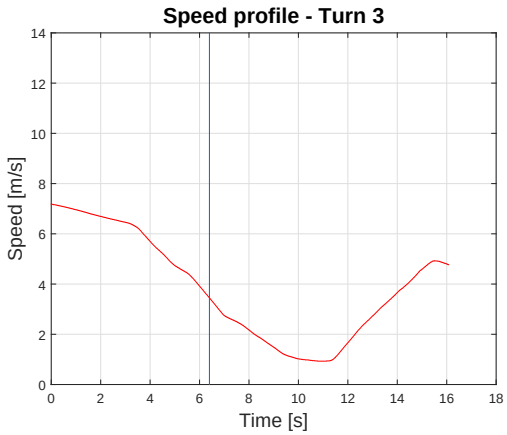
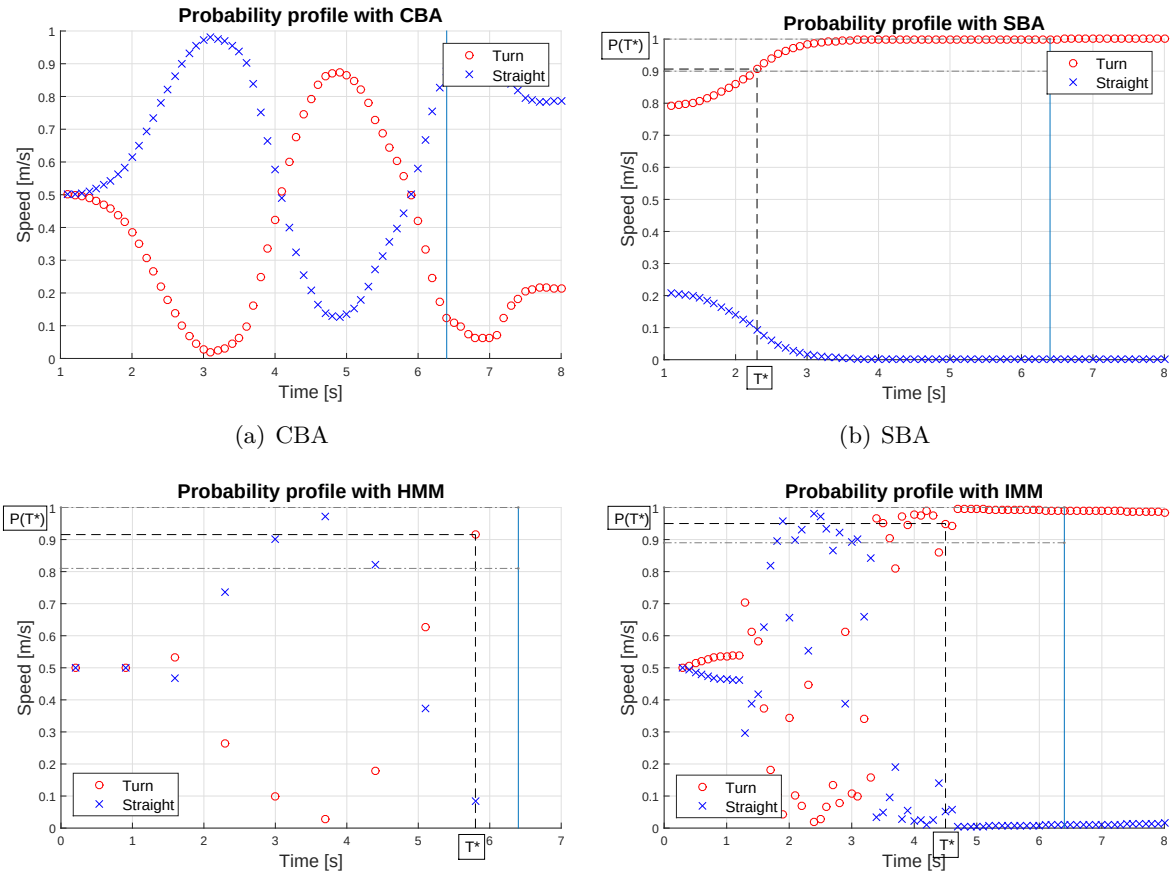


Figure 4.7: Turn 3 - Speed profile of a turning trajectory

Considering a smooth braking, then a low deceleration, the CBA, HMM and IMM have the same problem as the case before, struggling to find the correct driver intent.
In this turning trajectory we have $T_c = 6.4s$. As we can see in Figure 4.8(a), the CBA does not predict the correct intent stably. It has, in fact, an oscillating behaviour. This is due to the low similarity between the acceleration profiles of the hypotheses and the acceleration profile of this case.The HMM also shows bad performances in this case. In fact, it predicts the wrong intent for all the states, except the last two. The method predicts the correct driver intent with prediction horizon of 0.6 seconds, with $T^* = 5.8s$ and probability $P(T^*) = 0.96$.
The IMM and the SBA, instead, provide a decent prediction for this kind of trajectory. The IMM forecasts the driver intent thanks to the Kalman predictor that guesses the trajectory in the next state despite the low speed or the braking time. It predicts the correct intent with prediction horizon of 1.9 seconds, with $T^* = 4.5s$ and probability $P(T^*) = 0.95$.
The SBA predicts the driver intent extremely well with a prediction horizon of 4.1 seconds, with $T^* = 2.3$ and probability $P(T^*) = 0.9$. Unfortunately, the behaviour of the SBA is acceptable only for low speeds. In fact, in case we want to go straight with a low speed, the SBA will predict the turn intent in any case.



(a) CBA (b) SBA

(c) HMM (d) IMM

Figure 4.8: Turn 3 - Probability profile with: (a) Comparison Based Approach, (b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model

Turn 4

The final example consider a speed profile as shown in Figure 4.9. This case is the most complicated one: the car reaches the crossroad without braking and with low speed. Without considering the braking phase, the methods don't have useful data to compute the probability.

