Master of Science in Informatics at Grenoble
Master Mathématiques Informatique - spécialité Informatique
option Graphics Vision and Robotics

Learning Based Approach for Online Lane Change Intention Prediction

Puneet KUMAR

20^{th} June, 2012

Research project performed at E-Motion Team, Inria, France

Under the supervision of:
Dr. Christian LAUGIER (christian.laugier@inria.fr), Inria
Dr. Mathias PERROLLAZ (mathias.perrollaz@inria.fr), Inria
Stephanie LEFEVRE (stephanie.lefevre@inria.fr), Inria

Defended before a jury composed of:
Prof. James CROWLEY
Prof. Remi RONFARD
Prof. Marie-Christine FAUVET
Dr. Alexandros MAKRIS

June 2012

Abstract

Predicting driver's behaviors is a key component for future Advanced Driver Assistance Systems (ADAS). In this thesis, a novel approach based on Support Vector Machine (SVM) and Bayesian filter (BF) is proposed for online lane change intention prediction. The approach uses the multiclass probabilistic outputs of the Support Vector Machine as an input to the Bayesian filter, and the posterior output of the Bayesian filter is used for the final prediction of lane changes. A particle filter based lane tracker integrated with a Lexus experimental platform is used for real-world data acquisition for the purpose of training and testing. Data from different drivers on different highways were used for the robustness evaluation of the overall approach. In-depth analysis of the proposed approach is also done and presented. The proposed approach is able to predict driver's intention to change lane 1.3 seconds (average) earlier with maximum prediction horizon of 3.29 seconds.

Résumé

La prédiction de l'intention des conducteurs ouvre la voie à de nouveaux systèmes d'aide à la conduite. Dans ce contexte, ce document propose une nouvelle approche de prédiction de l'intention de changement de voies, basée sur l'utilisation combinée d'un SVM (Support Vector Machine) et d'un filtre Bayésien. La distribution de probabilités sur les trois classes possibles à la sortie du SVM (changement à droite, changement à gauche, pas de changement) est utilisée en entrée d'un filtre Bayésien qui assure la cohérence temporelle de l'estimation. La sortie du filtre fournit la prédiction finale. Ce rapport présente une étude en profondeur de la méthode proposée, ainsi que son évaluation expérimentale. Les tests et la validation sont réalisés à l'aide de données routières réelles acquises par un véhicule instrumenté. Un algorithme de suivi de marquages routiers, basé sur un filtre particulaire, est utilisé pour estimer le positionnement relatif du véhicule par rapport à la voie, cette information étant utilisée par le système de prédiction d'intention. Les premiers résultats sont encourageants puisque la manoeuvre est prédite en moyenne 1,3 secondes en avance, et jusqu'à 3,29 secondes. D'autre part, la robustesse est validée en considérant des conducteurs différents.

Contents

— 1 —
Introduction

1.1 Motivation

Imagine a world with no road accidents. Every year millions of road accidents are witnessed costing millions of human lives and huge economic loss. A study by World Health Organization [19] shows that almost 1.2 million people died globally as a result of road accidents in the year 2000. Another study by [11], as shown in figure 1.1, found that out of three major factors causing road accidents such as: driver factors, vehicle factors, and roadway factors, 57 % of the road accidents are solely due to the driver factors. Therefore, we can say that road accidents are serious global issue caused mainly due to driver's inability to understand the situation and the surroundings in a correct way so as to take right decisions at the right time to avoid road accidents. It is not possible to train all the drivers in the world in a way to decrease the driver factors of road accidents. Therefore, ADAS (Advanced Driving Assistance System) can help drivers to understand the traffic situation in a better way, assist or take necessary actions to make driving comfortable, improve the traffic flow, limit energy consumption, and avoid accidents or mitigate their consequences. For example, predicting in advance the behavior of a vehicle at intersection or predicting whether a vehicle on highway will go for a lane change or not can help a driver to understand the situation in a better way to avoid accidents. Such perception and understanding can further be integrated with other path planning or collision avoidance systems to make futuristic autonomous vehicles that can replace a driver and drive on their own instead of assisting a driver.

This thesis is situated within the context of **future Advanced Driving Assistance System** with prime focus on discussing and developing algorithms for **intention prediction** in real time using Machine Learning techniques. Further sections of this chapter discusses different technologies involved in ADAS, give better description to the problem addressed in this thesis and brief about the contributions made.

1.2 Background

Since the 80's, European Commission, DARPA and other organizations worldwide have been funding projects on developing intelligent vehicles and we have witnessed many advancements in this field. Nowadays many big automotive companies e.g. BMW, Volkswagen, Toyota

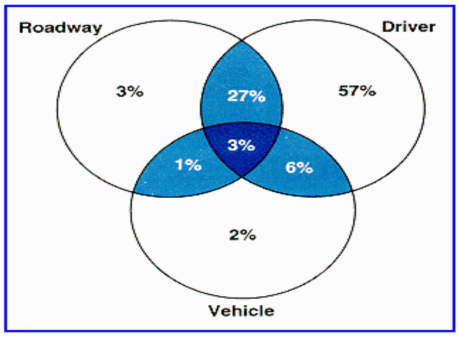


Figure 1.1 – Different Factors on road accidents

(a) Stanford's Car (b) CMU's Car (c) MIT's Car

Figure 1.2 – Autonomous Vehicles participated to DARPA challenge

etc. have started to test their driverless car systems. The autonomous cars demonstrated by many teams in DARPA Urban challenge (2007) show what our future cars could look like (Figure 1.2). Recently, on June 2011, Nevada Legislature in USA passed the first law in the world to authorize the use of autonomous cars on road, another big milestone in the future of AGVs and presents an example of the level of advancements that has been achieved in this field. Although, we still don't have a fully autonomous vehicle capable of fulfilling real time human needs, Advanced Driving Assistance Systems (ADASs) comes to rescue and bring us closer to our needs. ADASs are now available on many advanced cars for public use and has witnessed great advancements in recent years because of the advancements in computational capabilities and sensor technologies. The purpose of ADAS is to assist drivers driving the car in order to make their driving easier and safer.

1.3 Advanced Driving Assistance Systems

As stated earlier, the purpose of ADAS is to assist human drivers driving the car in order to make driving safer and easier. In other words, these automated systems take over partly or entirely the driving task to reduce the mental and physical workload, and to compensate for limitations of the drivers. To this end, an ADAS has to perceive the environment, analyze it, and then either inform the driver or warn the driver or take necessary actions.

1.3.1 Existing ADAS

Existing ADASs are very useful in driving scenarios and are used worldwide, but most of these systems doesn't have full control over the driving of the car and therefore can be overruled by the human drivers. There are many ADASs available in the market for personal use. Here are examples of some of the most important existing advanced driving assistance systems.

Adaptive Cruise Control (ACC)

ACC is an extension of a conventional cruise control system that uses radar or laser sensors to perceive the distance of the vehicle in front of it in the same lane in order to automatically accelerate or decelerate to maintain a safe distance from the vehicle. It ensures that the gap between the cars is sufficient to avoid accidents. ACC is also known as longitudnal support system because it concerns the driving task in the forward direction. The functionalities of ACC can be splitted into three major parts: the ACC controller that computes how the vehicle should accelerate or deaccelerate, the longitudinal control that manages the actuator systems to achieve the desired acceleration or deacceleration computed by the ACC controller, and the human-machine interface enabling the driver to operate, supervise and reclaim control when necessary [24].

Automatic Parking System (APS)

The purpose of APS is to enable vehicles to drive into the parking places autonomously. APS was first experimentally demonstrated in the mid 90's by INRIA [21] using the concept of Localization-Planning-Execution cycle to reach a specified location of the car relative to its environment. Nowadays, various companies are working on integrating the garages with the required technology and sensors to communicate with the APS enabled cars for the purpose of guidance, planning, and control. These systems are of great demand as they save time and space by optimized placement of the cars in the garages.

Blind Spot Information System (BLIS)

Blind spot is an area in the vicinity of the vehicle that cannot be directly observed by the driver. To observe this area a driver has to make extra efforts e.g. turning head, adjusting mirrors etc. Sometimes this area is not at all visible. The BLIS is a sensor or camera-based system that provides visual information to the driver about any object (vehicles, pedestrians, or cyclists) in vehicle's blind spot which in turn helps the driver to avoid accidents. This system is also considered as a lane change assistant (LCA) system.

Lane Departure Warning (LDW)

LDW is a camera, laser, and infrared sensors based system that can recognize lane markings and can warn a driver using visual, audio or vibrational warnings when a driver begins to leave its lane without using the turning signal. The other form of LDW, known as Lane Keeping System (LKS), has higher level of automation and can automatically take steps to ensure that the vehicle stays in its lane. These systems are highly useful in highway scenarios and can avoid many accidents.

Collison Avoidance System (CAS)

The Collision Avoidance System is an active safety system, which works in the background and tries to prevent accidents whenever the situation becomes too critical. To avoid a collision, it can override the driver's actions and take control of the lateral and longitudinal maneuvering of the vehicle [24]. These systems use database of information about the weather and traffic conditions along with sensors to better understand the situation.

1.3.2 Future ADAS

Future ADAS refer to more sophisticated, automated, and advanced ADASs that can understand and assess the traffic scenarios in a much better way to avoid accidents and make driving safer and comfortable. They are supposed to address the limitations of existing ADAS. Categorizing future ADAS completely at this stage will be difficult because we still don't have a clear picture of how a future ADAS will look like, but we can talk about some of the technologies that it will be integrated with. A future ADAS can have capabilities of 3D maps for planning, risk assessment for emergency braking, driver monitoring systems to monitor the state of a driver, lane change prediction system to predict in advance whether a vehicle will take a lane change or not, generate future trajectories of other vehicles on road, capabilities to override drivers control etc. Other features available with these technologies may be the availability of vehicle-to-vehicle (V2V) and vehicle-to-infrastructure (V2I) communication, precise geographic information system (GIS) for every road network, highly accurate GPS, much advanced and precise sensors, good processing power, communication of vehicles with live weather conditions, etc. There can be many other capabilities associated with future ADASs.

1.4 Problem Statement

This thesis is focused on discussing and developing algorithms for **lane change intention prediction** on highways in real situation. Lane change intention system is part of future ADAS or future autonomous vehicles. The problem is to predict in advance whether a vehicle under consideration will make a lane change or not. If yes, then which lane change, left lane change or right lane change. In this work, a complete lane change maneuver is defined as the segment of a vehicle trajectory that intersects with the curve that defines the lane markings and the instant of lane change is the time at which the vehicle trajectory and the lane curve intersect as depicted in figure 1.3. Mathematically, a vehicle trajectory can be defined by the deterministic function:

$$\Phi : \Phi(t) = \left(x(t), y(t), \frac{dx}{dt}(t), \frac{dy}{dt}(t), \frac{d^2x}{d^2t}(t), \frac{d^2y}{d^2t}(t) \right) \tag{1.1}$$

Where the parameters of the function Φ represent the longitudinal position, lateral position, longitudinal velocity, lateral velocity, longitudinal acceleration and lateral acceleration of the vehicle respectively [1]. If we define a lane curve as the function $\Theta(x,y)$, then the trajectory

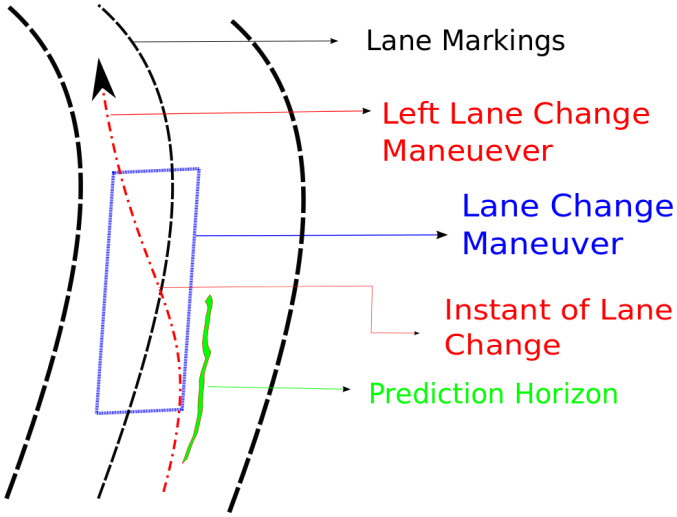


Figure 1.3 – Lane Change Scenario

segment during the time $[t_1, t_2]$ is a lane change trajectory segment, iff:

$$\exists t_0 \in [t_1, t_2] : \Theta(x_{t_0}, y_{t_0}) = \Phi\left(x(t_0), y(t_0), \frac{dx}{dt}(t_0), \frac{dy}{dt}(t_0), \frac{d^2x}{d^2t}(t_0), \frac{d^2y}{d^2t}(t_0)\right) \tag{1.2}$$

In this context, the problem is to:

- Predict the lane change intention at time $t < t_0$, i.e. predict lane change intention before it actually happens and also increase the prediction horizon.
- Differentiate a left lane change from a right lane change.
- Use real dataset for the evaluation purpose.
- Minimize false alarms (false prediction of lane changes) resulting due to the limitations of the sensors (camera, odometer), algorithms, and constraints of the real time scenario.

