In the name of ***God***

은닉 마르코프 모델을 이용한 운전자 행동 양식의 인식

Driver Behaviour Recognition Using Hidden Markov Models

Driver Behavior Recognition Using Hidden Markov Models

by

Reza Haghighi Osgouei

Division of Electrical and Computer Engineering
(Computer Science and Engineering)
Pohang University of Science and Technology

A thesis submitted to the faculty of the Pohang University of Science and Technology in partial fulfillment of the requirements for the degree of Master of Science in the Division of Electrical and Computer Engineering (Computer Science and Engineering)

Pohang, Korea
12. 21. 2011
Approved by
Seungmoon Choi (Signature)
Academic Advisor

Driver Behavior Recognition Using Hidden Markov Models

Reza Haghighi Osgouei

The undersigned have examined this thesis and hereby certify that it is worthy of acceptance for a master's degree from POSTECH

12. 19. 2011

Committee Chair Seungmoon Choi (Seal)

Member Seungjin Choi (Seal)

Member Bohuyng Han (Seal)

To my precious ***parents***,

To my lovely wife ***Niloufar*** and my beloved son ***Parsa***.

MECE
20100980

레자, Reza Haghighi Osgouei, Driver Behavior Recognition Using Hidden Markov Models, 은닉 마르코프 모델을 이용한 운전자 행동 양식의 인식, Division of Electrical and Computer Engineering (Computer Science and Engineering), 2012, 97P, Advisor: Seungmoon Choi. Text in English

ABSTRACT

In this work we addressed the problem of modelling human driving behavior using hidden Markov models (HMMs). It is part of a bigger objective towards capturing and transferring driving skills from an expert driver to a novice trainee. We believe driving behaviors are in result of driver's decision making rules. So we drew our attention to identify and recognize driver's decisions or in another sense driving rules using driving time-series signals. For this end, first a driving simulator based on a commercial racing wheel is developed to simulate a desired driving task. The required driving signals including acceleration pedal position, steering wheel angle, velocity and heading of the vehicle are collected using the driving simulator. Then inspired by the fact that the only variables a driver has control on them are velocity and heading, their first-order derivative are extracted as the two most important features of driving patterns. Following the same inspiration, we developed an automatic segmentation method to detect the local extrema of controlling variables and divide data samples into a number of segments. We suggest during each segment, the driver keeps the pedal and wheel operations unchanged. Not all data segments come from different sources; there might be some criteria to group similar segments. In this line we proposed three partitioning methods, threshold-based, GMM-based, and hierarchical, all originated from dividing the two

dimensional feature space into a number of classes. In our belief, data classes are the time-domain realization of driving behaviors. According to each class, the parameters of one HMM are optimized to be used later for recognizing driving behaviors. The achieved high average correct classification rate, between 85% to 95% depend on the partitioning criteria, reveals the efficacy of proposed approach in classifying and recognizing driving behaviors. Finally we made use of behavior recognizers to compare an expert and a novice driver's performances in order to provide some feedback. Two methods one road-dependent and another road-independent are proposed for this end. The evaluation results proved the applicability of proposed methods.

Contents

List of Figures

Chapter 1

Introduction

It is all started by a question: is there any way to capture and transfer driving skills from an expert driver to a novice trainee? In another sense we want to know 1) how an experienced driver behaves in various driving situations, 2) how we can represent those behavioural driving skills and lastly 3) how we can make use of them to provide some appropriate multi-modal feedbacks to the trainee during driving training process. In this project we focused on the first two problems but always we had this in mind that we should introduce a framework which can be easily expanded to cover the third problem.

One way to address the problems of interest is looking at them as a modelling problem. If we can construct a model which represents the way human *reacts* in response to some input *actions*, we already have the answers. In case of driving application, we drew our attention to driver's decisions or driving rules and how we can define and recognize them. In fact, the way a driver makes driving decisions is a kind of realization of driver's driving behaviours and skills. So we suggest that by studying driver's decisions we can collect some useful insights about driving skills. As a matter of fact, driving behaviours are spread over and embedded inside driving signals. Driving signals might come from different sources, for example pedal and wheel, velocity and heading, and some environmental circumstances. Different kinds of driving behaviours can be extracted from these driving signals.

1.1 Motivation

Driving simulators provide a simulated driving environment to the operator to have some experience prior to driving a real car. Various driving situations such as icy and rainy roads, sudden stops, high degree ramps, and etc. can be simulated safely in the virtual environment, while in the real world they might be harmful and dangerous for the operator to implement. Driving simulators only provides a virtual environment for a safe and free driving, you may have several accidents without paying any money as insurance or maintenance! Different driving simulators might be vary in the quality they simulate a real environment. But how about the facilities and features they are providing. Is there any significant difference between them? The answer is No. They all try to provide a virtual driving experience in a more realistic way. That is a good achievement but it is not enough.

Here is a question: how a novice driver can learn driving skills? By reading a book, by listening to an expert driver's lecture, by watching a video tutorial, or by practising? What kinds of other tools might be helpful in this regards? How can we take advantage of expertise of a skilful driver in a more efficient way? How about a framework which can capture the driving skills from an expert driver and transfer to a novice trainee? Is it too futuristic? Maybe this is possible in Hollywood films but how about in real world? Just imagine it can be real, in that case not only the training time and cost would be decreased significantly but also several lives can be saved too. How? If the driver has prior experience of a difficult situation like a brutal accident, he already knows how to manipulate the situation according to an expert driver's experience.

According to [2] "acquisition of the behavioural skills of a human operator and recreating them in an intelligent system has been a critical but rather challenging step in their development. In the recent years, the most popular approach is to learn the human skill through observing the demonstrator. However, due to inherent stochastic characteristics of the human behaviour, the actions applied in each instance of performing the task can be different. They all, however, represent the same skill set for the task. A framework is required to define the inherent characteristics of that skill, independent from the actions applied each time the task is performed. This is achieved by considering the most likely human action in performing the task as the skill employed to perform the task. Hidden Markov model (HMM) can be employed as a stochastic method to define the skills and model their uncertainties. HMM can easily represent

the two stochastic processes associated with skill learning including the mental state, the hidden process and resultant state, the observed process. In addition, HMM is a parametric model and its parameters can be optimized by Baum-Welch algorithm for accurate estimation."

1.2 Thesis Objective

In this work we proposed a method to identify and recognize driving behaviours. Driving behaviours are spread over various driving signals. We suggest by processing and analysing appropriate driving signals, we are being able to derive some characteristics of driving behaviours. For this end we customized a driving simulator to acquire required driving signals. Amongst all possible options we did collect and store acceleration pedal position, steering wheel angle, velocity and heading of the vehicle. From the fact that velocity and heading are the only controlling variables a driver has access, we thought might be monitoring their changes over the time can give us some insightful information about driving behaviours. We extracted the first order-derivative of controlling variables as the two most important features of driving behaviours. They are named translational acceleration and rotational velocity. Next an automatic segmentation method is developed to divide driving signals into a number of segments. It is based on detecting local extrema of velocity and heading signals in first place and setting them as boundaries of segment in second place. From the fact that not all of these segments comes from different sources, we looked for some appropriate methods to group similar segments. Later we realized that it is not necessary to do grouping based on data segments, we can do it on data samples either. In this way we proposed three different partitioning methods, threshold-based, GMM-based and hierarchical. The main idea behind all of them is dividing the 2d feature space in a way that data samples in each partition or class have similar physical interpretation. We suggested that each class is an abstract of a driving behaviour. Next for each class one continuous HMM is trained and later is used as behaviour recognizer in evaluation process. To evaluate optimized HMMs, the method of sliding windows over testing data set is hired.

The experimental results proved the usefulness of the proposed approach for two reasons: 1) the extracted features carefully divided the two dimensional feature space into several classes, in a way when we train HMMs based on each class, the highest

correct classification rate, about 95%, is achieved; that means the performance of the behaviour recognizers are highly promising, and 2) they have some physical meaning embedded inside, that means we are one step closer to the bigger objective, which is providing feedback according to recognized driving behaviours.

1.3 Thesis Outline

The thesis is organized as follows: in chapter two some related work is reviewed. The theory and background knowledge is given in chapter 3. In chapter 4 the experimental set-up for acquiring required driving signals along with the idea behind the feature extraction is given. In next chapter the automatic segmentation and partitioning methods with some experimental results are given. The proposed approach for recognizing driving behaviours using hidden Markov models is given in chapter 6. In chapter 7 two ideas for providing feedback are introduced. Finally the thesis with conclusions and future work in chapter 8 is ended.

Chapter 2

Previous Work

2.1 Survey of Driver Modelling Approaches

A critical survey of driver behaviour modelling approaches is given in [1]. ”Driver models are needed for different purposes from assessing vehicle dynamics to monitoring driver status or just simply to better understand the underlying dynamics in driver behaviour. In addition to several types of need for driver models, each related research field emphasizes a different aspect of the driver (i.e. cognition, perception, processing reaction, control). Driver modelling approaches can be roughly divided into following groups: human factors, control theoretic, stochastic/ non-linear and hybrid models. A schematic of driver modelling approaches is given in Figure 2.1.