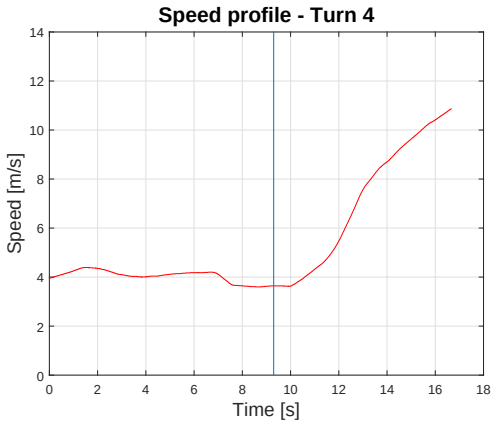
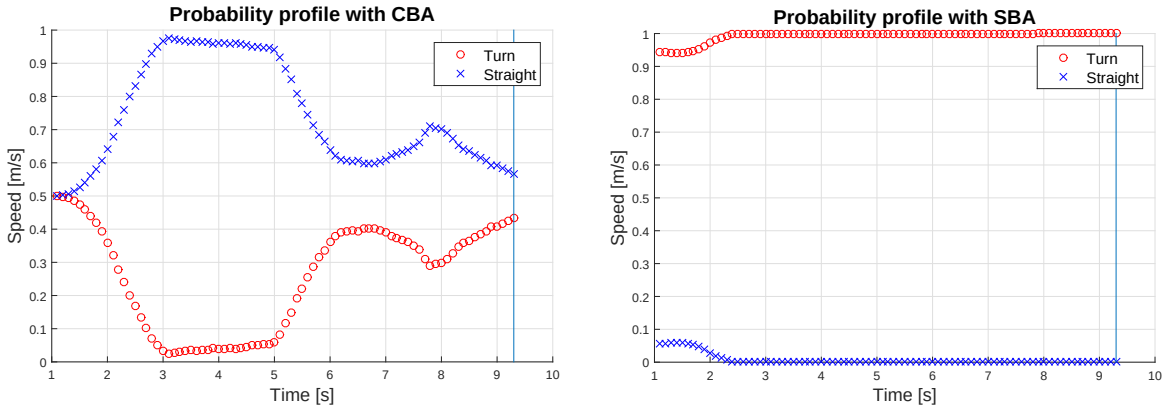


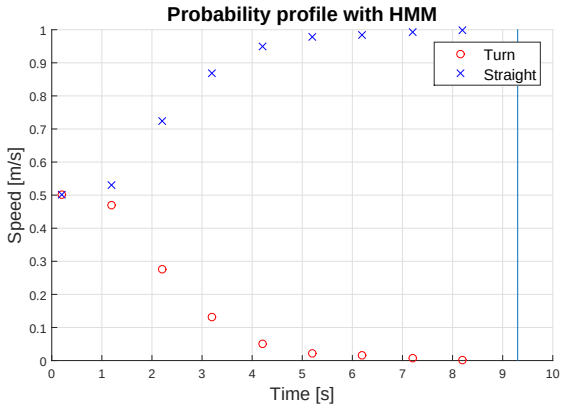
Figure 4.9: Turn 4 - Speed profile of a turning trajectory

As expected, all the algorithms that use the acceleration to compute the driver intent always obtained the wrong classification. Only the SBA predicts the correct intent in advance and with a high probability, for the same reason we explained in the case Turn 3. We also underline that the IMM method stops after 6 seconds since the covariance matrix becomes singular.

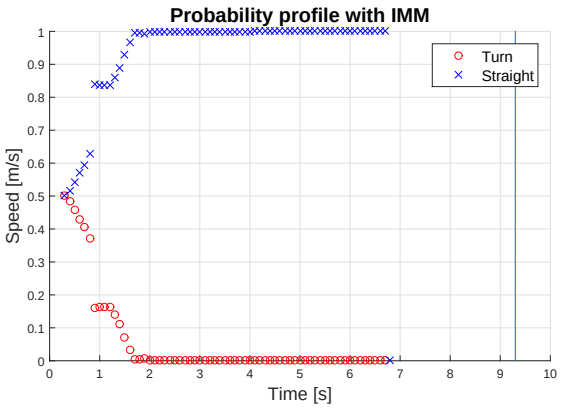


(a) CBA

(b) SBA



(c) HMM



(d) IMM

Figure 4.10: Turn 4 - Probability profile with: (a) Comparison Based Approach, (b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model. The values of T^* and $P(T^*)$ are not reported since the methods don't predict the correct intent

4.3.2 Straight trajectory results

Straight 1

In this case, we have a simple straight trajectory where the vehicle reaches the intersection with a stable speed (Figure 4.11).