1.5 Contributions

Following are the contributions of this thesis:

1. Proposed a new approach based on the combination of multiclass SVM probability estimate and Bayesian filter to address the problem of lane change intention prediction for real-world data.
2. The approach is able to predict lane change intentions in advance in all the situations tested in the real dataset collected using a Lexus experimental platform. The average precision of the approach is 0.7154 and recall is 1 when tested on 69 lane changes.
3. The prediction horizon in this approach is up to 3.29 seconds with average prediction time of 1.3 seconds.
4. The approach is able to differentiate between left lane change and right lane change.

1.6 Thesis Outline

Chapter 2 reviews the related work, chapter 3 discusses the different aspects and constraints of the problem and then proposes an approach to solve the lane change prediction problem within the discussed constraints. Chapter 4 talks about the implementation details of the proposed approach and chapter 5 presents the results along with an in-depth analysis of the approach. Chapter 6 concludes and outlines future work.

— 2 —
State of the art

2.1 Introduction

This thesis is focused on developing an algorithm for driver intention prediction for Advanced Driving Assistance System (ADAS) using discriminative learning based approach. Before moving further to develop the algorithm, this chapter is being introduced to give an insight about the current research advancements in the fields related to the problem addressed in this thesis.

The term **Intention prediction** itself has different interpretations and has been addressed by different researchers worldwide in different ways. Different terminologies such as: **behavior prediction**, **situation assessment or prediction**, **intention prediction** etc., represents the same type of problems. The terminologies varies from problem to problem and from researchers to researchers based on their interpretations but the core of the approach to the solution of these problems is almost the same. Different examples of these problems are: predicting whether a driver will take a lane change or not, predicting whether a vehicle is compliant or violent, predicting whether a vehicle under consideration will apply break or not, etc.

All these problems of ADAS are mainly addressed using machine learning techniques combined with some filtering or rule based techniques. Based on different works by different researchers and the scope of this thesis, there are mainly three categories in which the work of intention or behavior or situation prediction can be divided: Discriminative learning based approaches, Generative learning based approaches and Other approaches.

2.2 Discriminative learning based approaches

Discriminative approaches classify driver intention by direct mapping between input and output without explicitly modelling the underlying distribution of the variables. The relationship between variables is not visualizable in this case. Some of the examples of discriminative learning based techniques are Support Vector Machines (SVMs), Logistic regression, Neural Networks, Regularization Networks etc.

A popular work on lane change detection was proposed by [12]. The problem is addressed using a discriminative classifier, known as Support Vector Machine (SVM), to classify lane changes. The feature vector in this approach is formed using the variances of the features (speed, steering angle etc.) to form an input vector in a short duration of time. Variance is used

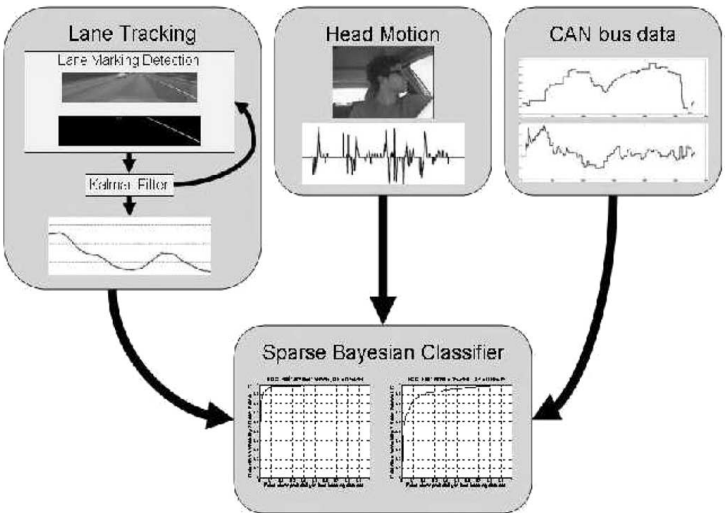


Figure 2.1 – Lane change intent analysis system flow chart proposed by [13]

because it captures changes which is critical to learn specific pattern. To find variances, overlapping and non-overlapping subwindow approaches are used. In both approaches, subwindows are created and variance is calculated for each subwindow of the input vector. It finally is concatenated to form the input feature vector of SVM for training and testing purposes. Authors tried many combinations of feature sets, window sizes, and overlapping vs. non-overlapping approaches to evaluate their proposed approach and claimed to get best results with 97.9% recognition accuracy with overlapping approach when used with features set consisting of all the lane positions. They also claim to get detected 87% of all true positives within 0.3 seconds of the start of the maneuver.

In [13], a sparse Bayesian classifier based discriminative learning approach for lane change intent analysis was proposed using lane position estimator, vehicle parameter estimator and driver head motion information. The flow chart of this approach is shown in figure 2.1. The idea here is to use time series data describing vehicle's surrounding, the driver's head motion and vehicle's internal state to create a feture vector for classification. Vehicle's surrounding in case of lane change detection is mainly consisted of lane information. The lane tracking system was used to get the lane information. This system mainly uses Steerable filter based lane detection, parabolic model of the road, and Kalman filter based lane tracking system. The Kalman filter uses lane detection information along with steering angle and wheel velocity from the CAN bus of the vehicle to update the state vector consisted of lane position, lane heading, lane curvature, lane width etc. The basic idea behind the head motion estimator used by the authors is to find the interframe head motion to construct the motion vector to identify head movements in order to find out if the driver is looking at the mirror or the surrounding while performing a lane change. On top of this a Sparse Bayesian classifier is used to classify lane change intentions. The results of this work showed that the inclusion of the driver's state (head motion) improves the prediction horizon.

The extension of the work by [13] proposed by [15] uses Relevance Vector Machine (RVM), a Bayesian extension of SVM, based discriminative classifier for lane change prediction. The framework used by the authors is shown in figure 2.2. It used ACC (Adaptive Cruise Control

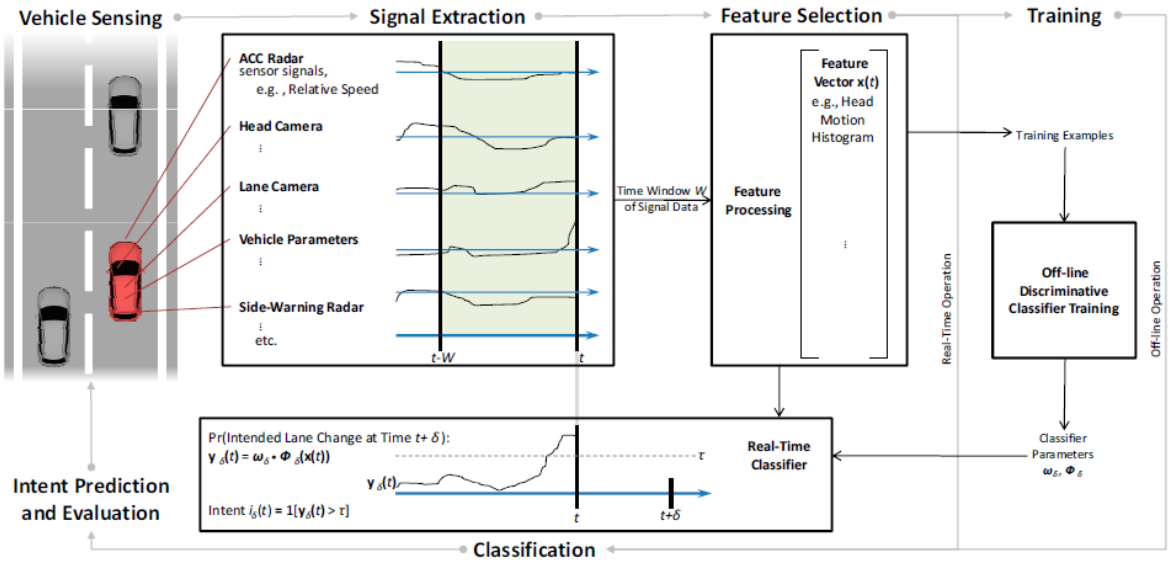


Figure 2.2 – Overview of Real-time Intent Detection System by [15]

radar), LDW (Lane Departure Warning Camera), SWA (Side Warning Assist) radars, and head tracking camera for the purpose of data collection. The dataset collected for the training is used offline to train a discriminative classifier (RVM) to get the trained model and this trained model is then used in real time to generate a lane change probability score. The feature vector constructed is of approximately 500 dimensions consisting of time series sensor data from the vehicle. The results of the proposed approach showed 80 percent detection rate but many false positives. To overcome false positives a multi-suppression technique is used which considers the fact that consecutive detection arises from the same intention. This is valid in a sense that the data is generated using sliding window approach, so a large set of data represents the same intention and each data point is very similar to its successive data point so the classification result should be similar until there is remarkable change in the input data which occurs when there is change in intention. This approach was able to detect lane changes upto 3 seconds in advance from the start of the maneuver mainly due to the head tracking camera.

Another work by [3] dealt with intentions at road intersections and considered two types of on-road agents: dangerous and harmless. The idea is to find whether a vehicle under consideration, called ”suspicious” vehicle, is harmless or dangerous for the ”host” vehicle. The approach in this work uses discriminative binary classification along with Bayesian filtering (BF) to classify agent intentions. The classifier used is Support Vector Machine (SVM) with radial basis kernel function. The output of the classifier is fed to a Bayesian filter with prior as the beta distribution over the probabilities of an agent being harmless and a binomial likelihood function. Finally a threshold detector is used to get the final classification result. A discounted factor is also used at the update step of Bayesian filter, called discounted BF, in order to speed up the convergence. The features used in this approach to train and test the SVM classifier are the relative distance, the heading of the suspicious vehicle relative to the host vehicle and the speed of the vehicle. This approach was tested in a simulated environment and the authors claimed to achieve 100% coverage of and 77% precision without discounted BF and 93% coverage of and

90% precision with discounted BF.

Reference [3], discussed above, was further extended and tested on real time data in [4]. This work focused on ”behavior” classification instead of ”intention” classification and the behaviors of a driver in this work is categorized as ”compliant” or ”violating”. The objective is to find whether a driver will stop before the stop bar if the traffic signal indicates to do so or not. Authors presented both discriminative and generative classification approaches to address this problem. The discriminative approach uses SVM-BF architecture as discussed above [3] and the generative approach uses Hidden Markov Models (HMMs) with the Expectation-Maximization (EM) algorithm to develop two distinct HMMs for compliant and violating behaviors. Authors claimed to get better results than the traditional methods when compared with SVM-BF and HMM based approaches. Also SVM-BF based approach was the top performer algorithm in all the tests so is suggested to be suitable for real applications.

Another recent work on future behavior prediction was proposed by [20] for an inner-city traffic condition. This work focuses on braking behavior of vehicles on straight road. The problem is addressed using the concept of multiclass learning in low dimensional representation of the current situation and predicted behavior, and uses both the dynamic and the static information of the vehicle and the traffic light respectively. The driver behavior is described using a set of elementary behaviors named: ”stopped”, ”braking” and ”other”. These elementary behaviors are called ”behavior primitives” and are decomposed using heuristics which use speed, gas pedal, and break pedal information. A Multi-Layer Perceptron (MLP) or a simple neural model is being used for the learning purpose having three behavior primitives as the outputs. As a result of this work authors claimed to get a prediction horizon *upto* 6 seconds which is a highly significant result when compared with other approaches with prediction horizon upto 2 seconds.

2.3 Generative learning based approaches

A generative learning based approach model all the underlying variables and manipulate them for classification. Input and output are represented by joint distribution and this joint distribution is conditioned for the purpose of classification. Some of the examples of Generative methods are Gaussian, Hidden Markov Model (HMM), Bayesian Networks, Markov Random Fields etc.

In [18], a graphical model, Hidden Markov Model (HMM) and Coupled HMMs were trained using experimental data to create models of seven different driving maneuvers. The results showed, on average, prediction of maneuvers 1 second before it actually started. Another work by [9] used simple Dynamic Bayesian Network or HMMs for predicting driving behavior in terms of future stop probability of a vehicle at an intersection using present and past observable data. Various factors causing changes in drivers intention were modeled using state transition probabilities and sequential inference through DBNs was used for prediction. The results of this approach showed prediction of future stop probability several seconds before its occurence and also found that drivers behaves according to certain habits while changing lanes.