”It is noticeable that especially lateral control has been modelled by control theoretic approach due to its continuous characteristics. An example of this type of driver model can be found in [3] employing control theoretic approaches for lateral control behaviour. This model includes driver's delays, feedback in form of lateral position error, and neuro-muscular response taken from an earlier work on flight-pilot modelling studies [4].

Other noticeable and widely known control theoretic models for lateral control can be listed as McRuers model [5] containing anticipatory, compensatory and precognitive control for better representation, and MacAdams optimal preview control model [6]. The common property of these models is that all of them agree with cross-over model [7] which can explain single loop manual tasks performed by humans. The advantage of control theoretic models is that they give a physical/causal relationship be-

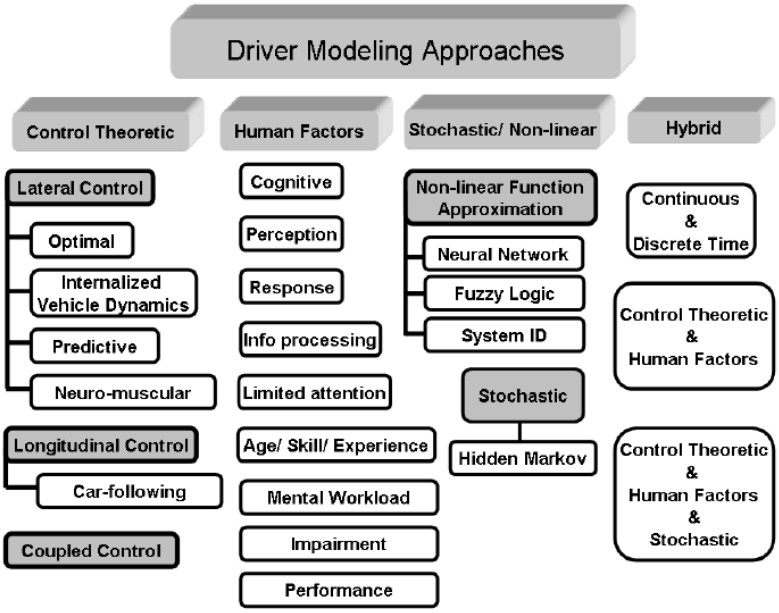


Figure 2.1: A schematic grouping of driver modelling approaches [1].

tween the input and output variables. In this aspect, the control theoretic models may give insight into driver behaviour. Although some of them ignore the non-linearity, they provide a sound mathematical framework in analysing driver behaviour in control level. However useful they may be, it should be noted though that most of the control theoretic models are designed to be used in improving vehicle dynamics and handling quality, but not for explaining driver behaviour or design of co-operative systems. These driver models have low fidelity in reproducing driver control commands that have similar characteristics to a real driver in time-domain. The main reason of this infidelity is that these models are designed for tracking the center of the lane or road-median almost perfectly, whereas a real driver would deviate from the median more as demonstrated by [8]. The human driver allows the lateral position error to build up until it reaches a threshold that driver perceives it as a deviation and makes correction. This is known as complacency, and it is also related to the fact that most of the sensory input that driver uses is not instant measurement of the lateral position but visual cues and vestibular feedback. In order to perceive and process this feedback takes time and it is not instantly used by the driver but delayed. A similar behaviour is observed also in longitudinal control while car following and was taken into account in Lubashewskys rational driver model [9]. Since the existing control theoretic mod-

els cannot account for complacency, [8] used driver simulator data to identify lateral control model of the driver using system identification tools and ARX models. In [10] driver steering model was identified with particular interest in structured and unstructured model uncertainty. Their work is important as they imply that the structured uncertainty can be used to monitor driver and use adaptive control framework to address the risk from driver performance deterioration whereas the unstructured certainty coming from unmodelled non-linearity can be addressed by robust control. The non-stationary, uncertain and non-linear nature of driver behaviour was understood by other researchers too. In [11] cascaded Neural Networks (NN) are used with some flexibility employing Extended Kalman Filters (EKF) for update and variable activation in newly added neuron layers. HMM is used to measure the stochastic similarity [12] between the model output and real driver data. This measure is reported to be better than mean square error since the nature of driving is stochastic and we should be looking for main trends in the data not the exact match in numerical sense. In fact, Markov Chains were used to sequence a bank of Kalman Filters for predicting driver actions using preparatory input actions [13]. Hidden Markov Models (HMM) were used to learn human action and transfer human skills for tele-robotics applications [14]. HMMs has later proved to be a very convenient tool in modelling driving control inputs or observed vehicle dynamics and it is widely used to model driver behaviour in several frameworks. In [15] HMM framework is used to recognize different driver manoeuvres and [16] used a similar framework for a top to bottom approach in search for 'drivermes', the meaningful smallest unit of driving signals. In another studies, HMMs were used to recognize manoeuvres and detect the driver distraction or driver faults using a hierarchical approach [17] [18]. Although HMMs are very powerful and can reproduce the driver behaviour with high stochastic fidelity, they lack the capability of explaining the physical/causal meaning of the resulting models.

In addition to mathematical approaches a large group of driver models are derived in human factor engineering. These models consider cognitive, perceptual, and neuro-muscular limitations of human. These models provide very important insight into driver behaviour especially explaining some of the uncertainty, delay and non-linear characteristics. In addition to this, the control theoretic and stochastic models tend to use measured (i.e. observable) data and they often stay in the control level modelling. The tactical and strategic levels in Michons hierarchical model [19] cannot be modelled with control theoretic or stochastic approaches. [20] proposed ACT-R

cognitive model of the driver modelling the information processing and inherent delays of the human cognitive system. As it can be seen, the models derived from human factor engineering are very useful; however, they do not represent a full driver model. Therefore, combined model structures including control aspects, stochastic processes and cognitive capabilities are proposed. These models can be described as 'hybrid' models. This approach is relatively new and very promising for obtaining comprehensive models. For example, [21] described human perception process by a discrete event technique the execution part is modeled by general predictive controllers and the velocity control is represented by a finite state machine to reveal its discontinuous control dynamics. In [22], researchers used a controller switching model for modelling collision avoidance manoeuvre employing piecewise polynomials. Furthermore [23] used similar approach for modelling vehicle following task dividing the car following control into four different modes. Another model using switching control is used by [24] employing simple control laws and defining the switching rule by a knowledge base."

2.2 Modelling using HMMs

Hidden Markov models are originally developed in the context of speech recognition around 40 years ago. One of the classic papers which introduced the main concept and its application to speech recognition is published by L. R. Rabiner [25]. The basic definition, notation, the three main problems and their solution is discussed thoroughly in the paper. Due to their proven ability, the attention of researchers on other areas such as robotics, human-machine interaction, virtual reality are attracted to use them in their applications. Recently we can find so many work in various application areas such as gesture recognition, hand-written recognition, human-robot cooperation systems in the literate.

In the context of driving application, though, we cannot see great interest to use HMMs to address the problem. One reason maybe is the highly stochastic nature of human driving behaviour. Moreover, both temporal and spatial dependency of human driving decision making rules, makes it hard to be fitted and represented by any kind of stochastic, or probabilistic methods. There are a few work in the literature which tried to address the problem from an specific point of view. For example some of them limited themselves to study the human driving behaviour in very specific moments

such as stopping time, lane changing, overtaking, turning and etc.

Naturally HMMs are very capable tools to study time-series and sequential data. In most cases some observations in the form of time-series driving signals are collected to be used as training ans testing data sets for the HMMs. Making use of HMMs means we are going to deal with a classification problem. In the current survey we focused on some important aspects of the previous researches. Amongst them we can name:

- Modelling framework: how they made use of hidden Markov models in their approach,
- Driving signals: what kind of driving signals are collected and processed,
- Experimental set-up: how they collected driving signals, using a simulated driving task or through driving of a real car.

One of the early work in the context of driving application is done under supervision of Prof. Yangsheng Xu [12]. They proposed a stochastic, discontinuous modelling framework, for abstracting human control strategies (HCS), based on Hidden Markov Models. A real-time graphic driving simulator (Figure 2.2 is developed as experimental platform to record human control data. In the simulator, the human operator has independent control over the steering of the car, the brake and the accelerator, although the simulator does not allow both the gas and brake pedals to be pushed at the same time. HMMs, as trainable statistical models, are applied to model human control strategy, not as a deterministic functional mapping, but rather as a probabilistic relationship between sensory inputs and control actions outputs. In their approach they made use of HMMs as the discontinuous controller having three separate phases:

- Input-space signals are first converted to an observation sequence of discrete symbols O^*, in preparation for Hidden Markov Model (HMM) evaluation.
- The resulting observation sequence O^* is then evaluated on a bank of discrete-output HMMs, each of which represents a possible control action A_i and each of which has previously been trained on corresponding human control data.
- Finally, the HMM evaluation probabilities are combined with prior probabilities for each action A_i to stochastically select and execute action A^* corresponding to input observation sequence O^*.