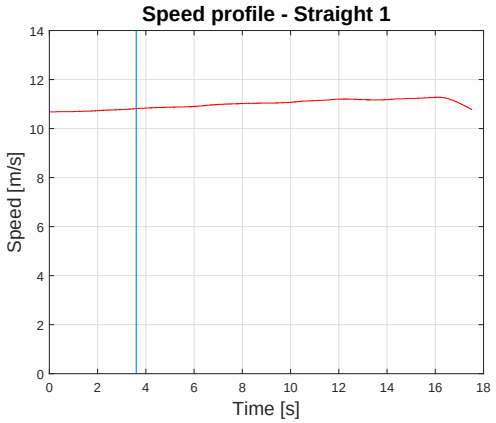
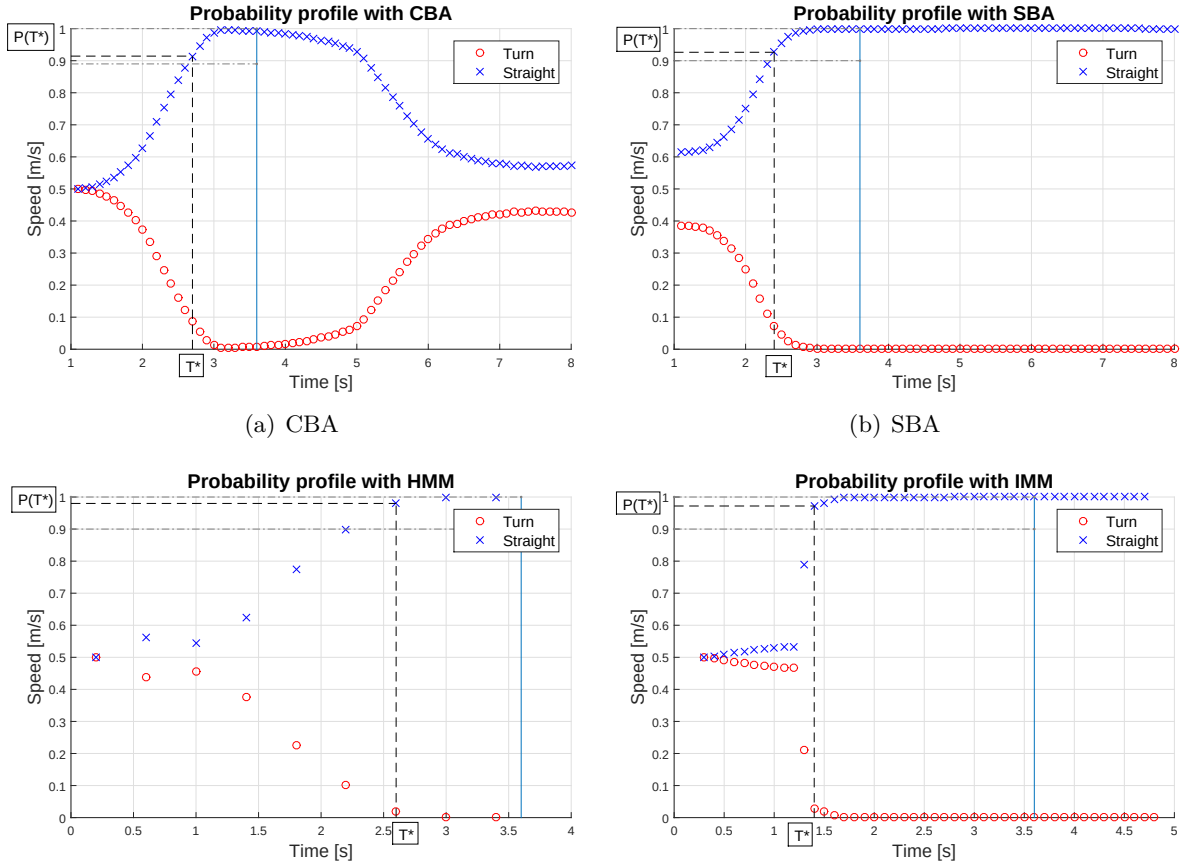


Figure 4.11: Straight 1 - Speed profile of a straight trajectory

The results of this first straight case are satisfying for all the approaches, as we can see in Figure 4.12. In this straight trajectory we have $T_c = 3.6s$. The SBA predicts the correct intent with prediction horizon of 1.2 seconds, with $T^* = 2.4s$ and probability $P(T^*) = 0.92$.
The CBA, instead, has a prediction horizon of 0.9 seconds, wih $T^* = 2.7s$ and $P(T^*) = 0.91$.
The HMM has a prediction horizon of 1 second, with $T^* = 2.6s$ and $P(T^*) = 0.98$.
In particular, the best performance is given by the IMM, that predicts the straight intent with prediction horizon of 2.2 seconds, with $T^* = 1.4s$ and probability $P(T^*) = 0.97$.



(a) CBA

(b) SBA

(c) HMM

(d) IMM

Figure 4.12: Straight 1 - Probability profile with: (a) Comparison Based Approach, (b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model

Straight 2

The behaviour of the straight trajectories included in our data set is always identical: the vehicle passes the intersection with a stable and constant speed and continues his initial path. In order to validate the results obtained in the previous case, we decide to select another trajectory (with speed profile as shown in Figure 4.13), similar to the previous one.

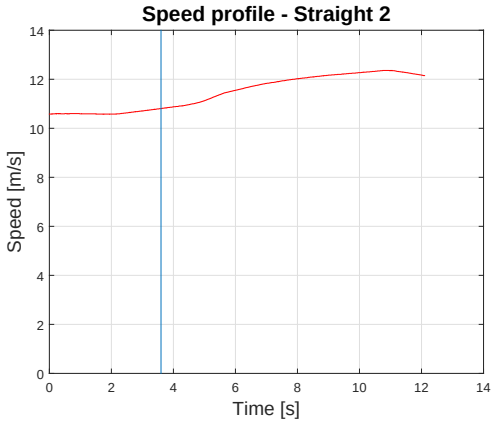
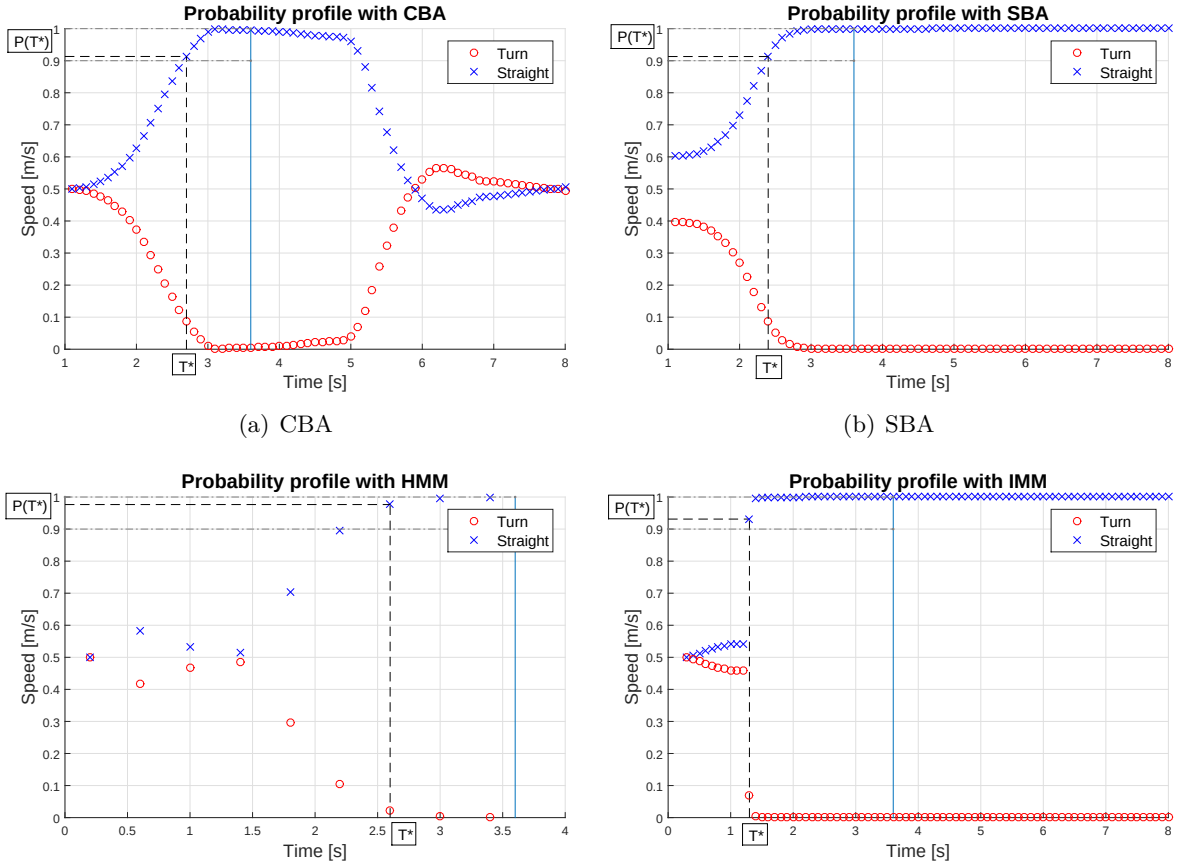