[14] proposed a probabilistic model based approach using HMMs for situation modeling and recognition. A situation in this work is defined as a distrubution over sequences of states

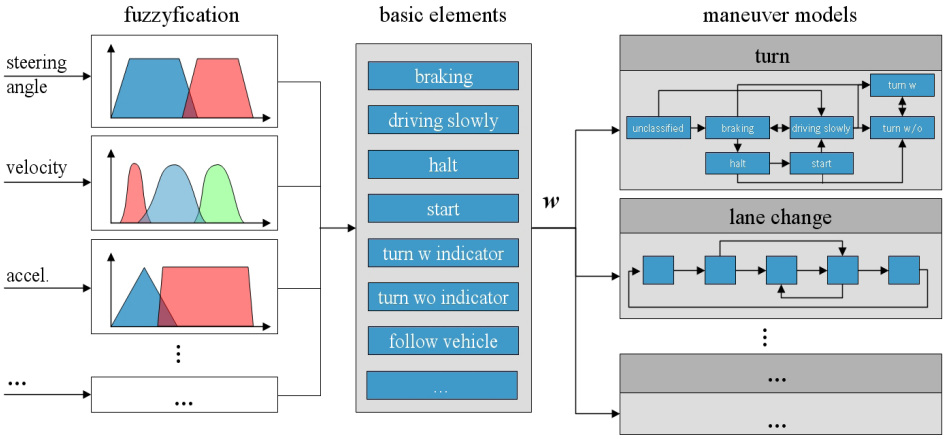


Figure 2.3 – Overall structure of the system proposed by [8]

having meaningful interpretation and a HMM is used to characterize each situation. The approach was tested on real data in a highway scenario with three situations as: passing, aborted passing, and following.

In [25], a layered or hierarchical HHMMs based generative approach has been used for behavior recognition in a normal traffic conditions. The behaviors in this work are defined as: move straight, left turn, right turn and overtake. The idea is to model behaviors in two layers with each layer consisting of one or more HMMs. This layered approach is similar to that of [14]. High level behaviors, such as: move straight, left turn, right turn and overtake are treated as the hidden states of single HMM at the upper layer. For each hidden state or behavior at the upper layer there exists a HMM at the lower layer representing sequence of transitions of the corresponding hidden state. These sequence of transitions basically represents the semantics to realize each hidden state or behavior. The upper layer HMM update gives the probability distributions over hidden states or behaviors which is further used for behavior recognition.

In [8], a Probabilistic Finite State Machine (PFSM) and fuzzy logic is used for maneuver recognition. The overall structure of the system is given in Figure 2.3. The input variables (velocity, steering angle etc.) are fuzzified using fuzzy rules and corresponding membership functions in order to estimate the basic elements (braking, halt, start etc.) constituting maneuvers. These basic elements represents the nodes of the state transition diagram representing PFSM as a directed graph which basically represents the possible sequences of basic elements of several driving maneuvers. Finally a Bayesian filter is used to find the probability distribution of the basic elements of PFSM for maneuver recognition. This approach is similar to the approach by [25] in which the lower layer of HHMM represents the semantics that form the basis for higher level behaviors. These semantics are similar to the fuzzified basic elements used in this approach. This approach is also similar to the work by [20] where behavior primitives are similar to the fuzzified basic elements in this approach. The major drawback of this approach is that the fuzzy rules are designed manually, therefore, depends completely on the designers skills and his understanding of the problem. Another issue is the breaking of the problem into such a set of basic elements that is sufficient to form effective fuzzy rules.

2.4 Other Approaches

[22] introduced a mind tracking architecture based on cognitive model of driver behavior implemented in a cognitive architecture. This approach was further improved and presented by [23]. The idea is to find the similarity between the driver's actual observed behavior and several simulated driver behaviors created using the cognitive model to infer driver's unobserved intentions.

— 3 —
Proposed Approach and Discussion

As stated in Chapter 1, the aim of this thesis is to focus on lane change intention prediction for an advanced driving assistance system using a combination of multiclass probabilistic output from a kernel based discriminative classifier known as SVM, and a Bayesian Filter (BF). The following sections of this chapter give an insight on why a kernel based SVM, the probabilistic output, and a Bayesian filter have been used to address this problem. This chapter also discusses about the experimental platform used for training and testing purposes.

3.1 Addressing the problem

If we look at the problem of lane change intention prediction a bit closely then we find that if we don't have access to the thinking or the psychology of the driver, if we can't model someone's brain activities, and if we can't control the complete traffic scenario, then the problem of lane change intention prediction becomes the problem of understanding, modeling and learning the families or classes of driving patterns or trajectories related to the lane changes and then trying to find out in which class a particular pattern belongs to by looking at some initial parts of it. This is not as simple as it seems because general driving patterns are highly stochastic and depend on many internal and external parameters of the driver and the surrounding environment. Also the sensors used to perceive the environment have their own limitations in terms of precision, speed, and reliability. For example, if the driving car has very good braking system and good maneuverability then the driving patterns changes even if the driver is the same. Also if there are a lot of vehicles on road, then the driving pattern of the vehicle changes based on the driving patterns of the other dynamic vehicles in front it. Therefore, it becomes a highly complex multi agent pattern prediction problem with many internal and external controlling parameters including the psychology of the drivers, and the surrounding.

We have seen in Chapter 2 that there are some approaches (e.g. [13]) in which either information about the drivers activities such as head motion along with other information is used or authors have tried to model driver's behavior using a cognitive architecture (e.g. [22],[23]). The approach in this thesis doesn't use internal or external information about the state of the driver. This thesis is mainly focused on predicting the lane changes based on the external parameters such as lane information, speed, steering angle etc. of the vehicle on highway scenarios and each agent on road is considered to be independent to each other. In this context, the problem can be rewritten as:

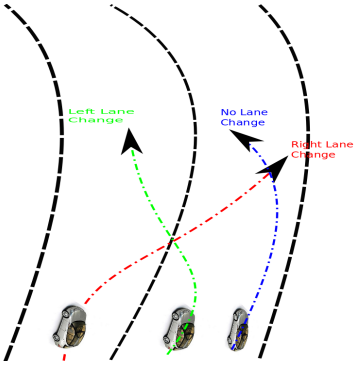


Figure 3.1 – Three lane change situations on highways

- **Problem Statement**: Predict lane change intention before it is actually happened.
- **Scenario**: Highway scenario with availability of external parameters (such as lane information, speed etc.) using real sensors (camera and odometers).
- **Hypothesis**: Assuming each agent on road to be independent to each other.

3.1.1 Why Kernel based SVM as a discriminative classifier?

The problem formulated above can be expressed as a multiclass classification problem with three classes as: left lane change, right lane change, and no lane change, representing three families of trajectories for each class as shown in figure 3.1. Other situations like multiple lane changes can be seen as the combinations of these three classes. So we have a notion of classes and this problem, therefore, can be seen as a multiclass classification problem in which each class represents a family of trajectories w.r.t. the road geometry.

Many classifiers such as: Logistic regression, Support Vector Machine, Naive Bayes etc. are available and choosing a classifier is generally a very important step for a specific problem. A kernel based large margin classifier known as Support Vector Machine (SVM) has been used to address this problem. Motivations behind using kernel based SVM are:

- a driver's state may lie in a high dimensional feature space [27] and a kernel maps the input data from a low dimensional space into a high dimensional space converting a nonlinear classification problem at low dimension into a linear classification problem at high dimension.
- our feature vectors are high dimensional ($\simeq 128 - 400$) and kernels can easily handle very high dimensional feature vectors.
- SVM is a maximum margin classifier, therefore, is expected to classify similar trajectories belonging to different classes more precisely.
- SVM's objective function is convex, therefore the solution is global optima.
- different studies (e.g. [2]) have shown that SVM gives better and highly promising results compared to a generative approach known as Hidden Markov Model when applied to behavior prediction problem for ADAS.
- a lane change can also be seen as a time series process and SVM has shown excellent results when applied to time series prediction problems [16].

Figure 3.2 – The Lexus experimental platform for real data generation

3.1.2 Why Multiclass Probability Estimates and Bayesian Filter?

Considering the limitations of sensors, algorithms, and real-world scenarios, many lane change false alarms are expected if only an SVM is used directly for the final decision making. This has been seen in our experimentation as discussed in Chapter 5. False alarms can be distracting, annoying, and very dangerous in real time. To address this problem of false alarms a Bayesian filter (BF) has been used on top of the multiclass classifier. Since a BF takes probabilistic inputs as likelihood and prior, a generalized Bradley-Terry model based Multiclass Probability Estimates of SVM [10] is used to obtain probabilistic outputs from the SVM. The BF takes the probabilistic output of the SVM as the likelihood input and the learned state transition matrix as the prior to find the posterior over all the possible classes. This posterior is finally used for the decision making.

3.2 Experimental platform for real data generation

Finally, to implement the proposed approach a highly realistic dataset and an experimental platform is required for the offline training of the SVM classifier along with the state transition matrix for Bayesian filter, and for the testing of the overall approach. The experimental platform used in this thesis is a Toyota Lexus car, shown in figure 3.2, equipped with a stereo vision camera, odometers and a powerful processor. This platform also has vision and IMU based real time lane tracking capability integrated with it. The data generated from the lane tracker has been used as an input to our lane change prediction system. A brief technical details about the lane tracker is given in Chapter 4.1.

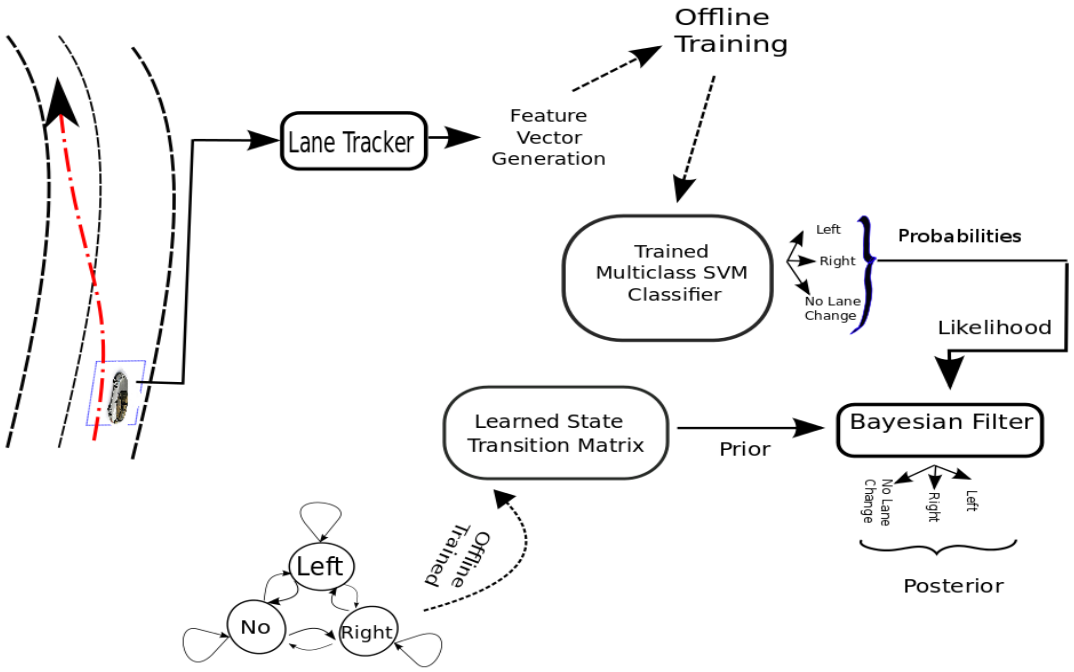


Figure 3.3 – Overall Approach

3.3 Why Vision and IMU based lane tracker?

In section 4.2, details about the designing of a feature vector for the SVM is discussed. The main components of the feature vectors are the position of the vehicle w.r.t. lane markings and the heading angle of the vehicle w.r.t. road curvature. There are many ways to get this information. One way is to put a precise GPS on the vehicle, use highly precise road network information along with information about the lane markings using Geographic Information System (GIS) and use a very precise Inertial Measurement Unit (IMU). IMU fused with GPS can be used to get an accurate position, GPS can be used to find latitude, longitude, and height which in turn can be used with GIS to find the position of the vehicle w.r.t. the lane markings, and can be used to convert the heading angle of the vehicle w.r.t. the road curvature. This is a simple way to gather data and implement the proposed approach but the major problem with this kind of system is that the publicly available GPS accuracy is not high enough to be used for this purpose and GPS is not available everywhere. In addition, a precise GIS is very costly so such systems are not realistic for mass production.