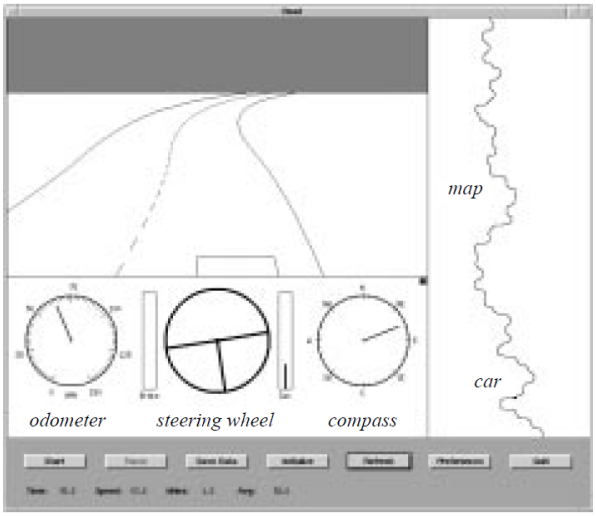


Figure 2.2: The driving simulator gives the user a perspective preview of the road ahead. The user has independent controls of the steering, brake, and accelerator (gas).

Takano et al. proposed an imitative learning of driving time series data for intellectual cognition toward future automobiles [26]. The driving pattern primitives consisting of states of the environment, vehicle and driver are symbolized by Hidden Markov Models (HMMs), which can be used for both recognition and generation of the driving patterns. The relationship among the HMMs can be represented by locating the HMMs in a multidimensional space. The contribution of each variable to the HMM space can be analysed such that important variables can be selected out of the driving data in order to reduce the size of the HMMs.

In their work left-to-right Hidden Markov Models is used to symbolize the driving patterns. The driving time series signal O consists of multi-modal data, such as environmental data, vehicle data and driver data. The driving time series signal is a sequence of driving patterns. So the measured signal has to be segmented into the driving patterns in order to symbolize the driving skills. They apply a segmentation method to the driving signal such that the driving pattern data O_k can be obtained.

The segmentation method is based on the criterion for prediction of following driving data. They adapted this segmentation method from their previous work. In that work they proposed a method for segmentation of human motions based on probabilistic correlation. First a human motion is divided into short sequences of motions. By encoding and compressing each short sequence using an HMM, a human motion

is represented by a sequence of notes corresponding to the HMMs. They then obtain a correlation matrix which represents the rules for appearances of the notes. Moments when the correlation is small are defined as boundaries of human patterns.

Lastly the parameters of HMMs are optimized using the segmented driving patterns by Baum- Welch algorithm. It should be noticed that each segmented driving pattern data is classified into an HMM which outputs the largest likelihood that the HMM generates the driving data, and the driving pattern data is used as a training data for the HMM. In this way, the HMMs are gradually optimized by using similar driving pattern data. These optimized HMMs extract features of the driving patterns. They named the HMMs *proto-symbols* since the HMMs form to the origins of symbols.

Moreover, in their work a hierarchical model is presented with the HMMs abstracting the primitive driving patterns in the lower layer, and another HMMs abstracting the long-term contextual driving patterns which are representation in the HMM space. Tests with a driving simulator and a actual vehicle demonstrate not only the validity of symbolization of driving pattern primitives, recognition and generation, but also availability of key feature selection. The extended hierarchical model is also proved to have a potential to predict the driving data appropriately.

In another attempt by Torkkola et al. they addressed the problem of driving sensor sequence modelling by drawing an analogy between driving manoeuvre recognition and speech recognition [16]. They assumed a driving manoeuvre typically consists of a sequence of actions that repeats with variations. These variations, as reflected in the sensor data stream, need to be captured in a sequential model. Speech recognition community has modelled the speech signal using stochastic graphical models, Hidden Markov Models (HMM). They took the same approach in modelling the sensor stream acquired from an automobile.

In their approach the sensor stream needs to be segmented in time into different context classes that are relevant to driving. The ”sentence” of driving should be segmented into ”words” of driving, that is, manoeuvres. However, no ”phonemes” exist for driving. Thus each different manoeuvre has to be modelled as a discrete entity with no shared parts. They attempted to discover such subunits for the purposes of modelling driving sensor data. They called these subunits ”drivemes”.

They defined drivemes as data units that occur as common patterns among the various driving manoeuvres. Since these are sequences on their own they could be modelled as separate Hidden Markov Models. The general idea was to model each

manoeuvre via HMMs and find the states or sequence of states that are common to the manoeuvres. These common states or state sequences represent the drivemes and can be used as building blocks to model the various manoeuvres of a driver.

The steps involved in learning the drivemes can be summarized as follows:

1. Collect driving data.
2. Annotate the data for different manoeuvres.
3. Build HMMs for each manoeuvre using the features/sensors that discriminate them the most.
4. Cluster the states of these HMMs. All those states that are similar will belong to the same cluster. These clusters will represent the common patterns among the various manoeuvre models.
5. Build tied state manoeuvre models using these cluster states. These clusters states or cluster state sequences that appear in common to more than one manoeuvre will characterize the various drivemes.

In a similar work by Colinon and Billard in the area of gesture recognition, they investigated the use of HMMs to both recognize and generate gestures, by extracting and learning the minimal set of features needed for the task, i.e. by selecting the optimal data representation. Their system uses known characteristics of human motion in a data preprocessing phase to extract the key-points [27]. Specifically, data are segmented at the inflexion points of the trajectory, in order to implicitly reflect the correlations across the joint angles. They tackled gesture learning as a data reduction problem, decomposed into a low-level processing part for segmenting the signal and a high-level processing part, that classifies the gestures using HMMs. To this end they considered a gesture as a sequence of typical events in the trajectory. Events are the inflexion points (i.e. local minima and maxima) of the joint angle trajectories. Such a segmentation aims at extracting known features of human motion.

For the reproduction of a gesture, the Viterbi algorithm is used to retrieve the best sequence of hidden states and the associated key-point values. The corresponding trajectory is then reconstructed by applying a 3rd-order spline fit when using the Cartesian trajectory, and by applying a cosine fit when using the joint angle trajectory. The cosine fit corresponds to a cycloidal velocity profile, and keeps the key-points as inflexion points during the reproduction.

In the same fashion that ”proto-symbols” or ”drivemes” are introduced, Hundtofte et al. introduced ”gestemes” to build a task language for segmentation and recognition of user input to cooperative manipulation systems [28]. They compared they work with previous work on recognition of user intent with man/machine interfaces which has used task-level HMMs with a single hidden state for each sub-task. But in contrast, many speech recognition systems employ HMMs at the phoneme level, and use a network of HMMs to model words. They analogously made use of multi-state, continuous HMMs to model action at the ”gesteme” level, and a network of HMMs to describe a task or activity.

They tried to answer two questions of interest:

1. Is it possible to model small portions of a task, which we term gestemes, using continuous HMMs applied to raw sensor data?

2. Can gestemes be generalized across activities? That is, is it possible to build a small vocabulary of gestemes that can be assembled to describe a much larger range of activities?

Their approach was to first develop multi-state HMMs that correspond to component actions, or ”gestemes” that make up a task. Then create a task-level system in which each task state corresponds to a gesteme in a one-to-one mapping. As a result, They were able to create a ”task language” that is used to model and segment two different tasks performed with a human-machine cooperative manipulation system. Experimental results show a recognition accuracy exceeding 85%. (Ref: Building a Task Language for Segmentation and Recognition of User Input to Cooperative Manipulation Systems)

Murphy et al. developed a system using hidden Markov models (HMMs) to recognize motions performed in a virtual environment with a haptic device. The output of this system was a list of motions used during completion of a task [29]. They first explored which observations were most important for accurate recognition of user motions. Then, they used a sequence of motions to evaluate skill by analysing the performance of individual users over multiple repetitions of a dynamic task. The results revealed that the system was able to achieve consistent recognition with a wide variety of observations.

Providing a navigation support by learning driving patterns was the problem which was addressed by Dejan Mitrovic [15]. He presented a simple and reliable method for

the recognition of driving events using hidden Markov models (HMMs). A data acquisition system was used to collect longitudinal and lateral acceleration and speed data from a real vehicle in a normal driving environment. Data were filtered, normalized, segmented, and quantified to obtain the symbolic representation necessary for use with discrete HMMs. Observation sequences for training and evaluation were manually selected and classified as events of a particular type. An appropriate model size was selected, and the model was trained for each type of driving events. Observation sequences from the training set were evaluated by multiple models, and the highest probability decides what kind of driving event this sequence represents. The recognition results showed that HMMs could recognize driving events very accurately and reliably. The proposed framework is given in Figure 2.3.