Figure 4.13: Straight 2 - Speed profile of a straight trajectory

As expected, the results are very similar to the previous one: all the methods provide a good performance in terms of ahead-time and probabilities levels.
In this straight trajectory we have $T_c = 3.6s$. The SBA predicts the correct intent with prediction horizon of 1.2 seconds, with $T^* = 2.4s$ and $P(T^*) = 0.91$.
The CBA, instead, has a prediction horizon of 0.9 seconds, with $T^* = 2.7s$ and $P(T^*) = 0.91$. The HMM has a prediction horizon of 1 second, with $T^* = 2.6s$ and $P(T^*) = 0.98$.
Again, the best performance is given by the IMM, that predicts the straight trajectory with a prediction horizon of 2.3 seconds, with $T^* = 1.3s$ and $P(T^*) = 0.93$.



(a) CBA

(b) SBA

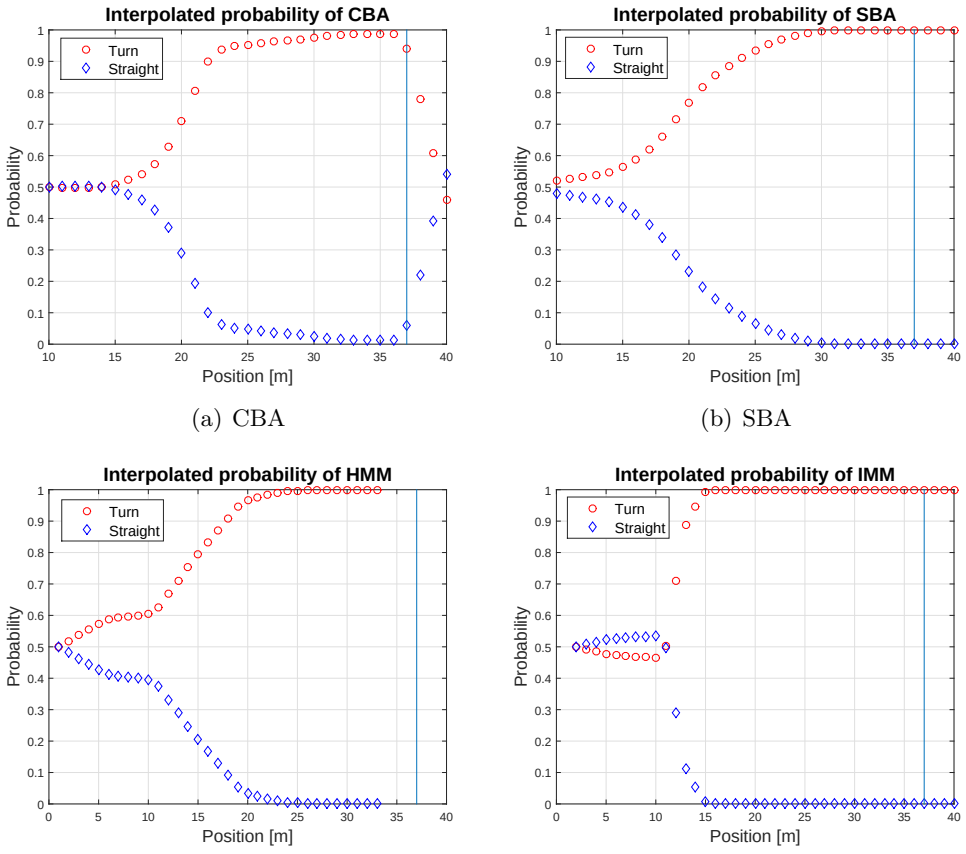
(c) HMM

(d) IMM

Figure 4.14: Straight 2 - Probability profile with: (a) Comparison Based Approach, (b) Simulation Based Approach, (c) Hidden Markov Model and (d) Interacting Multiple Model

4.4 Global results

In order to have a better comprehension of the methods' performance, we provide in this section an efficiency comparison.
The trajectories available were not enough to create meaningful charts of efficiency and, for this reason, we create and use fictitious trajectories. In particular, we add 8 fictitious trajectories for the turn intent and 13 for the straight one. In order to generate these fictitious trajectories, we considered the speed profile of measured real trajectories (as for the previous section), to which we add a random value to generate distinct trajectories.
In order to analyse the performance of the different methods, we decided to plot the probability profiles with the position of the vehicle instead of the time. As an example, we can see in Figure 4.15 the new layout of the case Turn 1, with a position interval of one meter.



(a) CBA (b) SBA

(c) HMM (d) IMM

Figure 4.15: Turn 1 - Probability profiles plotted with the position of the vehicle

In the sequel, we analyse the behaviour of the methods for four different positions: at 15 meters, at 20 meters, at 25 meters and 30 meters from the starting point. The probability associated to each case is presented in Figure 4.16. Looking at these bar charts we can see the probability evolution at the chosen positions. For example, for the case Turn 1, we can see that the SBA and CBA struggle to find the correct intent with high probability at 15 and 20 meters from the starting point. Instead, the HMM and IMM predict the correct intent for each selected point with high probability level.