Therefore, to make the system independent of GPS and GIS so as to increase the reliability and decrease the cost, a vision and IMU based lane tracker system integrated with the Lexus experimental platform has been used to implement the proposed approach.

3.4 Overall Approach

The overall approach as discussed in previous sections is shown in figure 3.3. There are three major steps used to address the lane change intention prediction problem in this such as:

data generation using lane tracker, multiclass probabilistic estimates using SVM, and Bayesian filter for removing false alarms. There are many other issues to be handled while using SVM and BF, such as: designing feature vectors, choosing kernels, learning the state transition matrix, likelihood etc. Details about handling these issues is explained in the next chapter.

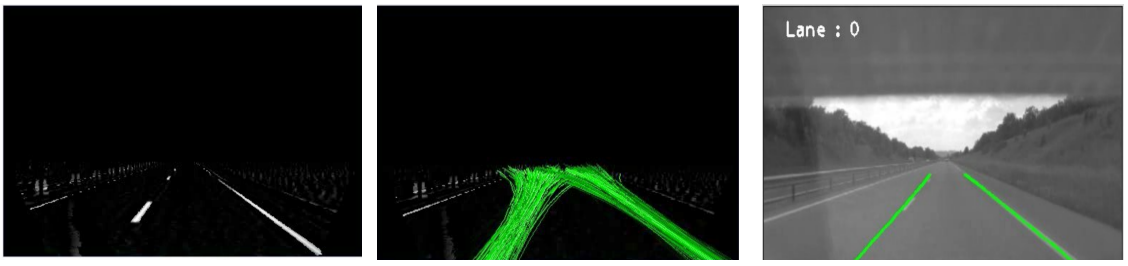
— 4 —
Implementation

Chapter 3 gave the overall approach to address the problem of lane change intention prediction. The basic three steps used are: data generation using the lane tracker, probabilistic classification using SVM, and Bayesian filtering. This chapter gives details about the implementation methodology and related issues with each steps. Finally, the detailed overall approach is shown at the end of this chapter.

4.1 The Lane Tracker

As stated earlier, the Lexus experimental platform integrated with the lane tracker is used for the real data generation for training and testing of the proposed approach. In general, the lane tracking problem is the integrated form of tracking and lane detection problems. It can be seen as a state estimation problem with state vector consisting of the lane parameters. These parameters defines the road geometry in terms of the lane positions with respect to the vehicle [6]. The estimation can be done using particle filter or any other recursive estimation technique. The lane tracker used in this thesis considers road to be flat and lane markings to be continuous and parallel. This assumption speeds up the tracking process. The output of the lane tracker is described in section 4.2.1. Briefly, the lane tracker steps used in this thesis are:

1. **Define lane position and geometry**: Parametric equations describing the lane positions and the road geometry is formulated.
2. **State vector prediction using Particle filter**: The parameters of the parametric equations described in the previous step are used to form a state vector. This state vector is recursively estimated using the past state vector and the odometer information (steering angle and speed). The updated state vector represents the current parameters defining the current lane positions and the road geometry.
3. **Grey-scale image capturing and ridge detection**: Ridges are detected to get an idea of the position of the lanes in the image space (see figure 4.1a).
4. **Horizon estimation**: Horizon is estimated using the estimated pitch angle of the mounted camera. The pitch is estimated along with other parameters in step 2. This reduces the image search space because the lane markings can't stay above the horizon.
5. **Voting scheme for finding the best particle**: Since particle filter is used for the estimation so we have many estimated state vectors, each state vector for each particle. A voting



(a) Ridge Detection (b) Lanes due to many particles (c) Lane due to highest voted particles

Figure 4.1 – Lane Tracker Outputs

scheme is used to find the particles nearest to the ridge. The particles nearest to the ridge are then chosen and the corresponding state vector is used as the predicted state vector describing the lane positions and road geometry (see figures figures 4.1b, 4.1c).

4.2 Multiclass Probability Estimates using SVM

As discussed in Chapter 3, the lane change intention prediction problem can be seen as a multiclass classification problem with three different classes: left lane change, right lane change, and no lane change. Generally an SVM generates the class number to which a particular input data belongs. To get the probabilistic estimates as the output of SVM, a generalized Bradley-Terry model based multiclass probability estimates proposed by [10] is used. Further subsections give brief about SVM and explains the implementation methodology.

4.2.1 Support Vector Machine

Based on statistical learning theory, Support Vector Machine (SVM) is a supervised learning algorithm that uses the concept of margin maximization for the purpose of classification. The basic idea is to find a hyperplane that maximizes the separation between the datapoints of different classes. It uses the concept of kernels to project data from low dimensions to higher dimensions thus converting a nonlinear problem in low dimensions into a linear problem in higher dimensions. Below, a brief description about the mathematics behind the SVM for binary classification is given [5], [26], [17], and the idea of it's multiclass extension is also discussed in brief.

Let $S = \{(x_1, y_1), ..., (x_m, y_m)\} \subseteq \chi \times \gamma$ denote a set of m training examples with x_i as the training example and y_i the corresponding label, where $\chi \in \Re^n$ and $\gamma \in \{-1, 1\}$. Then the purpose of SVM is to minimize an objective function in order to find a hyperplane $< W, b >$ that maximizes the margin between datapoints of both classes. Here W is a vector representing the parameters of the hyperplane and b is a scalar representing the intercept term. The optimization problem of SVM is:

$$min_{W,\zeta \geq 0,b} \frac{1}{2} W^T W + \frac{C}{m} \sum_{i=1}^{m} \zeta_i \tag{4.1}$$

$$s.t. y_1(W^T x_1 + b) \geq 1 - \zeta_1 \tag{4.2}$$

$$\ldots y_m(W^T x_m + b) \geq 1 - \zeta_m \tag{4.3}$$

Here C is the **regularization parameter** and ζ is the **slack variable**. The regularization parameter C controls the twin goals of maximizing margin and ensuring that most of the examples have functional margin of atleast 1. The slack variable reforms the hard margin SVM classifier into soft margin classifier to allow errors in the training dataset. The above optimization problem contains convex objective function with linear constraints and can be solved using quadratic programming (QP). This objective function is further represented in its dual form to allow us to use kernels to work efficiently in very high dimensional space. Dual optimization representation of the above SVM optimization function is:

$$max_\alpha \sum_{i=1}^{m} \alpha_i - \sum_{i=1}^{m} \sum_{j=1}^{m} y_i y_j \alpha_i \alpha_j (x_j \cdot x_i) \tag{4.4}$$

$$s.t. \sum_{i=1}^{m} y_i \alpha_i = 0 \tag{4.5}$$

$$\forall_{i=1}^{m} : 0 \leq \alpha_i \leq \frac{C}{n} \tag{4.6}$$

Here α_i are the Lagrange multiplier and $(x_j \cdot x_i)$ is the dot product between x_j and x_i. This dual form of the SVM is solved using SMO (Sequential Minimal Optimization) algorithm and the obtained α_i are used to find the optimized value W^*:

$$W^* = \sum_{i=1}^{m} \alpha_i^* y_i x_i \tag{4.7}$$

Using W^*, finding b^* is straightforward. W^* is further used in the decision function to find if the given example x belongs to class $(y = 1)$ or not based on its value is positive or negative. The decision function is written as:

$$h(x) = sgn((W^*)^T x + b) \tag{4.8}$$

$$= sgn\{(\sum_{i=1}^{m} \alpha_i^* y_i x_i)^T x + b\} \tag{4.9}$$

$$= sgn \sum_{i=1}^{m} \alpha_i^* y_i (x_i \cdot x) + b \tag{4.10}$$

Here, sgn represents the sign of the decision function. If $h(x) > 0$, testing example x belongs to class $(y = 1)$, otherwise it belongs to the other class. In the dual form of SVM discussed earlier, $(x_i \cdot x)$ is the dot product of the new test vector x with the example x_i and can also be represented as $K(x_i, x) = \phi(x_i).\phi(x)$, where K is known as the **kernel function** and ϕ is the feature mapping (represents high dimensional feature space). A valid kernel (follows Mercer's theorem) enable us to calculate $K(x_i, x)$ in linear time without even explicitly finding $\phi(x_i)$ and $\phi(x)$ which takes $O(n^2)$ time. So kernels allow us to learn in high dimensional feature space given by ϕ in linear time, thus it is widely used by machine learning community.

In case of multiclass classification problem with k classes, the same idea is extended and k different hyperplane parameter vectors W_i are obtained for each class and the parameter vector that gives maximum value of the hypothesis $W_i^T x$ with the given example x gives the corresponding label. The standard steps used to effectively implement SVM for any related problems are [7]:

1. Data Collection for training and testing
2. Designing a suitable feature vector
3. Kernel selection (if required)
4. Find best parameters using cross-validation
5. Use these parameters for training and testing purposes

Each of these steps are explained in following sub-sections.

Data Collection

As discussed in the earlier section, data are collected using the experimental platform integrated with lane tracking capabilities and inertial sensors. The data from the experimental platform consists of:

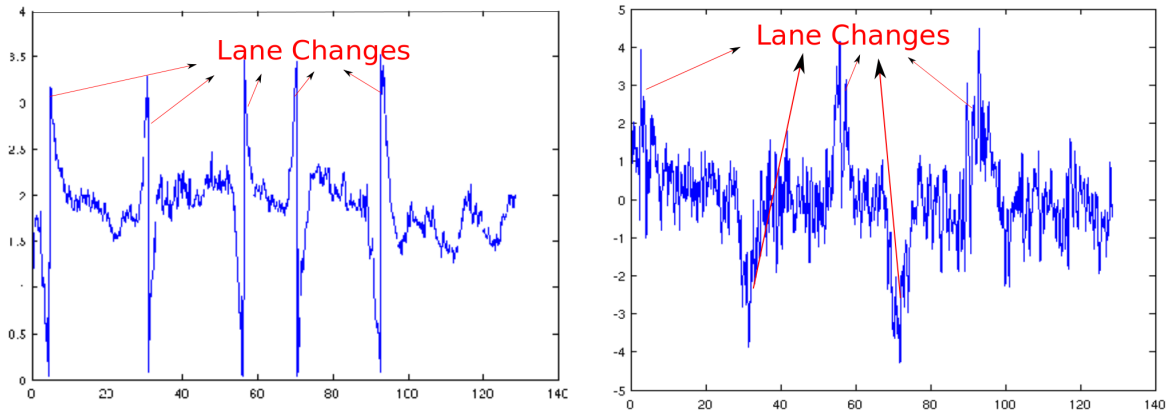
- **Data from the Lane Tracker (Estimated road model)**
  - Road width
  - Lateral position of the vehicle w.r.t. left lane marking
  - Vehicle steering angle
  - Camera pitch
  - Road curvature
  - Variation of road curvature
- **Data from the CAN bus**
  - Velocity
  - Acceleration
  - Steering Angle
  - Time stamp

This data has been collected at 32 frames per second in a log file for the training purpose. They were first converted into feature vectors for training and testing purpose as explained in the following section.

Designing a suitable feature vector

Designing a suitable feature vector is a very important step for any discriminative classifier because selection of features is directly related to the discriminative capabilities of the classifier. Although there is not any fixed theory behind designing or choosing feature vectors, we have to choose the best features based on the data available and our understanding of the problem and then try other meta-features (features formed using the combinations of existing features) also for validation.