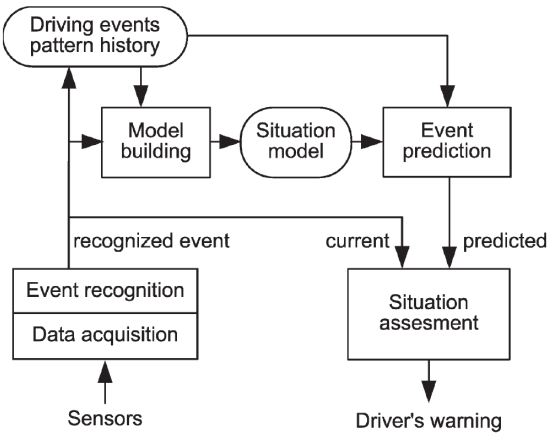


Figure 2.3: The framework of the driver warning system based on learning driving patterns.

Chapter 3

Theory and Background

The aim of this project is about modelling human driving behaviours. They are in result of how a driver makes decisions in response to environmental circumstances. Basically to be able to construct a model describing a system, we need to have some insight about the underlying (physical) process. Naturally a physical process can be either deterministic or stochastic. They come from two different areas with their own characteristics and properties. Due to inherently stochastic nature of human behaviour, the aim of this chapter is delivering some basic information about stochastic (or statistical or probabilistic) modelling approaches and related topics.

3.1 Time Series Data

A record of phenomenon irregularly varying with time is called time series. A time series typically consists of a set of observations on a variable, taken at equally spaced intervals over time.

There are two aspects to the study of time series - analysis and modelling. The aim of analysis is to summarise the properties of a series and to characterise its salient features. This may be done either in the time domain or in the frequency domain. In the time domain attention is focused on the relationship between observations at different points in time, while in the frequency domain it is cyclical movements which are studied. The two forms of analysis are complementary rather than competitive, The same information is processed in different ways, thereby giving different insights into the nature of the time series.

There is a wide variety of time series that can be classified into several categories

from various viewpoints: Continuous and discrete, Univariate and multivariate, Stationary and non-stationary, Gaussian and non-Gaussian, Linear and non-linear, Missing observations and outliers.

3.2 What is a model?

There are two typical (and rather different) views about the meaning of model [30]. One view is a *mechanistic* one, in which models elucidate the mechanism by which something happens. These models are very powerful but are also very difficult to create, often requiring years of experimental work and difficult intellectual insights. A different view of models considers them as *black boxes* and makes no claims that the mechanism of the model matches anything in the real world. In this approach a model is evaluated on the basis of its accuracy in prediction, not by the mechanism used. Making numerically accurate and fully mechanistic models is rarely possible in real world task.

When there is the need of analysing and modelling a database of sequences, the predictions that can be obtained by a black-box model are somewhat limited. The models we will focus on, namely Hidden Markov Models, fall somewhere between the extremes of mechanistic models and pure black-box models. They do not provide mechanistic explanations but they have an internal structure that can provide an insight into the characteristic dynamics of the modeled process. We will also see that this kind of structure can be easily modified, according to domain knowledge, in order to improve the model performances.

From a general point of view, a model can be used for three main purposes: describing the details of a process, predicting its outcomes, or for classification purposes, i.e., predicting a single variable k which takes values in a finite unordered set, given some input data $x = (x_1, , x_n)$. It is easy to understand that not all the models could perform well on all three kinds of tasks. When there is the need of modelling processes characterized by great complexity and affected by randomness, usually the main approach is to focus only on the aspects required for solving the task; in this way the computational complexity of the models can be controlled, in such a way to obtain practically useful ones. This idea led to different families of specialized models, for example models designed for classification, like the Support Vector Machines.

3.3 Statistical Modelling

Given a collection of variables, each variable being a vector of readings of a specific trait on the samples in an experiment, the problem is in what way does a variable Y depend on other variables $X_1, X_2, ..., X_n$ in the study. The solution is a statistical model defines a mathematical relationship between the X_i's and Y [31]. The model is a representation of the real Y that aims to replace it as far as possible. At least the model should capture the dependence of Y on the X_i's. The response variable is the one whose content we are trying to model with other variables, called the explanatory variables. In any given model there is one response variable, Y, and there may be many explanatory variables like $X_1, X_2, ..., X_n$.

In the statistical analysis of time series, measurements of a phenomenon with uncertainty are considered to be the realization of a random variable that follows a certain probability distribution. Time series models and statistical models, in general, are built to specify this probability distribution based on data. A basic criterion is needed always for evaluating the closeness between the true probability distribution and the probability distribution specified by a model.

Given a *random variable* Y, the probability that the event $Y \leq y$ occurs, $Prob(Y \leq y)$ can be defined for all real numbers $y \in R$. Considering this to be a function of y, the function of y defined by $G(y) = Prob(Y \leq y)$ is called the probability distribution function (or distribution function) of the random variable Y.

If a probability distribution or a density function is given, we can generate data that follow the distribution. On the other hand, in statistical analysis, when data $y_1, , y_N$ have been obtained, they are considered to be realizations of a random variable Y. That is, we assume a random variable Y underlying the data, and when we obtain the data, we consider them as realizations of that random variable. Here, the density function $g(y)$ defining the random variable is called the true model. Since this true model is usually unknown for us, given a set of data, it is necessary to estimate the probability distribution that generates the data. Here, the density function estimated from data is called a statistical model and is denoted by $f(y)$.

3.4 Hidden Markov Models

Hidden Markov Models (HMM) are stochastic methods to model temporal and sequence data. They are especially known for their application in temporal pattern recognition such as speech, handwriting, gesture recognition, part-of-speech tagging, musical score following, partial discharges and bioinformatics.

Hidden Markov Models were first described in a series of statistical papers by Leonard E. Baum and other authors in the second half of the 1960s. One of the first applications of HMMs was speech recognition, starting in the mid-1970s. Indeed, one of the most comprehensive explanations on the topic was published in "A Tutorial On Hidden Markov Models And Selected Applications in Speech Recognition", by Lawrence R. Rabiner in 1989 [32]. In the second half of the 1980s, HMMs began to be applied to the analysis of biological sequences, in particular DNA. Since then, they have become ubiquitous in the field of bioinformatics.

Dynamical systems of discrete nature assumed to be governed by a Markov chain emits a sequence of observable outputs. Under the Markov assumption, it is also assumed that the latest output depends only on the current state of the system. Such states are often not known from the observer when only the output values are observable.

Hidden Markov Models attempt to model such systems and allow, among other things, (1) to infer the most likely sequence of states that produced a given output sequence, to (2) infer which will be the most likely next state (and thus predicting the next output) and (3) calculate the probability that a given sequence of outputs originated from the system (allowing the use of hidden Markov models for sequence classification).

The "hidden" in Hidden Markov Models comes from the fact that the observer does not know in which state the system may be in, but has only a probabilistic insight on where it should be.

The aforementioned technique is the original use of HMMs to address problems of interest. They can be used in another manner, as a sequence classifiers in a classification problem. This is how we made use of HMMs in our current project. Distinct Hidden Markov Models can be trained individually on data obtained from each class of a classification problem. If we create a set of those models and specialize each model to recognize each of the separate classes, we can then use the HMMs ability to calculate the likelihood that a given sequence belongs to the model to determine the

most likely class of an unknown sequence.

Hidden Markov Models can be trained individually on data obtained from each class of a classification problem. If we create a set of models and specialize each model to recognize each of the separated classes, then we will be able to explore the HMMs ability to calculate the likelihood that a given sequence belongs to itself to perform classification of discrete sequences.

After all models have been trained, the probability of the unknown-class sequence can be computed for each model. As each model specialized in a given class, the one which outputs the highest probability can be used to determine the most likely class for the new sequence (Figure 3.1).

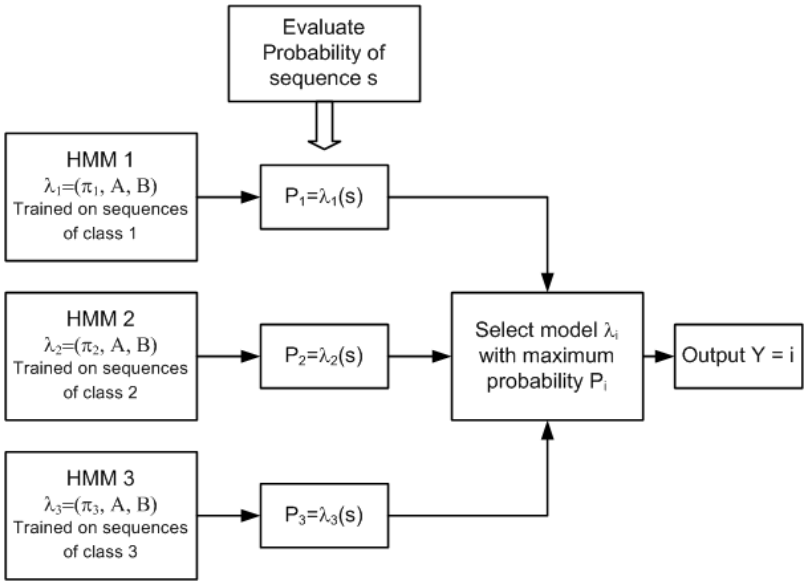


Figure 3.1: Schematic representation of the sequence classification procedure.