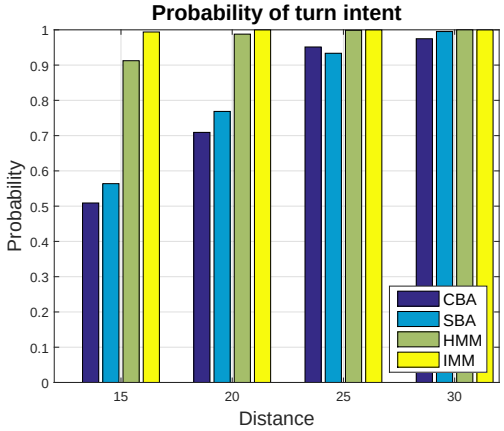
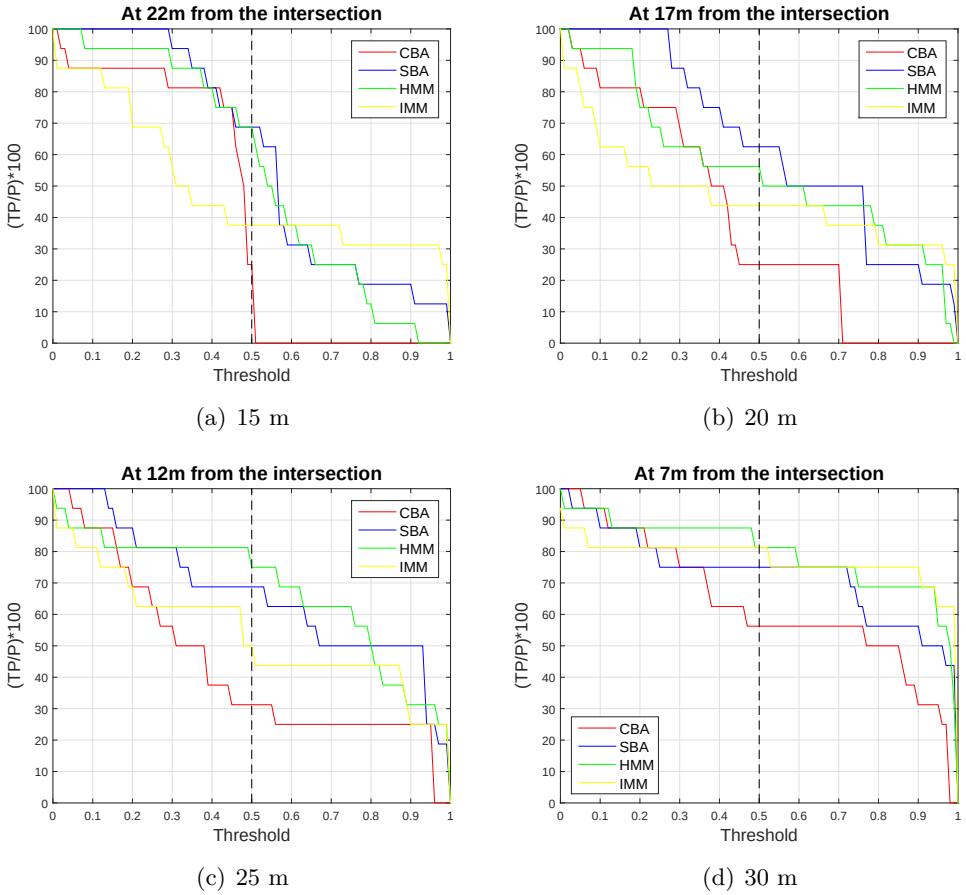


Figure 4.16: Turn 1 - Probabilities of the selected positions

These results let us think that IMM and HMM are the best methods for the prediction of the driver intent. In order to confirm or reject this first impression, we now present the result of the efficiency analysis. We will use the statistical measures of sensitivity and specificity, also known in statistics as classification functions:

- Sensitivity (also called true positive rate) measures the proportion of positives that are correctly identified;
- Specificity (also called true negative rate) measures the proportion of negatives that are correctly identified.

In our case, we compute the sensitivity as the number of true positive TP (turning trajectories correctly identified) over the number of turning trajectories P. Instead, we compute the specificity as the number of true negative TN (straight trajectories correctly identified) over the number of straight trajectories N.