Intuitively, the features most likely to discriminate different classes of lane changes are: **lateral position** of the vehicle w.r.t. left lane (x), and the **steering angle** of the vehicle w.r.t. road



(a) Lateral Position w.r.t. Time (b) Steering Angle w.r.t. Time

Figure 4.2 – Lateral position and Steering angle plot

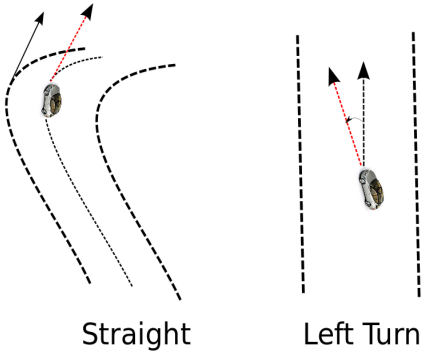
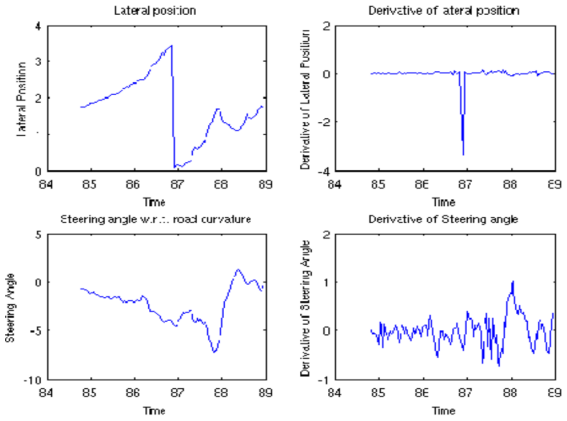


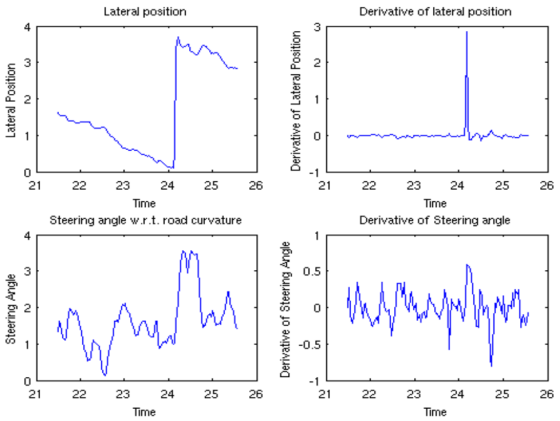
Figure 4.3 – Straight vs Turning (defined w.r.t. road geometry)

curvature (ϕ). To validate this intuition, (x) and (ϕ) are plotted using real time data captured from the experimental platform (see figure 4.2). Figure 4.2a shows that during left lane change, x first decreases, reaches to zero and then just after the lane change reaches it's maximum and then starts decreasing until it reaches almost a constant value. The pattern is just opposite in case of a right lane change, therefore, x has totally different patterns for different types of lane changes and thus is suitable as a feature. Similarly, figure 4.2b shows that the pattern of ϕ is noticably different for different lane changes and thus can be chosen as a feature. Another benefit of choosing ϕ as a feature is that it makes the algorithm **independent of the road structure** because ϕ is the heading angle of the car with respect to the road curvature. A better illustration can be seen in figure 4.3 explains it in a better way. A vehicle following the road curvature is defined to have a straight path inspite of the wavy road. All the maneuvers either straigth, left or right are always taken with respect to the road curvature.

Along with x and ϕ, their first derivatives, $\dot{x}$ and $\dot{\phi}$ (see figure 4.4) also have different patterns for different lane changes and therefore are chosen as features. Since speed limit is



(a) x, $\dot{x}$, ϕ, $\dot{\phi}$ for Right lane change



(b) x, $\dot{x}$, ϕ, $\dot{\phi}$ for Left lane change

Figure 4.4 – Difference between right lane change and left lane change

different for different highways, therefore, intuitively we can say that choosing velocity as a feature will not be so fruitful. Still, for the sake of validation, different features have been used to make different feature vectors and the result is shown in Chapter 5.

Another issue while making a feature vector for the time series data is the length of the feature vector. Since a lane change is a continuous process and can last for 1 second, 2 seconds or even more or less based on the situation or the driver, a single data point at a single moment of time can not represent the complete lane change process precisely. Therefore, to generate ground truth data, a window is selected around the point when the vehicle just reaches the lane markings for a lane change and all the datapoints within that window are labelled as the true datapoints representing corresponding lane changes. This window size should be sufficient enough to capture a lane change process completely. Within this window a subwindow is selected and different features at different times within this subwindow are concatenated to form the final feature vector of fixed length. This subwindow is moved within the window in order to generate different data points representing lane changes. This process is shown in figure 4.5. The effects of window and subwindow sizes, and different feature vectors are discussed in details in Chapter 5.

Mathematically, if $(t1 + t2)$ represents the window size (see figure 4.5), f represents the feature length for each feature (subwindow size), fps represents the data frames collected per second, and lateral position (x), steering angle (ϕ) and their derivatives are the selected features, then a feature vector at time T can be represented as:

$$F_T = [X_T, \dot{X}_T, \Phi_T, \dot{\Phi}_T] \in \Re^n$$

where,

$$t = \frac{1}{fps}$$

$$n = 4 \times f$$

$$X_T = [x_{T-f\times t}, x_{T-(f-1)\times t}, ..., x_{T-t}, x_T]$$

$$\dot{X}_T = [\dot{x}_{T-f\times t}, \dot{x}_{T-(f-1)\times t}, ..., \dot{x}_T]$$

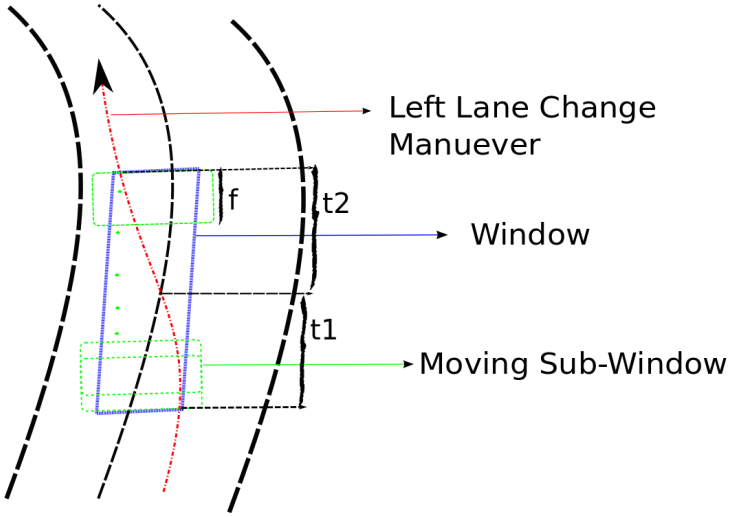


Figure 4.5 – Creating window and subwindow for making feature vectors

$$\Phi_T = [\phi_{T-f\times t}, \phi_{T-(f-1)\times t}, ..., \phi_T]$$

$$\dot{\Phi}_T = [\dot{\phi}_{T-f\times t}, \dot{\phi}_{T-(f-1)\times t}, ..., \dot{\phi}_T]$$

After the selection of features, all the training data is labelled and represented in the form of $\{(F_1, y_1), ..., (F_m, y_m)\}$ for the training purpose, where F_i represents the ith feature vector and y_i represents its corresponding label. Similarly while testing, the input data is converted in the form of a feature vector and trained model of SVM is used to find the corresponding label.

Kernel Selection

The motivation behind using kernel is that it maps the data from low dimensional space into high dimensional space converting a nonlinear classification problem at low dimension into a linear classification problem at high dimension [5]. The basic four kernels generally seen in the literature are:

1. linear: $K(X_i, X_j) = X_i^T X_j$
2. sigmoid: $K(X_i, X_j) = tanh(\gamma X_i^T X_j + r)$
3. polynomial: $K(X_i, X_j) = (\gamma X_i^T X_j + r)^d, \gamma > 0$
4. radial basis function (RBF): $K(X_i, X_j) = exp(-g\|X_i - X_j\|^2), d > 0$

Here, g, r, and d are the kernel parameters. The thumb rule is to choose a kernel that gives best results in the cross validation (m-fold cross validation: divide dataset into m parts, choose (m-1) parts for training and the remaining part for testing, repeat this for each part). In this work, RBF kernel is used and its performance is compared with other kernels and discussed in Chapter 5.

Finding the best parameters

Different parameters used in the proposed approach till now are (figure 4.5):

1. $t1$: It represents the difference between the time of starting of lane change maneuver and the ground truth.
2. $t2$: It represents the difference between the time of end of lane change maneuver and the ground truth.
3. feature length (f) : It represents the length of each feature that makes the complete feature vector.
4. C and g : C and g are the regularization and the kernel parameters respectively.

The results of SVM is highly sensitive to these parameters. The methodology to find the best values of these parameters and the effects of these parameters on the learning is discussed in Chapter 5.

4.2.2 Generalized Bradley-Terry model based probabilistic output

The approach proposed by [10] is used to get the probabilistic outputs from the multiclass SVM. This approach extends the Bradley-Terry model for paired individual comparisons into paired team comparisons. The algorithm for "one-against-rest" multiclass probability estimate using the approach proposed by [10] can be briefly written as:

Algorithm:
Let there are k classes.
Objective: find $p_s = P(x \; in \; class \; s)$, likelihood of example x for class s.
Assumption: Classes are balanced, i.e. number of training data in each class is almost the same.
Also r_i is known, then:
If $\sum_{i=1}^{k} r_i = 1$,
optimal $p = [r_1, \cdots, r_k]^T$.
else
find the root of $\sum_{s=1}^{k} \frac{(1+\delta) - \sqrt[2]{(1+\delta)^2 - 4r_s\delta}}{2\delta} - 1 = 0$.
put δ to find optimal p_s using, $p_s = \frac{(1+\delta) - \sqrt[2]{(1+\delta)^2 - 4r_s\delta}}{2\delta}$.

4.3 Bayesian filter to reduce false alarms

Previous section gave details about how to use SVM to get multiclass probability estimates representing different lane change maneuver probabilities for the given feature vector. In real data, from many experimentation discussed in Chapter 5, it has been observed that SVM alone for the classification gives many false alarms. These false alarms can be very dangerous while driving. To address this problem of false alarms a Bayesian filter (BF) has been used on top of the SVM. Therefore, SVM is not used to make the final classification decision rather it's giving probabilities as the output which acts as one of the input to the BF. The posterior output of the BF is then used for the final classification decision making.

4.3.1 BF Algorithm

The Bayesian filtering algorithm is mainly the recursive form of Bayes rule which gives an inverse relationship between posterior, prior and likelihood.

$$posterior \propto likelihood \times prior$$

More precisely, a Bayesian filter in discrete form can be written as:

$$P(X_t|Z_{1:t}) \propto P(Z_t|X_t, Z_{1:t-1})P(X_t|Z_{1:t-1}) \tag{4.11}$$

$$\propto P(Z_t|X_t)[\sum_{x_{t-1}} P(X_t|x_{t-1})P(x_{t-1}|Z_{1:t-1})] \tag{4.12}$$

where, $P(X_t|Z_{1:t-1})$ is the posterior distribution at time t, $P(Z_t|X_t)$ is the likelihood, $P(X_t|x_{t-1})$ is the state transition probability and $P(x_{t-1}|Z_{1:t-1})$ is the prior.

4.3.2 Implementation of the Bayesian filter

To implement the Bayesian filter we need to compute three terms likelihood, state transition probability, and prior. The prior at time t is the posterior at time $t-1$, therefore can be calculated recursively and initialized uniformly at time $t = 1$. To find the likelihood and the state transition probability we first need to define X_t. Since our aim is to predict lane change which further has been divided into three classes **left lane change**, **right lane change** and **no lane change**, therefore X_t in our problem can take three values each representing a particular class. So the equation can be rewritten as:

$$P([M_t = L]|Z_t) \propto P(Z_t|[M_{t=L}])[\sum_{m_{t-1}=L,R,N} P(M_t = L|M_{t-1} = m_{t-1})P(M_{t-1} = m_{t-1}|Z_{1:t-1})]$$

where, M_t represents maneuver at time t and L, R and N represents left, right and no lane changes respectively. Like wise we can write the posterior equations for R and N. Till now we have formulated the Bayesian filter in a way to fit the lane change prediction problem.

Likelihood

$P(Z_t|[M_{t=L}])$ represents the likelihood. The probabilistic output of SVM at each time step can directly be used as the likelihood (or measurement).

State Transition Matrix

$P(M_t|M_{t-1})$ represents the state transition probability. The corresponding state transition diagram for all the three states or classes L, R, N is shown in Figure 4.6

This state transition diagram can be represented in the form of a 3×3 matrix and our task is to find all the nine parameters of this matrix offline using the training data. The normal way to find these parameters is by doing maximum likelihood (ML) estimation using expectation maximization algorithm. But since in our case all the states $M_{1:T}$ in our training can be labeled the state transition matrix is simply the normalization of the co-occurrences (counts):

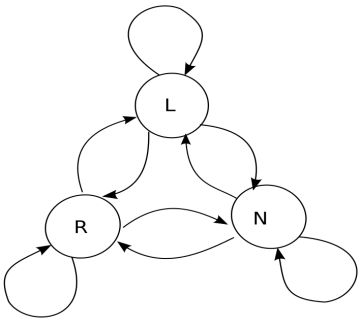


Figure 4.6 – State transition diagram for left, right and no lane change

$$S_{ML}(i,j) = \frac{S(i,j)}{\sum_k S(i,k)}$$

where, $S(i,j)$ is the number of $i \rightarrow j$ transition the training sequence.

$$\begin{array}{c} \\ L_{t-1} \\ R_{t-1} \\ N_{t-1} \end{array} \begin{array}{c} \begin{array}{ccc} L_t & R_t & N_t \end{array} \\ \begin{pmatrix} s_{11} & s_{12} & s_{13} \\ s_{21} & s_{22} & s_{23} \\ s_{31} & s_{32} & s_{33} \end{pmatrix} \end{array}$$

Therefore, the probabilistic output of the SVM acts as the likelihood and learned state transition matrix acts as the prior to the Bayesian filter and the posterior from the Bayesian filter is then used for the final classification decision making.