3.4.1 Discrete-Time Markov Model

Consider a system that may be described at any time being one of a set of N distinct states index by $1, 2, ..., N$. At regular spaced, discrete times, the system undergoes a change of state (possibly back to same state) according to a set of probabilities associated with the state. The time instances for a state change is denoted t and the actual state at time t as q_t. In the case of a first order Markov chain, the state transition probabilities do not depend on the whole history of the process, just the preceding state is

taken into account. This is the Markov property and is defined as:

$$P(q_t = j|q_{t-1} = i, q_{t-2} = k, ...) = P(q_t = j|q_{t-1} = i) \tag{3.1}$$

Also consider that the right hand of 3.1 is independent of time, which leads to a set of state transitions probabilities, a_{ij} , of the form:

$$a_{ij} = P(q_t = j|q_{t-1} = i), 1 \leq i, j \leq N \tag{3.2}$$

These state probabilities, a_{ij} , has the following properties (due to standard stochastic constrains):

$$\begin{aligned} a_{ij} &\geq 0 \quad \forall j, i \\ \sum_{j=1}^{N} a_{ij} &= 1 \quad \forall i \end{aligned} \tag{3.3}$$

The state transition probabilities for all states in a model can be described by a transition probability matrix:

$$A = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1N} \\ a_{21} & a_{22} & \dots & a_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ a_{N1} & a_{N2} & \dots & a_{NN} \end{bmatrix} \tag{3.4}$$

The only thing remaining to describe the system is the initial state distribution vector (the probability to start in some state). And this vector is described by:

$$A = \begin{bmatrix} \pi_1 = P(q_1 = 1) \\ \pi_2 = P(q_1 = 2) \\ \vdots \\ \pi_N = P(q_1 = N) \end{bmatrix} \tag{3.5}$$

The stochastic property for the initial state distribution vector is:

$$\sum_{i=1}^{N} \pi_i = 1 \tag{3.6}$$

Where the π_i is defined as:

$$\pi_i = P(q_1 = i), 1 \leq i \leq N \tag{3.7}$$

These are the properties and equations for describing a Markov process. The Markov model can be described by A and π.

3.4.2 Discrete-Time Hidden Markov Model

The discrete-time Markov model described in previous section is too restrictive to be applicable to many problems of interest. Therefore it will be extended to the discrete-time hidden Markov model. The extension done is that every state will now not be deterministic, instead it will be probabilistic. This means that every state generates a observation, o_t, according to some probabilistic function. The production of observations in this stochastic approach is characterized by a set of observation probability measures, $B = b_j(o_t)_{j=1}^N$, in which the probabilistic function for each state, j, is:

$$b_j(o_t) = P(o_t|q_t = j) \tag{3.8}$$

In general the discrete observation density HMMs are based on partitioning the probability density function (pdf) of observations into a discrete set of small cells and symbols $v_1, v_2, ..., v_M$ one symbol representing each cell. This partitioning subject is usually called vector quantization and there exists several analysis methods for this topic. After a vector quantization is performed, a codebook is created of the mean vectors for every cluster.

The corresponding symbol for the observation is determined by the nearest neighbour rule, i.e. select the symbol of the cell with the nearest codebook vector. The observation symbol probability distribution, $B = b_j(o_t)_{j=1}^N$ will now have the symbol distribution in state j, $b_j(o_t)$, defined as:

$$b_j(o_t) = b_j(k) = P(o_t = v_k|q_t = j), 1 \leq k \leq M \tag{3.9}$$

The estimation of the probabilities $b_j(k)$ is normally accomplished in two steps, first the determination of the codebook and then the estimation of the sets of observation probabilities for each codebook vector in each state. The major problem with discrete output probability is that the vector quantization operation partitions the continuous acoustic space into separate regions destroying the original signal structure.

As the output distributions of different states are typically highly overlapping the partitioning introduces small errors. The unreliability of parameter estimates due to the finiteness of the training data prevents the presentation of very large codebooks. This introduces quantization errors which may cause degradation in the discrete HMM performance when the observed feature vectors are intermediate between two codebook symbols. This is why continuous observation densities can be used instead to increase the recognition rate, but this approach also gives that more calculations is needed.

3.4.3 HMMs with Continuous Observation Densities

To create continuous observation density HMMs, $b_j(o_t)$s are created as some parametric probability density functions or mixtures of them. To be able to reestimate the parameters of the probability density function (pdf) some restriction must be placed on the model for the pdf. This restriction is that the pdf is any log-concave or elliptically symmetric density. The most general representation of the pdf, for which a reestimation procedure has been formulated, is a finite mixture of the form:

$$b_j(o_t) = \sum_{k=1}^{M} c_{jk} b_{jk}(o_t), j = 1, 2, ..., N \tag{3.10}$$

Where M is the number of mixtures and the following stochastic constraints for the mixture weights, c_{jk}, holds:

$$\begin{aligned} \sum_{k=1}^{M} c_{jk} &= 1 \qquad\qquad j = 1, 2, ..., N \\ c_{jk} &\geq 0 \quad j = 1, 2, ..., N, k = 1, 2, ..., M \end{aligned} \tag{3.11}$$

And $b_{jk}(o_t)$ is a D-dimensional log-concave or elliptically symmetric density with mean vector μjk and covariance matrix Σ_{jk}:

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, \Sigma jk) \tag{3.12}$$

The most used D-dimensional log-concave or elliptically symmetric density, is the Gaussian density. The Gaussian density can be found as:

$$b_{jk}(o_t) = N(o_t, \mu_{jk}, \Sigma jk) = \frac{1}{(2\pi)^{D/2}|\Sigma_{jk}|^{1/2}} e^{-\frac{1}{2}(o_t - \mu_{jk})^T \Sigma_{jk}^{-1}(o_t - \mu_{jk})} \tag{3.13}$$

To approximate simple observation sources, the mixture Gaussians provide an easy way to gain a considerable accuracy due to the flexibility and convenient estimation of the pdfs. If the observation source generates a complicated high dimensional pdf, the mixture Gaussians become computationally difficult to treat, due to the excessive number of parameters and large covariance matrices.

When huge number of parameters are to be estimated, the practical problem is the availability of the necessary amount of well representative training data. With insufficient training data some parameters will get more or less arbitrary values and especially for the covariance matrices, this may have drastical consequences.

3.4.4 Types of Hidden Markov Models

Different kinds of structures for HMMs can be used. The structure is defined by the transition matrix, A. The most general structure is the *ergodic* or fully connected HMM. In this model can every state be reached from every other state of the model. As shown in Figure 3.2a, for an $N = 4$ state model, this model has the property $0 < a_{ij} < 1$ (the zero and the one has to be excluded, otherwise is the ergodic property not fulfilled). The state transition matrix, A, for an ergodic model, can be described by:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \\ a_{N1} & a_{N2} & a_{43} & a_{44} \end{bmatrix} \tag{3.14}$$

In speech recognition, it is desirable to use a model which models the observations in a successive manner - since this is the property of speech. The models that fulfills this modelling technique, is the left-right model or Bakis model, see Figure 3.2b,c. The property for a left-right model is:

$$a_{ij} = 0, \ j < i \tag{3.15}$$

That is, no jumps can be made to a previous states. The lengths of the transitions are usually restricted to some maximum length, typical two or three:

$$a_{ij} = 0, \ j > i + \Delta \tag{3.16}$$

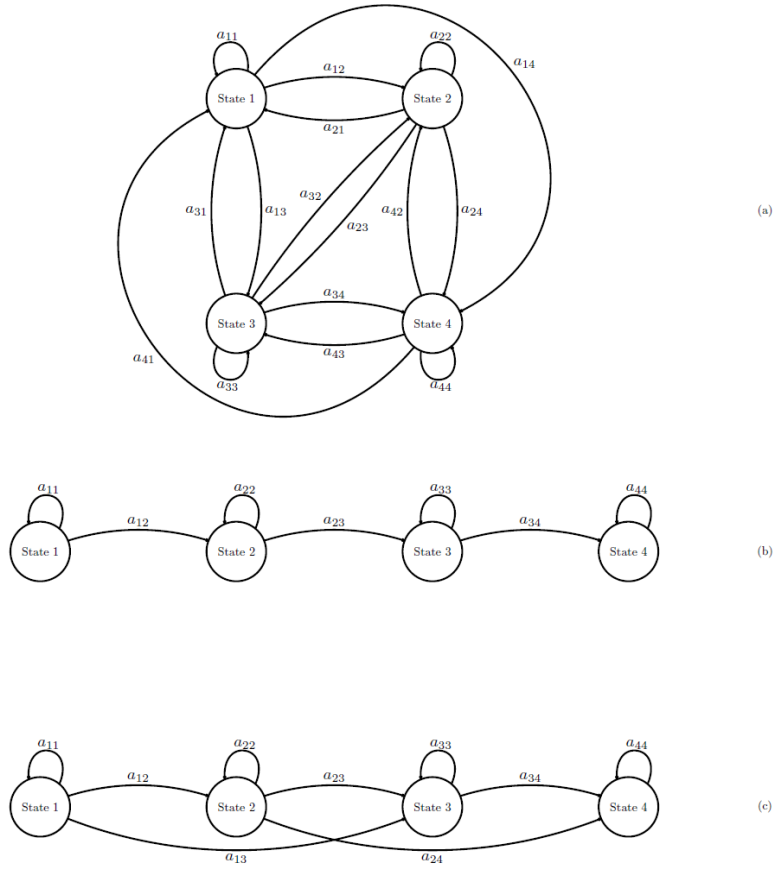


Figure 3.2: Different structures for HMMs

Note that, for a left-right model, the state transitions coefficients for the last state has the following property:

$$\begin{aligned} a_{NN} &= 1 \\ aNj &= 0,\ j < N \end{aligned} \tag{3.17}$$

In Figure 3.2b and Figure 3.2c two left-right models are presented. In Figure 3.2b is $\Delta = 1$ and the state transition matrix, A, will be:

$$A = \begin{bmatrix} a_{11} & a_{12} & 0 & 0 \\ 0 & a_{22} & a_{23} & 0 \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{3.18}$$