(a) 15 m

(b) 20 m

(c) 25 m

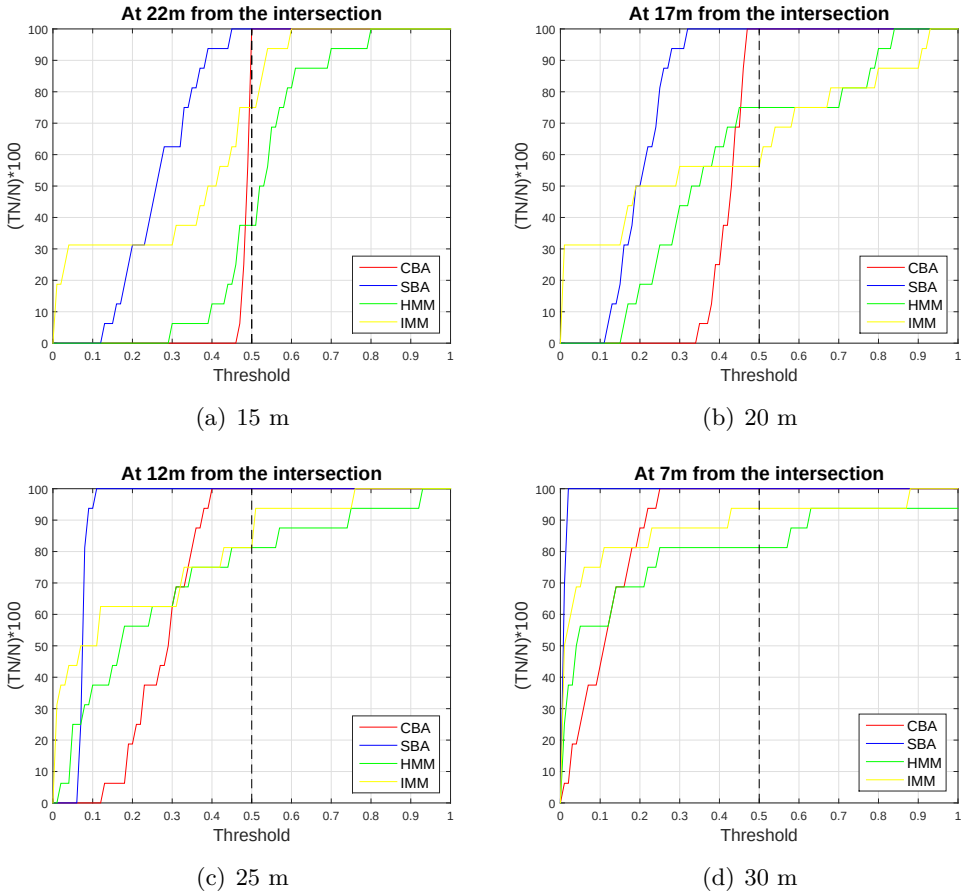
(d) 30 m

Figure 4.17: Sensitivity of each method varying the position and the threshold

We calculate all those values varying the threshold of probability [1]. Considering Algorithm 7, we obtain the following charts.

In Figure 4.17 we can see the efficiency results for the turning trajectories. In order to consider a performance as good, we want high sensitivity for all thresholds greater than 0.5, that represents a random decision.
We can see in Figure 4.17(d) that only at 7 meters from the intersection we obtain high sensitivity values, while in the other cases we can not predict with confidence the correct intent, unless using a low threshold value. The two methods with the highest sensitivity are the IMM and the HMM. Also

[1] If we test a turn trajectory and we set a threshold of 0.6, if the algorithm predicts that the probability of turning intent is greater than 0.6 we have a true positive, otherwise we have a false positive. Instead, if we have a straight trajectory and we set the same threshold of 0.6, if the algorithm predicts that the probability of turning intent is smaller than 0.6 we have a true negative, otherwise we have a false negative



(a) 15 m (b) 20 m

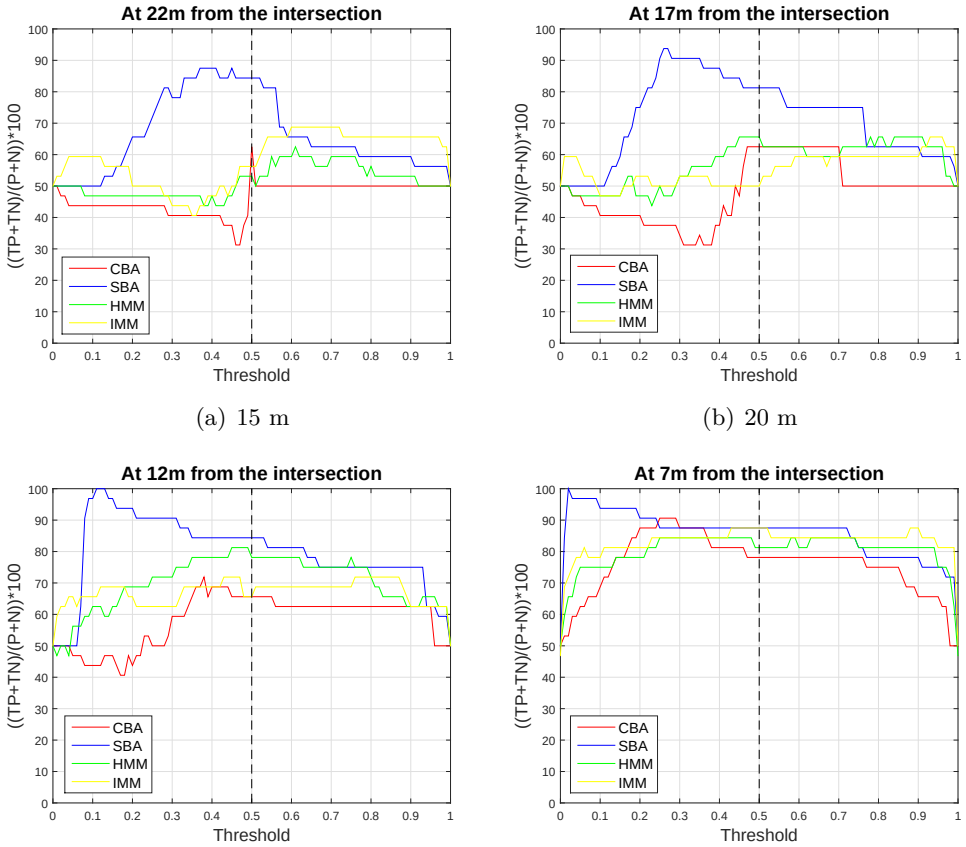
(c) 25 m (d) 30 m

Figure 4.18: Specificity of each method varying the position and the threshold

the SBA has good performance, while the CBA presents a bad sensitivity ratio in almost all the cases.

In Figure 4.18 we can see the efficiency results for the straight trajectories. In order to consider a performance as good, we want high specificity for all thresholds lower than 0.5.
In this case we have higher efficiency for all the methods. In fact, they succeed to predict the correct intent already at 12 meters from the intersection. The Bayesian Approach, in particular the SBA, predicts better the straight trajectories, obtaining a specificity equal to 1 for almost all the thresholds, as shown in Figure 4.18(d). The CBA has also high specificity, but only with higher thresholds.
The HMM and IMM have a similar behaviour. In fact, they both have high specificity with lower thresholds, if compared with the CBA.