4.4 Detailed Overall Approach

The detailed overall approach is shown in figure 4.7. Dotted arrows points to the graphical form of the output of each block and solid arrows represents the process flow.

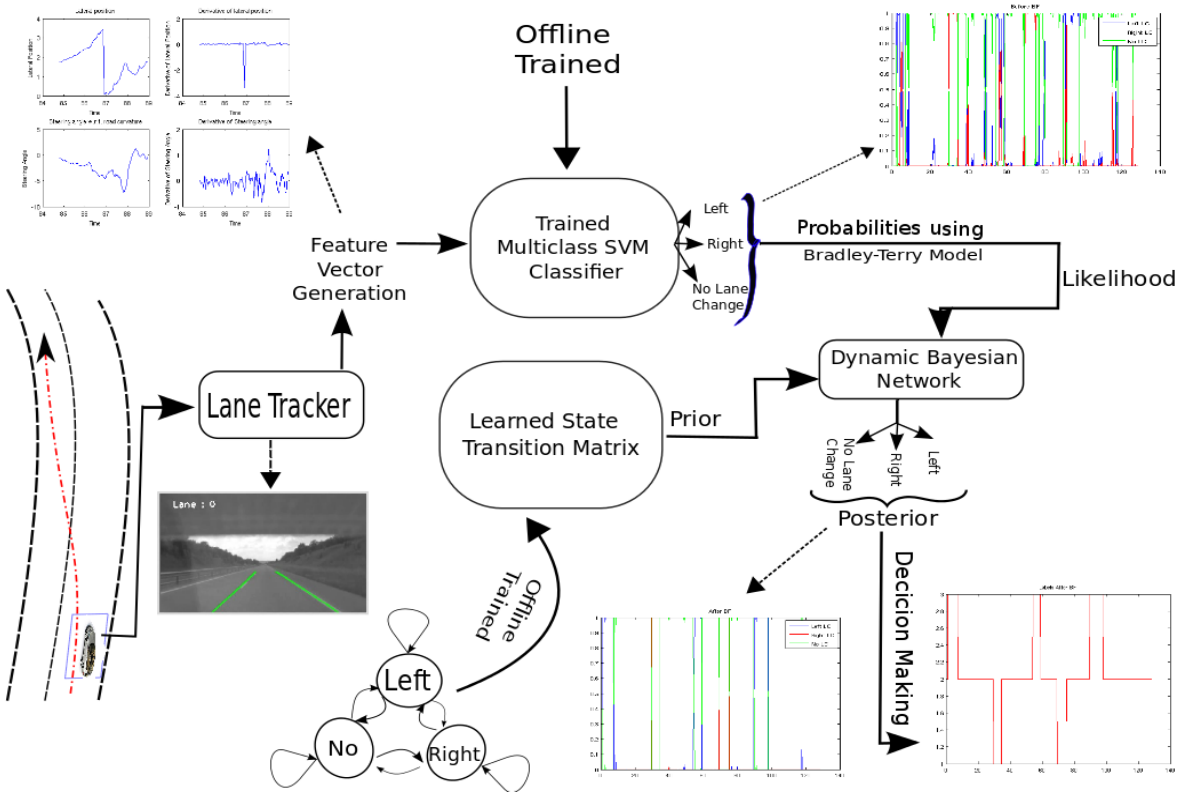


Figure 4.7 – Detailed Overall approach

— 5 —
Results and Discussion

Chapter 4 gave the technical details about the implemetation of the proposed approach. In this chapter, the proposed approach is evaluated using real-world data. Section 5.1 gives description about the different terminologies used throughout the chapter, section 5.2 and 5.3 compares the results between SVM and SVM+BF based approaches. Section 5.4 is introduced for the robustness evaluation of the overall approach and section 5.5 discusses the startegies used to find the best values of different parameters used to implement the proposed approach and finally section 5.6 compares the effects of different sizes of the training dataset on the proposed approach.

The experimental platform used to collect real-world data is a Lexus LS600h (see chapters 4 and 5) equipped with a TYZX stereo camera placed behind the windshield. The stereo baseline is 22 cm, with a field of view of 62°. Camera resolution is 512×320 pixels with a focal length of 410 pixels. The lane tracker integrated with the experimental platform was used to collect different datasets with more than 200 lane changes (110 left lane change, 90 right lane change) on a highway near Grenoble, France. Before discussing about the results of training and testing using these datasets, first of all let's have a look on some of the terms frequently used in this chapter.

5.1 Terminologies and Evaluation Metrics

Different terminologies and the evaluation metrics used in this thesis are defined in this section.

- Ground Truth: It represents the moment at which the vehicle under consideration is about to reach the lane markings during the process of lane change.
- Classes: For the purpose of classification, each maneuver has been assigned a class number. 1 for right lane change, 2 for no lane change, and 3 for left lane change.
- Prediction point: The time at which there is a change in class while testing i.e. from 1 to 2, or from 2 to 3 etc.
- Prediction time: Prediction time is the time difference between the ground truth and the corresponding prediction point while testing. To find prediction time, first of all a prediction point with the same class as that of the ground truth is searched in a window of 4 seconds (assumuing it to be within 4 seconds window if predicted) before the ground truth time in

the classified data (or tested data). If there are many prediction points with the same label as that of ground truth within 4 seconds then the prediction point nearest to the ground truth is chosen (worst case) and then this prediction point is used to find the prediction time. If the ground truth class is classified beyond the ground truth time in the testing data then this is considered as a flase detection because the aim of this thesis is to predict the classes in advance.

- Precision, Recall and F1-Score: In terms of classification, precision is the probability that the output of the classifier belongs to a relevant class, whereas, recall is the probability that a relevant class will be the output of the classifier. F1-Score is the harmonic mean of precision and recall. In this thesis, most of the time the **average** of precision, recall and F1-Score for left lane change class and right lane change class is used for the performance evaluation purpose.

$$Precision = \frac{true\ positive}{true\ positive + false\ positive}$$

$$Recall = \frac{true\ positive}{true\ positive + false\ negative}$$

$$F1 - Score = \frac{2(Precision)(Recall)}{(Precision + Recall)}$$

5.2 Results with SVM

The training data consists of 22 left lane changes, 24 right lane changes and 24 trajectories belonging to no lane change situation. The reason for choosing this set of training data is discussed in section 5.6. The testing data consists of total of **69 (left + right)** lane changes.

We have seen in section 4.2.1 that the implementation overall approach involves many parameters e.g. window size ($t1$ + $t2$), subwindow size (or feature leangth), regularization parameter (C), kernel parameters (g) etc. Choosing these parameters is very crucial as these parameters effects the overall results greatly. Section 5.5 explains how to find the best values of these parameters. Different values of these parameters used in this section are:

1. Radial basis kernel with kernel parameter (g) = 0.0625
2. Regularization parameter (C) = 8
3. Lane change maneuver window parameters: t1 = 2, t2 = 2
4. Subwindow parameter or Feature length (f) = 32
5. Feature vector consisting of lateral position of the lane w.r.t. vehicle (x), steering angle w.r.t. road curvature(ϕ), and their derivatives ($\dot{x}$) and ($\dot{\phi}$).
6. Data collection rate (fps) = 32, final classification rate = 5 classifications per second.

Two people were used to generate the ground truth (time when the vehicle just reaches the lane markings for crossing the lane) and their average was taken as the final ground truth. Figure 5.1 represents the ground truth data for 5 lane changes (3 left + 2 right) testing data for visualization and understanding of the plots. In reality the algorithm was tested on total of 69 lane changes. As discussed earlier, right lane change is labeled as 1, no lane change as 2, and left lane change as 3. This testing data was used as an input to the trained multiclass

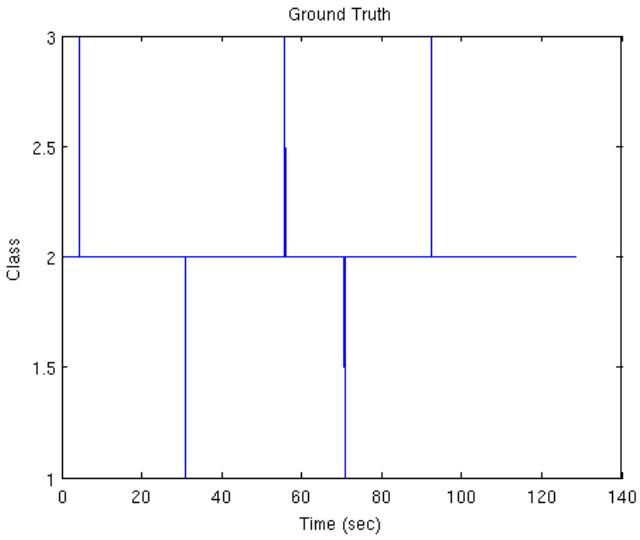
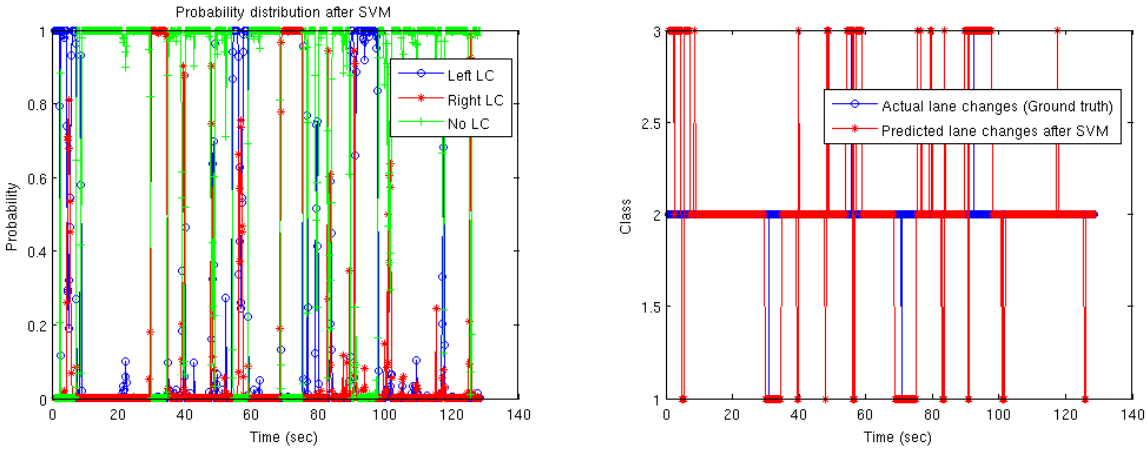


Figure 5.1 – Ground Truth data for testing

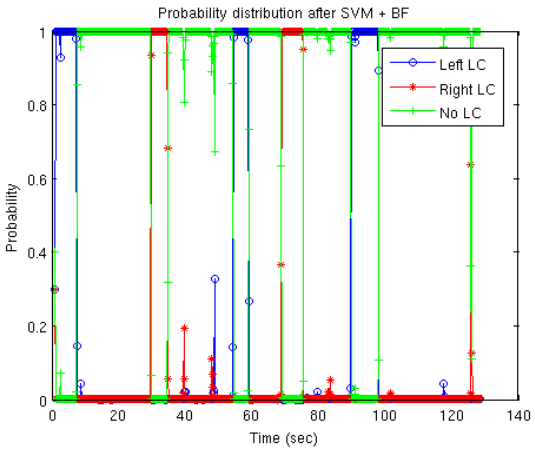


(a) Probabilistic output using SVM

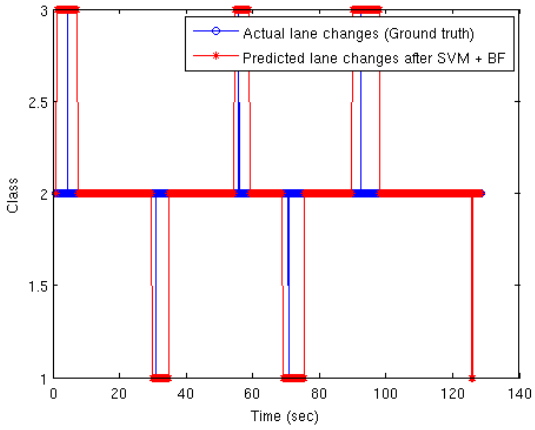
(b) Predicted Lane changes using SVM

Figure 5.2 – Results using SVM

SVM classifier and different probability measures for different classes were taken as the output (see figure 5.2a). For a particular feature vector input at a particular instant, the SVM outputs the probability of this feature vector belonging to a particular class. Note that, these classes are independent to each other so the probability for each class may not sum to one for any particular moment. The testing feature vector at any moment belongs to the class having the highest probability (see figure 5.2b). If as a final output we select the highest value of probability,figure 5.2b, we observe many false alarms. There are situations at which the classification fluctuates from left lane change to right lane change and again comes back to the left lane change within 6-7 frames (6/32 seconds) which is not possible in real time.