And in Figure 3.2c is $\Delta = 2$ is and the state transition matrix, A, will be:

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 \\ 0 & a_{22} & a_{23} & a_{24} \\ 0 & 0 & a_{33} & a_{34} \\ 0 & 0 & 0 & a_{44} \end{bmatrix} \tag{3.19}$$

The choice of constrained model structure like a left-right model requires no modification in training algorithms (described later in Problem 3 - Parameter Estimation). This because any state probability set to zero will remain zero in the training algorithms.

3.4.5 Summary of elements for an Hidden Markov Model

The elements of a discrete-time hidden Markov model will now be summarized.

- Number of states N. Although the states are hidden, for many practical applications there is often some physical significance attached to the states or to sets of states of the model. The labels for the individual states is $1, 2, ..., N$, and the state at time t is denoted q_t.
- Model parameter M. If discrete observation densities are used, the parameter M is the number of classes or cells that should be used. If continuous observation densities are used, M is represented by the number of mixtures in every state.
- Initial state distribution $\pi = \pi_i{}_{i=1}^{N}$, in which π_i is defined as:

$$\pi_i = P(q_1 = i) \tag{3.20}$$

- State transition probability distribution $A = [a_{ij}]$ where:

$$a_{ij} = P(qt + 1 = j | q_t = i),\ 1 \leq i, j \leq N \tag{3.21}$$

- Observation symbol probability distribution, $B = b_j(o_t)_{j=1}^{N}$, in which the probabilistic function for each state, j, is:

$$b_j(o_t) = P(o_t | q_t = j) \tag{3.22}$$

  The calculation of $b_j(o_t)$ can be found with discrete- or continuous observation densities. In this thesis, the continuous observation densities are used.

It should now be clear that a complete specification of an HMM requires two model parameters N and M. The specification of the three sets of probability measures π, A and B are also necessary. For convenience will these probability measures use the notation, λ:

$$\lambda = (A, B, \pi) \tag{3.23}$$

3.4.6 Three Basic Problems for Hidden Markov Models

Given the basics of an HMM from the previous section, three basic problems arise for applying the model in a speech recognition task:

- **Problem 1: Evaluation**: Given the observation sequence $O = (o_1, o_2, ..., o_T)$, and the model $\lambda = (A, B, \pi)$, how is the probability of the observation sequence, given the model, computed? That is, how is $P(O|\lambda)$ computed efficiently?
- **Problem 2: Decoding**: Given the observation sequence $O = (o_1, o_2, ..., o_T)$, and the model $\lambda = (A, B, \pi)$, how is a corresponding state sequence, $q = (q_1, q_2, ..., q_T)$, chosen to be optimal in some sense (i.e., best "explains" the observations)?
- **Problem 3: Learning**: How are the probability measures, $\lambda = (A, B, \pi)$, adjusted to maximize $P(O|\lambda)$?

The first problem can be seen as the recognition problem. With some trained models, each model represents a word, which model is the most likely if an observation is given (i.e. what word is spoken)? In the second problem the hidden part of the model is attempted to be uncovered. It should be clear, that for all except the case of degenerated models, there is no "correct" state sequence to be found. Thereby it is a problem to be solved best possible with some optimal criteria. The third problem can be seen as the training problem. That is given the training sequences, create a model for each word. The training problem is the crucial one for most applications of HMMs, because it will optimally adapt the model parameters to observed training data - i.e., it will create the best models for real phenomena.

3.4.7 Extensions to HMMs

Hidden Markov models have been applied successfully in several areas and a number of extensions has been proposed. Some of these extensions is briefly described in the following.

Structured Hidden Markov Models: It is sometimes useful to use HMMs in specific structures in order to facilitate learning and generalization. For example, even though a fully connected HMM could always be used if enough training data is available it is often useful to constrain the model by not allowing arbitrary state transitions. In the same way it can be beneficial to embed the HMM into a greater structure; which, theoretically, may not be able to solve any other problems than the basic HMM but can solve some problems more efficiently when it comes to the amount of required training data.

Layered Hidden Markov Model: A layered hidden Markov model (LHMM), (Oliver et al., 2004) consists of N levels of HMMs where the HMMs on level $N+1$ corresponds to observation symbols or probability generators at level N. Every level i of the LHMM consists of K_i HMMs running in parallel, see Figure 3.3.

At any given level L in the LHMM a sequence of T_L observation symbols $o_L = \{o_1, o_2, ..., o_{T_L}\}$ can be used to classify the input into one of K_L classes, where each class corresponds to each of the K_L HMMs at level L. This classification can then be used to generate a new observation for the level $L1$ HMMs. At the lowest layer, i.e. level N, primitive observation symbols $o_p = \{o_1, o_2, ..., o_{T_p}\}$ would be generated directly from observations of the modelled process. For example in a trajectory tracking task the primitive observation symbols would originate from the quantized sensor values. Thus at each layer in the LHMM the observations originate from the classification of the underlying layer, except for the lowest layer where the observation symbols originate from measurements of the observed process.

It should be noted here that it is not necessary to run all levels at the same time granularity. For example it is possible to use windowing at any level in the structure so that the classification takes the average of several classifications into consideration before passing the results up the layers of the LHMM.

Instead of simply using the winning HMM at level $L+1$ as an input symbol for the HMM at level L it is possible to use it as a probability generator by passing the complete probability distribution up the layers of the LHMM. Thus instead of having a

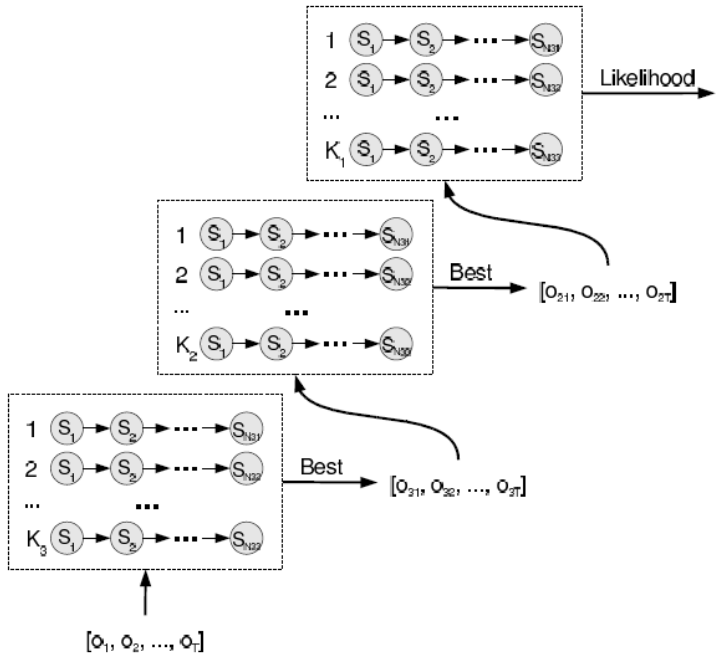


Figure 3.3: A layered hidden Markov model

"winner takes all" strategy where the most probable HMM is selected as an observation symbol, the likelihood $L(i)$ of observing the ith HMM can be used in the recursion formula of the level L HMM to account for the uncertainty in the classification of the HMMs at level $L+1$. Thus, if the classification of the HMMs at level $n+1$ is uncertain, it is possible to pay more attention to the a-priori information encoded in the HMM at level L.

It should be noted here that a LHMM could in practice be transformed into a single layered HMM where all the different models are concatenated together. Some of the advantages that may be expected from using the LHMM over a large single layer HMM is that the LHMM is less likely to suffer from over-fitting since the individual sub-components are trained independently on smaller amounts of data. A consequence of this is that a significantly smaller amount of training data is required for the LHMM to achieve a performance comparable of the HMM. Another advantage is that the layers at the bottom of the LHMM, which are more sensitive to changes in the environment such as the type of sensors, sampling rate etc, can be retrained separately without altering the higher layers of the LHMM.