The charts in Figure 4.19 show a combination of the sensitivity and



(a) 15 m

(b) 20 m

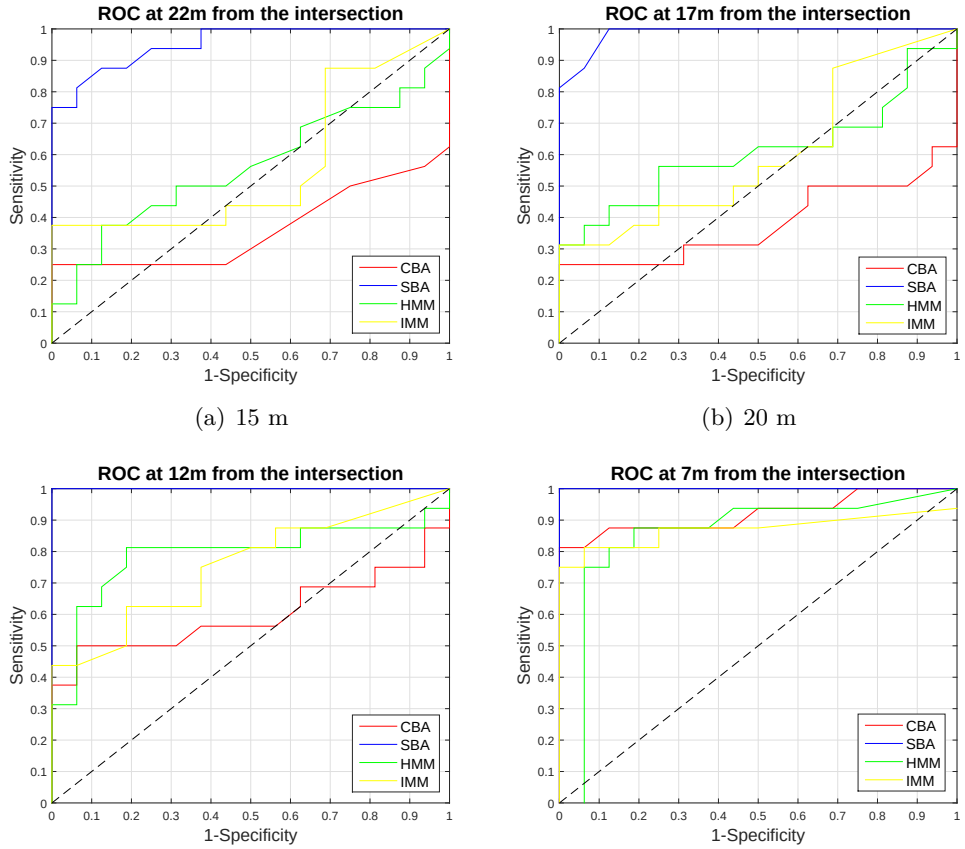
(c) 25 m

(d) 30 m

Figure 4.19: Efficiency of the methods varying the position and the threshold

specificity curves, thus the combination of Figure 4.17 and Figure 4.18. In particular, the left side of the charts presents the efficiency of the method to predict a straight trajectory, while the right side presents the efficiency to predict the turning one. A result is considered good if an high value of efficiency is observed for all thresholds.
We notice immediately that the SBA is the most performant method, with the highest efficiency for almost all the cases, and is the best method for the prediction of a straight trajectory. We can also notice in Figure 4.19(d) that the HMM and IMM methods have an high efficiency for all thresholds, both for the turning trajectory and for the straight trajectory. The Comparison Based Approach is the worst method in terms of efficiency, providing useful performance only when the vehicle is at 7 meters from the intersection, as shown in Figure 4.19(d).

The charts in Figure 4.20 are the sensitivity-specificity graphs, also called



(a) 15 m

(b) 20 m

(c) 25 m

(d) 30 m

Figure 4.20: ROC curves of each method, the dashed line rapresents the behaviour of a fictitious method with random decision

the Receiver Operating Characteristic (ROC) curves. In order to have good results, the ROC curves must be far from the dashed line: more the curves are distant from the dashed line and more the method will be efficient. From Figure 4.20, it is immediately possible to notice that the SBA is the most efficient method. This result is due to the great performance in the prediction of straight trajectories. The HMM and IMM show good performances with high efficiency for distances less then 12 m from the intersection as shown in Figure 4.20(c). Instead, the CBA obtains high efficiency only at 7 m from the intersection, as shown in Figure 4.20(d).

Algorithm 7 Algorithm create to find TP/P, TN/N and TP+TN/Total

Initialization of the vectors
$p[\,]$ = vector of probability of a specific method
NT = number of turn trajectories (P)
NS = number of straight trajectories (N)
k = counter for true positive
q = counter for true negative
$n[\,]$ = vector of threshold
l = number of threshold in the vector $n[\,]$

for $j = 1$ to l **do**
  for $i = 1$ to NC **do**
    if $p(i) >= n(j)$ **then**
      $k = k + 1$
    end if
  end for
  for $i = NT + 1$ to $NT + NS$ **do**
    if $p(i) <= n(j)$ **then**
      $q = q + 1$
    end if
  end for
  $a(j) = k/NT$
    ▷ a(j) is the j-th element of true positive over the total number of turn trajectories, and thus the sensitivity
  $b(j) = q/NS$
    ▷ b(j) is the j-th element of true negative over the total number of straight trajectories, and thus the specificity
  $c(j) = (k + q)/(NT + NS)$
 ▷ c(j) is the j-th element of the sum of true positive and true negative over the total number of samples trajectories
  $k = 0$
  $q = 0$
end for

Chapter 5

Conclusion

In this work, we considered different methods for calculating the driver intent probability when the vehicle approaches a generic intersection. Our goal was to identify and analyse the most performant methods. For the sake of simplicity, we considered only two possible trajectories: go straight or turning at the intersection. We considered the data related to the dynamics of the vehicle (position, speed and acceleration). Since we consider as input of the classifier solely the longitudinal dynamics of the car, all the algorithms must recognize the two intents based only on the braking pattern of the vehicle. In all the cases, if the car approaches the intersection with a stable, low speed and without a clear deceleration, almost all the methods fail to recognize the real driver intent. Furthermore, another weakness of the considered methods is that they are designed to work in a specific intersection. Also, the available set of real data at our disposal was rather limited. By consequence, we worked with a very small number of trajectories. In order to have better results we should increase the number of hypotheses and, in the case of HMM, train the algorithm with real data at the considered specific intersection. The following conclusion holds for the different methods.

- The *Simulation Based Approach* is the most appropriate method to predict a straight trajectory. It predicts the driver intent with large prediction horizon and high probability. However, in the cases when turning with low speed, the calculated probability is unreliable. As already underlined before, having low speeds makes SBA unable to distinguish the difference between a turning trajectory with a low speed and a straight trajectory with the same low speed profile. In order to improve this method we suggest to add additional hypotheses to the scenario.