(a) Probabilistic output using SVM + BF



(b) Predicted Lane changes using SVM + BF

Figure 5.3 – Results using SVM + BF

5.3 Results with combination of SVM and Bayesian filter

We have seen in the previous section that the results of the SVM on the testing dataset gives many false alarms. To reduce these false alarms a Bayesian filter (as discussed in Chapter 3 and 4) is used on top of the SVM. The likelihood is taken from the SVM and the state transition matrix is learned from the dataset. The learned state transition matrix is:

$$\begin{array}{c} \\ L_{t-1} \\ R_{t-1} \\ N_{t-1} \end{array} \begin{array}{c} \begin{array}{ccc} L_t & R_t & N_t \end{array} \\ \begin{pmatrix} 0.9920 & 0 & 0.0013 \\ 0 & 0.9920 & 0.0013 \\ 0.0080 & 0.0080 & 0.9974 \end{pmatrix} \end{array}$$

From this matrix we can see that the probability of moving from left lane change to right lane change (or vice veras) just at the next instant is zero and this is true in real scenario also. Figure 5.3a represents the probability distribution for each classes after filtering out the false alarms using Bayesian filter (BF). Figure 5.3b presents the predicted classes using the filtered probability distribution. Table 5.1 gives the performance measures in terms of the average precision, recall and prediction time for SVM and SVM+BF for the comparison purpose. We can see from the table that the precision is improved from 0.2857 to 0.7154 which is a highly remarkable improvement. This increase in precision improves F1-Score also because recall remains the same for both the cases. The average prediction time is decreased by 0.022 seconds.

Figure 5.4 represents the histogram of prediction time for all the testing dataset consisting of 69 lane change. We can observe from the histogram that most of the lane changes are getting predicted almost 1.3 seconds earlier and the maximum prediction horizon reaches till 3.29 seconds. **So, in conclusion of this section, we can say that using Bayesian filtering on top of the SVM improves results remarkably in terms of precision and recall. The average prediction time is decreased by a very small fraction (0.022 secs) which is acceptable with**

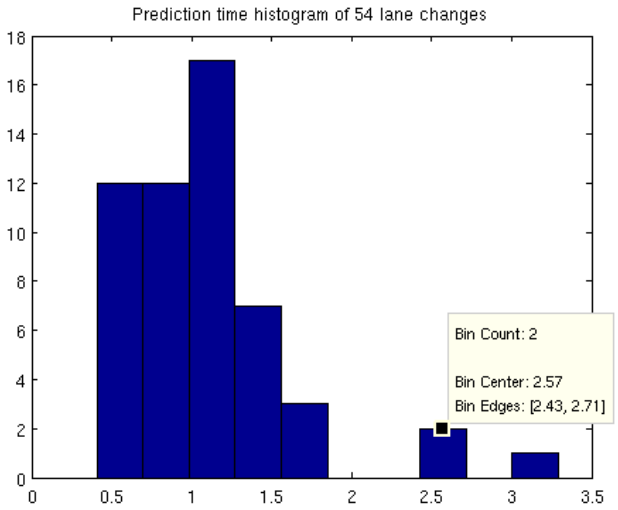


Figure 5.4 – Histogram of prediction time for testing data with 69 lane changes

Table 5.1 – Comparison of SVM and SVM+BF (Tested on total of 69 lane changes)

Cases	Precision	Recall	Avg prediction time (sec)
SVM	0.2857	1	1.2947
SVM + BF	0.7154	1	1.2718

such a remarkable improvement on precision. The maximum prediction horizon for the proposed approach is 3.29 seconds. BF reduced many false alarms but still we don't have results with zero false alarms, therefore, there is still scope for improvements. The reason for not getting zero false alarms may be the real data set that we are dealing with and the fluctuations observed in the lane tracker.

5.4 Robustness Evaluation

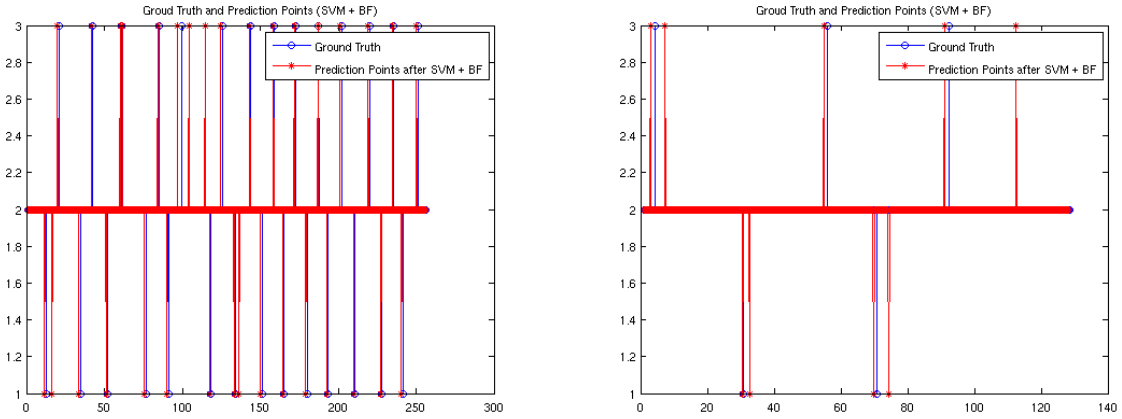
The goal of this section is to test the effects of training and testing data of different drivers. Two people were asked to drive and the classifier was trained and tested for different data respectively. If there are two drivers x and y, then different cases that can arise are:

- **Case 1**: Training with x and testing with y, figure 5.5a.
- **Case 2**: Training with y and testing with x, figure 5.5b.
- **Case 3**: Training with x and testing with different driving scenarios of x i.e. testing with x1 and x2 (see figures 5.6a and 5.6b).

Table 5.2 represents the different performance metrices for each cases. We can observe from the table that the recall is 1 for all the cases. Also the F-1 score, precision and prediction time are almost the same. The small variation in these metrics may be because of values of the

Table 5.2 – Robustness Evaluation

Cases	Precision	Recall	F1-Score	Avg. Prediction Time (secs)
Case 1	0.7849	1	0.8717	1.0032
Case 2	0.55	1	0.7084	0.9495
Case 3(a)	0.8235	1	0.9032	0.9711
Case 3(b)	0.6465	1	0.7786	1.0687



(a) Training with x and testing with y

(b) Training with y and testing with x

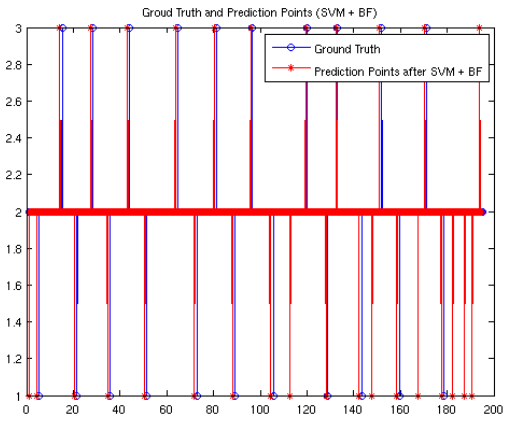
Figure 5.5 – Robustness Evaluation

different parameters used in the training and testing of the SVM. Effects of these parameters is discussed in the next section.

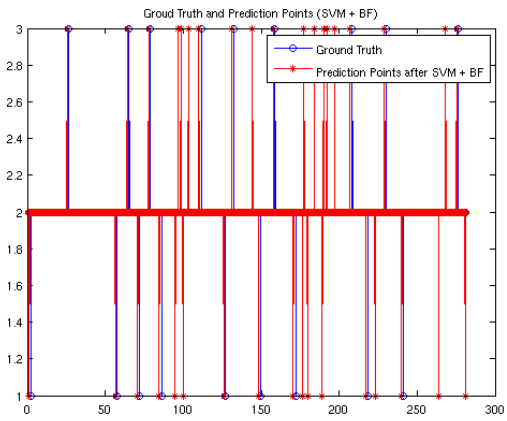
5.5 Finding the best parameters

There is a total of five parameters (see section 4.2.1) involved during the implementation of the proposed approach (SVM+BF) and each parameter has remarkable effects on the prediction result, therefore, best values of these parameters should be chosen for testing purpose. Simultaneously finding the best parameters is a five dimensional grid search problem and since the values of each parameter may vary in a large range, therefore, it is computationaly very expensive and time taking job. For example, if each parameter takes 20 values and the learning takes 1 minute for each iteration then total time required to find the best parameter will be $(20)^5 \times 1 \simeq 6$ years. In our case, each parameter takes much more than 20 and learning takes almost 3-4 minutes, therefore, it is almost impossible to find best parameters using this approach. To make it feasible to find the best parameters for the overall proposed approach (SVM+BF) some strategies which were used are as follows:

- **Step 1**: Fix $t1$, $t2$ and find best C and g using grid search with exponentially growing sequences of C and g, [7]. For example, $C = 2^{-3}, 2^{-1}, ..., 2^3$ and $g = 2^{-4}, 2^{-2}, ..., 2^2$. In this



(a) Training with x and testing with x1



(b) Training with x and testing with x2

Figure 5.6 – Training with x and testing with different maneuvers of x

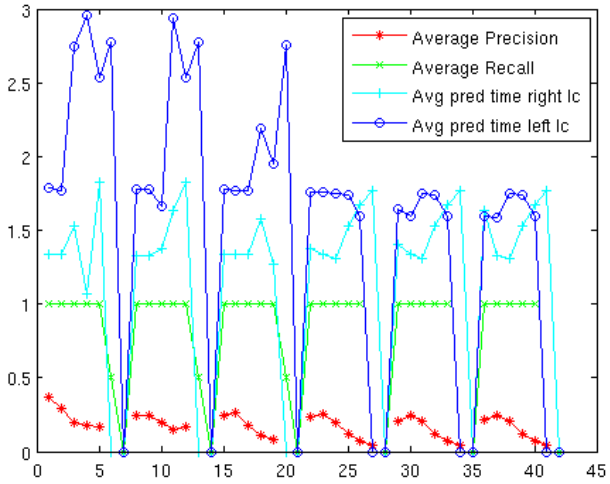
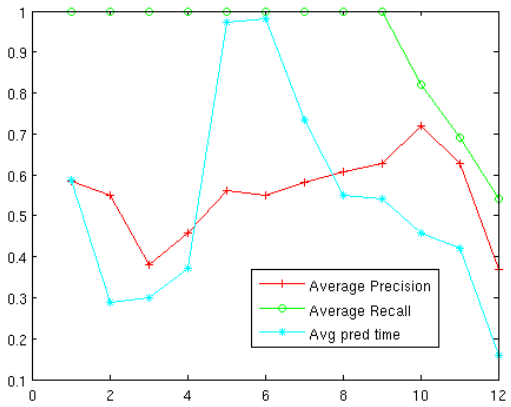
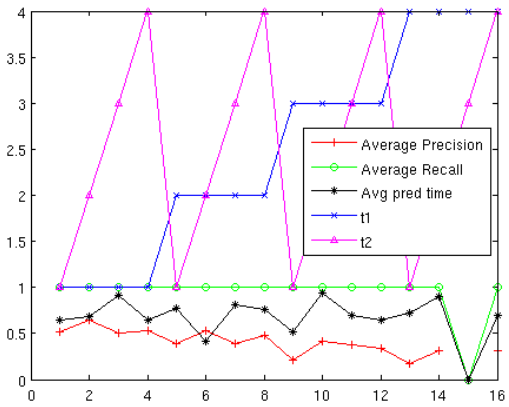


Figure 5.7 – Finding best combination of (C,g) parameters for the algorithm

case, the fixed values of $t1$, $t2$ and f are 2, 2 and 32 respectively. While fixing these parameters it was made sure that these parameters gave good results while testing in different scenarios. To confirm this many trials were done. Figure 5.7 represents different values of precision, recall, and prediction time (separately for left and right lane changes) for different combinations of C and g. We can see from the plot that there are many situations at which the prediction time is very high but recall and precision are vey low leading to dangerous situations in real time. Therefore, we choose C and g combinations that gives very high precision and recall at the same time. *Different* (C, g) *combinations giving high precision and recall found are: (0.25,0.0625), (1,0.125),(8,0.0625) etc*. **Note that** the precision plot has many breaks i.e. it is not continous at many points and the reason behind this is that the classifier for these (C, g) combinations was not at all able to predict any lane change (left and right).