Hierarchical Hidden Markov Models In the Hierarchical Hidden Markov Model

(HHMM) each state is considered to be a self contained probabilistic model. More precisely each state of the HHMM is itself a HHMM. This implies that the states of the HHMM emits sequences of observation symbols rather then single observation symbols as is the case for the standard HMM states.

When a state in a HHMM is activate, it will activate its own probabilistic model, i.e. it will activate one of the states of the underlying HHMM, which in turn may activate its underlying HHMM and so on. The process is repeated until a special state, called a production state, is activated. Only the production states emit observation symbols in the usual HMM sense. When the production state has emitted a symbol, control returns to the state that activated the production state. The states that does not directly emit observations symbols are called internal states. The activation of a state in an HHMM under an internal state is called a *vertical transition*. After a vertical transition is completed a *horizontal transition* occurs to a state within the same level. When a horizontal transition leads to a terminating state control is returned to the state in the HHMM, higher up in the hierarchy, that produced the last vertical transition.

Remember that a vertical transition can result in more vertical transitions before reaching a sequence of production states and finally returning to the top level. Thus the production states visited gives rise to a sequence of observation symbols that is produced by the state at the top level. The structure of the HHMM is shown in Figure 3.4.

The methods for estimating the HHMM parameters and model structure are more complex than for the HMM and the interested reader is referred to (Fine et al., 1998).

It should be pointed out that the HMM, HHMM and LHMM all belong to the same class of classifiers. That is, they can be used to solve the same set of problems. In fact, both the HHMM and LHMM can be transformed into a standard HMM. However, both HHMMs and LHMMs utilize their structure to solve a subset of the problems more efficiently.

3.4.8 Implementation

There are a few ready-to-use toolkits or packages to implement hidden Markov models using them. They are generally support HMMs with discrete observations and amongst them we can find some functionalities to support HMMs with continuous observations.

HTK: The Hidden Markov Model Toolkit (HTK) is a portable toolkit for building and manipulating hidden Markov models. HTK is primarily used for speech recog-

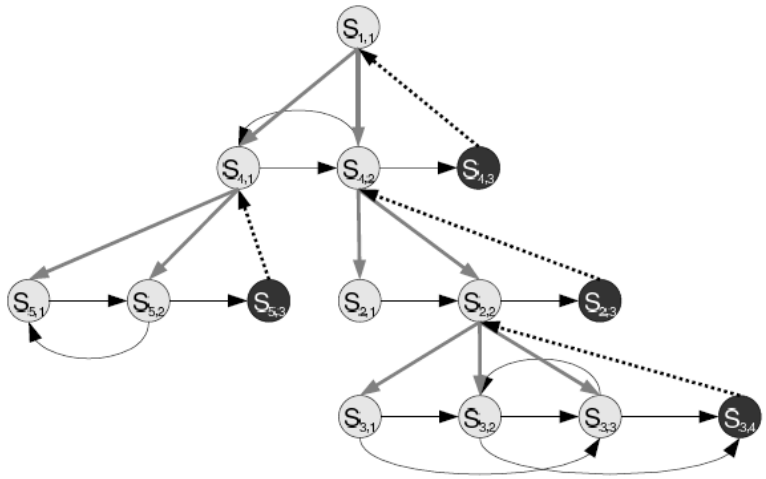


Figure 3.4: Illustration of the structure of a HHMM. Gray lines shows vertical transitions. The horizontal transitions are shown as black lines. The light gray circles are the internal states and the dark gray circles are the terminal states that returns control to the activating state. The production states are not shown in this figure.

nition research although it has been used for numerous other applications including research into speech synthesis, character recognition and DNA sequencing. HTK is in use at hundreds of sites worldwide.

HTK consists of a set of library modules and tools available in C source form. The tools provide sophisticated facilities for speech analysis, HMM training, testing and results analysis. The software supports HMMs using both continuous density mixture Gaussians and discrete distributions and can be used to build complex HMM systems. The HTK release contains extensive documentation and examples.

HTK was originally developed at the Machine Intelligence Laboratory (formerly known as the Speech Vision and Robotics Group) of the Cambridge University Engineering Department (CUED) where it has been used to build CUED's large vocabulary speech recognition systems (see CUED HTK LVR). In 1993 Entropic Research Laboratory Inc. acquired the rights to sell HTK and the development of HTK was fully transferred to Entropic in 1995 when the Entropic Cambridge Research Laboratory Ltd was established. HTK was sold by Entropic until 1999 when Microsoft bought Entropic. Microsoft has now licensed HTK back to CUED and is providing support so that CUED can redistribute HTK and provide development support via the HTK3 web site.

Hidden Markov Model (HMM) Toolbox for Matlab: This toolkit offers the most convenient way to handle HMMs both with discrete and continuous observations. It is written by Kevin Murphy in 1998 and last updated in 2005. It is distributed under the MIT License. This toolbox supports inference and learning for HMMs with discrete outputs (dhmm's), Gaussian outputs (ghmm's), or mixtures of Gaussians output (mhmm's). The Gaussians can be full, diagonal, or spherical (isotropic). It also supports discrete inputs, as in a POMDP. The inference routines support filtering, smoothing, and fixed-lag smoothing.

GHMM: General Hidden Markov Model library: The General Hidden Markov Model library (GHMM) is a freely available C library implementing efficient data structures and algorithms for basic and extended HMMs with discrete and continuous emissions. It comes with Python wrappers which provide a much nicer interface and added functionality. The GHMM is licensed under the LGPL. Some of its features are: Discrete and continuous emissions, Mixtures of PDFs for continous emissions, Non-homogenous Markov chains, Pair HMMs, Clustering and mixture modelling for HMMs, Graphical Editor HMMEd, Python bindings XML-based file format, Portable (autoconf, automake).

jHMM: jHMM is a library and application for handling Hidden Markov Models that is written in Java. The models are encoded in a simple XML format that allows you to have arbitrary states, transitions, and alphabets. The library implements sequence state predictions and Baum-Welch training of the model parameters. jHMM is licensed and distributed under the terms of the BSD License.

EM for Hidden Markov Models: This is a software package written in Matlab by Zoubin Ghahramani, Professor of Information Engineering in Department of Engineering at University of Cambridge. It is developed for modelling time series data. It supports both real-valued Gaussian observations or discrete-valued observations.

Jahmm: Jahmm is a Java implementation of Hidden Markov Model (HMM) related algorithms. It's been designed to be easy to use (e.g. simple things are simple to program) and general purpose. It is available under the new BSD license.

This library is reasonably efficient, meaning that the complexity of the implementation of the algorithms involved is that given by the theory. However, when a choice must be made between code readability and efficiency, readability has been chosen. It is thus ideal in research (because algorithms can easily be modified) and as an academic tool (students can quickly get interesting results).

It gives an implementation of the Viterbi, Forward-Backward, Baum-Welch and K-Means algorithms, among others.

Matlab: In the Matlab there is no built-in complete package available to deal with HMMs. It supports only discrete emissions case with limited functionalities. In the implementation the developers considered a hidden Markov model (HMM) is one in which you observe a sequence of emissions, but do not know the sequence of states the model went through to generate the emissions. Analyses of hidden Markov models in Matlab seek to recover the sequence of states from the observed data.

Analyzing hidden Markov models in Matlab covers 1) Generating a Test Sequence, 2) Estimating the State Sequence, 3) Estimating Transition and Emission Matrices, 4) Estimating Posterior State Probabilities, 5) Changing the Initial State Distribution.

Chapter 4

Experimental Set-up, Data Acquisition, Feature Extraction

To study human driving behaviours, firstly we need some driving signals. Observable driving signals can be categorized into three groups [33]: ”1) driving behaviour, e.g, accelerator (gas) and brake pedal pressures (positions) and steering wheel angles; 2) vehicle status, e.g., velocity, acceleration, and engine speed; and 3) vehicle position, e.g., following distance, relative lane position, and yaw angle (heading). Amongst them we focus on accelerator pedal position and steering wheel angle and velocity and heading of the vehicle.” As we decided to study driving behaviours using continuous hidden Markov models, we needed some realizations of them as the mentioned driving time-domain signals.

This chapter organized as follows: first an overview of the proposed system for recognizing human driving behaviours is given. Then the customized driving simulator as the experimental set-up is introduced. In the next section the data acquisition process is provided. In the last section the idea behind feature extraction from driving signals is given.