- The *Comparison Based Approach* is less efficient when compared to SBA. In our opinion, the main problem lies in the complexity of modelling the acceleration of a real vehicle in the preliminary settings. In all the cases, in fact, we have large differences between the acceleration profile of our hypotheses and the acceleration measured from the GPS device. These differences produce small prediction horizons and low probabilities.
  This method can also be improved. As SBA, we can increase the number of hypotheses in order to have a more accurate prediction. Moreover, if we consider a scenario with more than a single vehicle, we can use the IDM equation to model the acceleration profile, considering also the relative distance between two following vehicles.

- The *Hidden Markov Model* predicts with high probabilities and high efficiency both the turning and the straight trajectory. This method can be largely improved, since in the training phase the method is able to integrate all the characteristics of a trajectory, and including them in the final emission matrix. For example, we can add other states in the Markov chain or add other emission values in the description of the trajectory, instead of only using four levels of acceleration.

- At last, the *Interacting Multiple Model*, as the HMM, predicts with high probability levels and high efficiency both the turning and straight trajectories. Thanks to the Kalman filter, it's possible to predict the chosen trajectory anticipating the driver behaviour. From our point of view, the best improvement is to create a more sensitive and accurate models for the Kalman filters, and adding different hypotheses that are able to describe all the driver behaviours when approaching an intersection.

To conclude, it's not possible to identify, in a objective way, one method that outperforms the others in every situation. Each method has its own advantages and drawbacks.
SBA is the method with the highest efficiency, but without a large prediction horizon. CBA is the worst method analysed in this thesis, but it has a good potential if we consider a scenario with more than a single vehicle. HMM has good performances in terms of probability levels and prediction

horizon, and can be largely improved in the training phase. IMM does not have good efficiency with respect to SBA, but it is the method with the largest prediction horizon.
We therefore suggest to use more than one method, focusing in the preliminary settings and in the training phase so to have the most accurate hypothesis model possible.

Bibliography

[1] Holger Berndt and Klaus Dietmayer. Driver intention inference with vehicle onboard sensors. In *Vehicular Electronics and Safety (ICVES), 2009 IEEE International Conference on*, pages 102–107. IEEE, 2009.

[2] Alessandro Colombo. A mathematical framework for cooperative collision avoidance of human-driven vehicles at intersections. In *Wireless Communications Systems (ISWCS), 2014 11th International Symposium on*, pages 449–453. IEEE, 2014.

[3] Alessandro Colombo and Domitilla Del Vecchio. Least restrictive supervisors for intersection collision avoidance: A scheduling approach. 2014.

[4] Martin Liebner, Felix Klanner, Michael Baumann, Christian Ruhhammer, and Christoph Stiller. Velocity-based driver intent inference at urban intersections in the presence of preceding vehicles. *Intelligent Transportation Systems Magazine, IEEE*, 5(2):10–21, 2013.

[5] Martin Liebner, Christian Ruhhammer, Felix Klanner, and Christoph Stiller. Generic driver intent inference based on parametric models. In *Proc. IEEE Intelligent Vehicles Symp.(IV)*, pages 268–275, 2013.

[6] Robin Schubert and Gerd Wanielik. Unifying bayesian networks and imm filtering for improved multiple model estimation. In *Information Fusion, 2009. FUSION'09. 12th International Conference on*, pages 810–817. IEEE, 2009.

Appendix A

Trajectory's algorithms

Algorithm 8 Algorithm used to create a random set of initial parameters

n = numbers of iterations
rand = Random value between 0 to 1
for $i = 1$ to n **do**
    $vmaxq(i) = vmax + 5rand$
    $vinq(i) = vin + 3rand$
    $vcurveq(i) = vcurve + 3rand$
    $vafterq(i) = vafter + 5rand$
    $costq(i) = cost + rand$
    $s1q(i) = s1 + 5rand$
    $s2q(i) = s2 + 5rand$
                ▷ We add a random value for each parameter
end for

Algorithm 9 Algorithm to create a turning trajectory

Initialization of the vectors
$x[\,]$ = vector of space
$v[\,]$ = vector of speed
$a[\,]$ = vector of acceleration
wn = white noise
dt = minimum time step
$Tmax$ = maximum time used to compute the trajectory
$s1q$ = position where the vehicle starts to braking
$s2q$ = position where the vehicle stops to braking

$v[1] = vinq + wn$
$s[1] = a[1] = 0$
▷ Initialization of the first element in the vectors
for $i = 2$ to $Tmax$ **do**
  $x(i) = x(i-1) + v(i-1)dt + \frac{1}{2}a(i-1)dt^2$
  $v(i) = vinq + wn$
  ▷ First constant part of the trajectory
  if $x(i) > s1q$ **then**
    $v(i) = vinq - \frac{vinq - vcurveq}{s2q - s1q}(x(i) - s1q) + wn$
    ▷ Braking part of the trajectory
  end if
  if $x(i) > s2q$ **then**
    $v(i) = v(i-1) + costq(vafterq - v(i-1))dt + wn$
    ▷ Acceleration part of the trajectory
  end if
  if $v(i) > vdopo$ **then**
    $v(i) = vafterq + wn$
    ▷ Reach the regime speed
  end if
  $a(i) = \frac{v(i) - v(i-1)}{dt}$
end for

Algorithm 10 Algorithm for create a straight trajectory

Initialization of the vectors
$x[\,]$ = vector of space
$v[\,]$ = vector of speed
$a[\,]$ = vector of acceleration
wn = white noise
dt = minimum time step
$Tmax$ = maximum time used to compute the trajectory

$v[1] = vinq + wn$
$s[1] = a[1] = 0$
▷ Initialization of the first element in the vectors
for $i = 2$ to $Tmax$ **do**
$x(i) = x(i-1) + v(i-1)dt + \frac{1}{2}a(i-1)dt^2$
$v(i) = vinq + wn$
$a(i) = \frac{v(i)-v(i-1)}{dt}$
▷ In this trajectory we consider the acceleration equal to 0 then, as consequence, we have a constant speed
end for