(a) Finding best feature length (f)



(b) Finding best values of (t1,t2)

Figure 5.8 – Finding best parameters (f,t1,t2)

We can say that the classifier was totally biased towards no lane change class, therefore, true positives and false positives became zero leading to **NaN** (not a number) value of precision.

- **Step 2**: After finding best C and g, these parameters were fixed to their best values, $t1$ and $t2$ were again fixed to 2, and the best value of f was found. The value of f was varied between 4 to 92 with increment of 8 in each iteration. Figure 5.8a represents the average precision, average recall, and average prediction time for different values of f. We choose f that gives best precision and recall along with good prediction time. We can see from the plot that very small values of f gives low precision as well as very less prediction time and increasing f beyond some values decreases both precision and recall along with the prediction time. Intermediate values of f gives the best precision, recall and prediction time. Thus some of the best values of f found using this approach are: [36,44,52].
- **Step 3**: Now the values of C, g and f are fixed to their best values and the iteration is done over a range of $t1$ and $t2$ to find their best values. $t1$ and $t2$ were varied between 1 to 4 seconds with 1 second increment step. Figure 5.8b represents the plot of average precision, average recall, average prediction time, $t1$ and $t2$. **A very interesting point** with $t1 = 4$ and $t2 = 3$ in the plot shows precision = NaN, recall = 0, and prediction time = 0. With these combinations of $(t1,t2)$ and the best values of (C,g,f) i.e. (8,0.0625,32) the classifier is completely biased towards no lane change class. The reason behind this is that such a long window ($t1+t2 = 7$ seconds) around the ground truth represents right or left lane change feature vectors also as no lane change feature vectors, therefore, **nullifying the overall discriminative capabilities of the classifier**. Some of the best values of $(t1,t2)$ found using this approach are: $(1,2)$, $(2,2)$, $(2,3)$, $(3,2)$.

In conclusion of this section we can say that the parameters effects the overall approach remarkably. Choosing a bad parameter can lead to situations with very poor precision and recall, therefore, the best values of these parameters giving highest precision and recall should be chosen for the training and testing purposes.

Table 5.3 – Effects of the size of different training datasets

Training Datasets	No. of lane changes	Precision	Recall	F1-Score	Avg. Prediction Time (secs)
Dataset-1	11	0.2666	1	0.4167	1.7434
Dataset-2	11	0.2833	1	0.4285	2.5294
Dataset-3	29	0.7	1	0.7857	2.1031
Dataset-4	38	0.6429	1	0.7222	2.0655
Dataset-5	46	0.8334	1	0.9000	1.9903
Dataset-6	75	0.75	1	0.8334	1.7521
Dataset-7	91	0.80	1	0.8750	1.4198
Dataset-8	119	0.4167	1	0.5834	1.1064

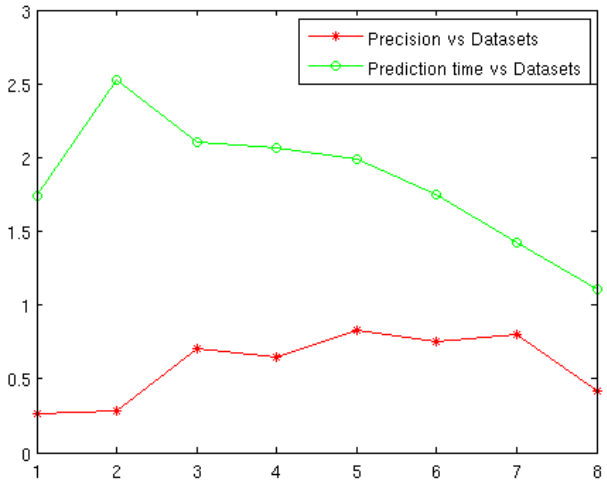


Figure 5.9 – Effects of different datasets

5.6 Effects of the size of the training data

Any learning algorithm greatly depends on the type and the size of the dataset for training. This section discusses on how to choose the size of training dataset and their effects. In our case we have a dataset of almost 200 lane changes and have to choose the number of lane changes for training purpose. Also the training dataset should not be biased towards any of the classes. In general, the dataset should be balanced and sufficient enough to train the classifier in order to differentiate different classes with a good margin. To understand this in details, the training dataset was divided into 8 different datasets with **increasing** number of lane change maneuvers and was tested on a **common** testing dataset. The testing was done on the overall proposed approach (SVM + BF). The performance measure for each training and testing process is shown in table 5.3. We can see from the table that the average recall is 1 in each case but the average precision, average F1-score and average prediction time is varying as the **size** of training data is changing. Since, average F1-score is measured from the combination of average precision

and average recall, therefore, only the plots of average precision and average prediction time w.r.t. datasets is shown in figure 5.9. We can see from the figure 5.9 that the average precision is increasing with increasing the dataset size and then it reaches it's maximun (0.8334) and starts decreasing with further increase in the dataset size. The trend for the average prediction time is not the same. The average prediction time is maximum (2.5294 seconds) for Dataset-2 but for this dataset the precision is 0.2833. **This represents that if we try to maximize our average prediction time while choosing the dataset size for training then we may end up in a situation with very less precision leading to many false alarms not acceptable for commercial ADAS. Therefore, we have to give preference to any one, either average precision or average prediction time, and the obvious choice for real time scenario will be the precision. So we took Dataset-5 as our standard dataset for training purpose**.

Conclusions and related future work is discussed in next chapter.

— 6 —
Conclusions and Future Work

6.1 Conclusions

- In this thesis lane change intention prediction problem is addressed using a combination of discriminative classifier (Support Vector Machine) and Bayesian filter.
- The overall approach was tested on almost 100 lane change real-world data collected using a Lexus experimental platform.
- The lane change intention prediction horizon is up to 3.29 seconds with an average prediction time of 1.3 seconds.
- Adding a Bayesian filter on top of the SVM improved the average precision from 0.2857 to 0.7154 with recall of 1, thus increasing the realiability of the system.
- Using a Bayesian filter decreased the average prediction time by 0.022 seconds which is acceptable with such a remarkable improvement on the precision.
- Still the precision is not 1 because of some false alarms. The most probable reason for such false alarms is the fluctuations and jumps of the lane tracker. Sometimes the lane tracker was not able to detect lanes properly thus giving highly noisy data. The other reasons for such false alarms can be the limitations of the sensors and of the algorithm.

6.2 Future Work

- Testing the proposed with more drivers and in different scenarios (e.g. slow traffic) for better evaluation.
- Implementing the proposed approach using Structural SVM and trying to improve the results.
- Improving the lane tracker in order to get better input data which in turn may decrease the number of false alarms.
- Implementing the proposed approach using C++ and integrating it with the Lexus platform.

Bibliography

[1] Abdourahmane Koita Abderrahmane Boubezoul and Dimitri Daucher. Vehicle trajectories classification using support vectors machines for failure trajectory prediction. In *IEEE ACTEA*, pages 486–491, 2009.

[2] G. S. Aoude. *Threat Assessment for Safe Navigation in Environments with Uncertainty in Predictability*. PhD thesis, Massachusetts Institute of Technology, Department of Aeronautics and Astronautics, Cambridge, MA, July 2011.

[3] G. S. Aoude and J. P. How. Using support vector machines and Bayesian filtering for classifying agent intentions at road intersections. Technical Report ACL09-02, Massachusetts Institute of Technology, Cambridge, MA, September 2009. Aerospace Controls Lab.

[4] Georges Aoude, Vishnu Desaraju, Lauren H. Stephens, and Jonathan P. How. Behavior classification algorithms at intersections and validation using naturalistic data. In *Intelligent Vehicles Symposium*, pages 601–606, 2011.

[5] Nello Cristianini and John Shawe-Taylor. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

[6] J. Goldbeck and B. Huertgen. Lane detection and tracking by video sensors. In *Intelligent Transportation Systems, 1999. Proceedings. 1999 IEEE/IEEJ/JSAI International Conference on*, pages 74 –79, 1999.

[7] C. W. Hsu, C. C. Chang, and C. J. Lin. A practical guide to support vector classification. Technical report, Taipei, 2003.

[8] Till Hülnhagen, Ingo Dengler, Andreas Tamke, Thao Dang, and Gabi Breuel. Maneuver recognition using probabilistic finite-state machines and fuzzy logic. In *Intelligent Vehicles Symposium*, pages 65–70, 2010.

[9] Toru Kumagai, Yasuo Sakaguchi, Masayuki Okuwa, and Motoyuki Akamatsu. Prediction of driving behavior through probabilistic inference. In *Proceedings of the Eighth International Conference on Engineering Applications of Neural Networks*, pages 8–10, 2003.

[10] Tzu kuo Huang, Ruby C. Weng, Chih jen Lin, and Greg Ridgeway. Generalized bradley-terry models and multi-class probability estimates. *Journal of Machine Learning Research*, 7:2006.

[11] Harry Lum and Jerry A. Reagan. Interactive highway safety design model: Accident predictive module. In *Public Roads*, volume 59, United States Department of Transportation - Federal Highway Administration, 1995.

[12] Hiren M. Mandalia and Dario D. Salvucci. Using support vector machines for lane change detection. In *In Proceedings of the Human Factors and Ergonomics Society 49th Annual Meeting*, 2005.

[13] Joel C. McCall, David P. Wipf, Mohan M. Trivedi, and Bhaskar D. Rao. Lane change intent analysis using robust operators and sparse bayesian learning. *IEEE Transactions on Intelligent Transportation Systems*, 8(3):431–440, 2007.

[14] Daniel Meyer-Delius, Christian Plagemann, and Wolfram Burgard. Probabilistic situation recognition for vehicular traffic scenarios. In *ICRA*, pages 459–464, 2009.

[15] Brendan Morris, Anup Doshi, and Mohan M. Trivedi. Lane change intent prediction for driver assistance: On-road design and evaluation. In *Intelligent Vehicles Symposium*, pages 895–901, 2011.

[16] Klaus-Robert Müller, Alex J. Smola, Gunnar Rätsch, Bernhard Schölkopf, Jens Kohlmorgen, and Vladimir Vapnik. Predicting time series with support vector machines. In *Proceedings of the 7th International Conference on Artificial Neural Networks*, ICANN '97, pages 999–1004, London, UK, UK, 1997. Springer-Verlag.

[17] Andrew Ng. Cs229 lecture notes support vector machine. *Lecture notes CS229*.

[18] Nuria Oliver and Alex P. Pentland. Graphical models for driver behavior recognition in a smartcar. In *Intelligent Vehicles Symposium*, 2000.

[19] World Health Organization. *The injury chart book : a graphical overview of the global burden of injuries*. World Health Organization, Geneva.

[20] Michaël Garcia Ortiz, Jannik Fritsch, Franz Kummert, and Alexander Gepperth. Behavior prediction at multiple time-scales in inner-city scenarios. In *Intelligent Vehicles Symposium*, pages 1068–1073, 2011.

[21] Igor Paromtchik and Christian Laugier. Automatic parallel parking and returning to traffic maneuvers. In *Video Proc. of the IEEE Int. Conf. on Robotics and Automation*, pages 16–20, 1998.

[22] Dario D. Salvucci. Inferring driver intent: A case study in lane-change detection. In *in Proceedings of the Human Factors Ergonomics Society 48th Annual Meeting*, pages 2228–2231, 2004.

[23] Dario D. Salvucci, Hiren M. Mandalia, Nobuyuki Kuge, and Tomohiro Yamamura. Lane-change detection using a computational driver model. *Human Factors*, 49(3):532–542, 2007.

[24] Herrn Julien H. Simon. *Learning to drive with Advanced Driver Assistance Systems. Empirical studies of an online tutor and a personalised warning display on the effects of learnability and the acquisition of skill.* PhD thesis, Technischen UniversitÃt Chemnitz, April 2005.

[25] Christopher Tay. *Analysis of dynamic scenes: application to driving assistance*. PhD thesis, Institut National Polytechnique de Grenoble, France, 2009.

[26] Ioannis Tsochantaridis, Thomas Hofmann, Thorsten Joachims, and Yasemin Altun. Support vector machine learning for interdependent and structured output spaces. In *Proceedings of the twenty-first international conference on Machine learning*, ICML '04, pages 104–, New York, NY, USA, 2004. ACM.

[27] D. Wipf and B. Rao. Driver intent inference annual report. Technical report, University of California, San Diego, San Diego, 2003.