4.1 Overview

As it is mentioned earlier in chapter 3, there are two ways to make use of HMMs in typical applications: 1) as a (hidden) state observer, and 2) as classifiers in a classification problem. In the former type only one HMM is trained for the whole process and by studying the state transitions and state sequence (hidden path), a relation between

observations and hidden states is built. It is usually hired when a unique and specific task is under supervision. For instance assume an application in which we intend to train a robotic arm by human demonstration. In this case the given task is performed by a human adept operator, then an HMM is trained to model his performance, and finally the trained HMM is used by robotic arm to follow the same patterns human operator did. Here we mostly focus on hidden states, their characteristics, and their relation with observations. The usage of second type usually happens in classification problems where we have several classes and for each class one HMM is trained to be used later as belonging class recognizer. Distinct Hidden Markov Models can be trained individually on data obtained from each class of a classification problem. If we create a set of those models and specialize each model to recognize each of the separate classes, we can then use the HMMs ability to calculate the likelihood that a given sequence belongs to the model to determine the most likely class of an unknown sequence. In this project, we trained HMM-based classifier to recognize driving behaviours. HMMs are naturally suitable tools to model driver behaviour for the following reasons [1]:

- "HMMs can model the stochastic nature of the driving behaviour, providing sufficient statistical smoothing while offering effective temporal modelling,
- The variations in the driving signals across the drivers can be modelled (driver identification) or suppressed (driver-independent route models) according to the requirements of the desired task."

For being able to make use of HHMs in our driving application, we proposed a system from collecting driving signals to training and recognising which is shown in Figure 4.1.

First driving (time-series) signals are acquired through a simulated driving task performed by a subject driver. Appropriate pre-processing algorithms such as low-pass filter and triangular sliding-average smooth are applied to increase the ratio of signal to noise. Then translational acceleration and rotational velocity as the two most important features are extracted from velocity and heading signals respectively. In the segmentation, the local extrema of velocity and heading signals, correspond to the moments translational acceleration and rotational velocity cross the zero axes, are used for dividing driving signals into a number of segments. Every time a crossing zero is detected, it is marked as the beginning of a new segment. By the end of this process

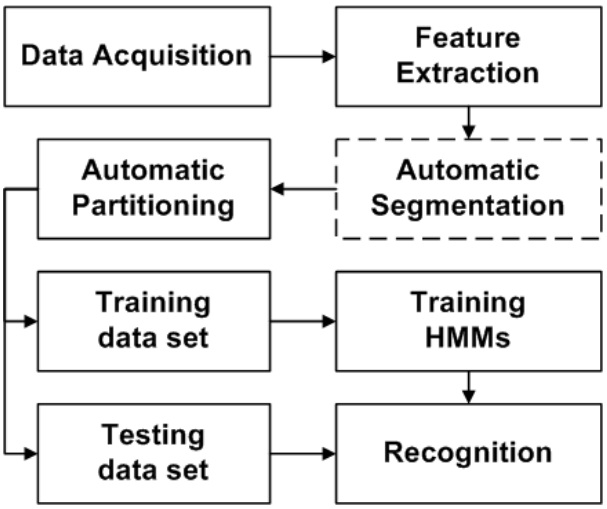


Figure 4.1: Proposed modelling approach

the boundaries of segments are determined. Now we need a way to categorize these segments. Various criteria can be hired for this purpose. We are going to evaluate three of them: threshold-based, GMM-based and hierarchical partitioning. Theses methods not only can be applied on data segments but also on data samples. The result of partitioning is a set of partitions or classes which similar behaviours (driver's decisions or driving rules), either data segments or data samples, are collected together. Now we are ready to train the HMMs. For each class one HMM is trained according to the data collected in each class. The parameters of HMM are optimized by the famous method of expectation maximization, called Baum-Welch method. After training, we are being able to evaluate their performance in response to unknown driving behaviour. For this end, first a rectangular sliding window are applied onto testing data set, then feed into each trained HMM. The one with highest likelihood measure is chosen as the belonging class. If both feeding test data class and recognized class are the same, then the recognition is done accurately.

4.2 Experimental Set-up

The experimental set-up is designed in a way to collect several driving signals through a simulated driving task. The building-block of the system is given in Figure 4.2. Racing wheel acts as an interface between subject driver and simulated virtual environment, acquires driving commands (pedal and wheel operations) from driver and

sends to graphic and physics engines. The graphic engine is updated every time a new command is read from racing wheel. In return it passes some of them into physics engine to apply the right physics for the vehicle. The physics engine itself updates some variables in graphic engine and the final outcome is sent to display to update the scene for the driver.

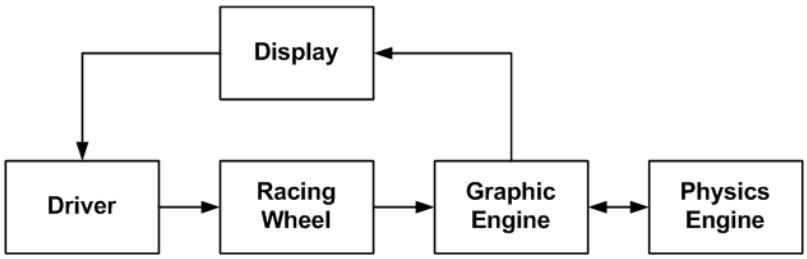


Figure 4.2: Overview of the experimental set-up

The driving simulator is based on Logitech G27 Racing Wheel force feedback device (Fig. 4.3). The whole package consists of a driving wheel, acceleration, brake and clutch pedals, and gear shifter. In addition a real adjustable driving seat is incorporated to the simulator to provide the sense and feeling of sitting in a real car (Figure 4.4).

Figure 4.3: Logitecg G27 Racing Wheel

The Irrlicht Engine version 1.7.2 is used as the core of GUI (Figure 4.5). The Ir-

Figure 4.4: Driving Simulator

rlicht Engine is a cross-platform high performance realtime 3D engine written in C++. It features a powerful high level API for creating complete 3D and 2D applications such as games or scientific visualizations. It integrates all state-of-the-art features for visual representation such as dynamic shadows, particle systems, character animation, indoor and outdoor technology, and collision detection. All this is accessible through a well designed C++ interface, which is extremely easy to use.

To implement the physics of the vehicle, another open-source solution named Newton Game Dynamics version 1.5.3 is used. The API provides scene management, collision detection, dynamic behaviour and yet it is small, fast, stable and easy to use for real time simulation of physics environments. The engine implements a deterministic solver, which is not based on traditional LCP or iterative methods, but possesses the stability and speed of both respectively. This feature makes the product a tool not only for games, but also for any real-time physics simulation. Its implementation is done through a wrapper, IPhysics, specially designed to be compatible with Irrlicht Engine.

Two simulation roads were used in experiments: small road (Figure 4.6), and big road (Figure 4.7). They are designed in Autodesk 3ds Max 2009. They consist of several left-hand and right-hand curves with some straight lanes. Some statistics about

Figure 4.5: GUI

the roads are given in following Figure 4.8.

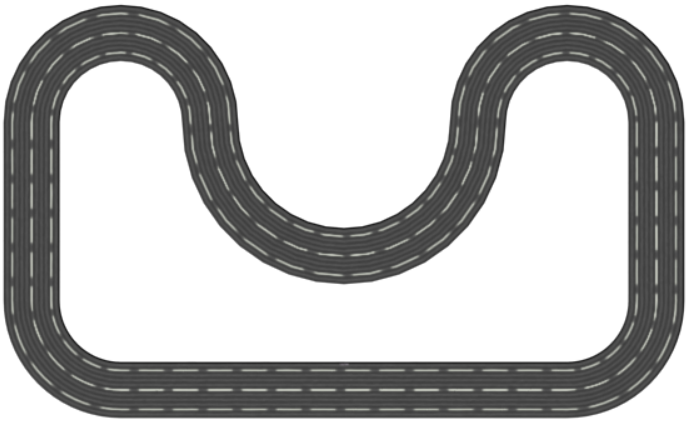


Figure 4.6: Simulated driving small road

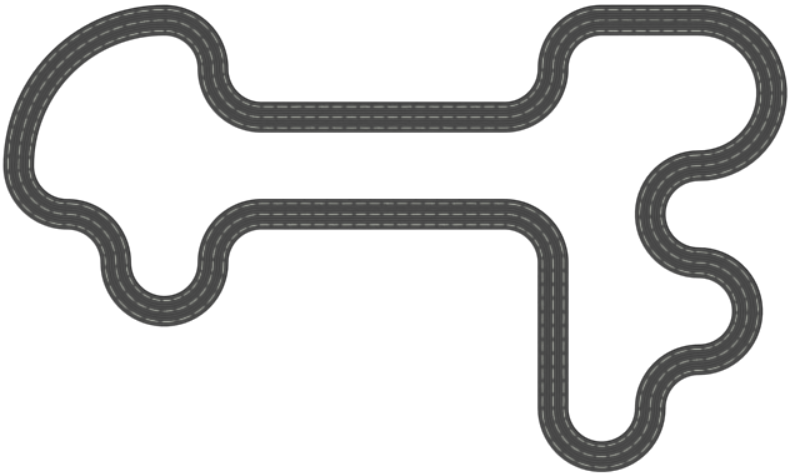


Figure 4.7: Simulated driving big road

Parameters	Small road	Big road
Total length (m)	462.735	1022.545
Straight lane (m)	180	300
Right-hand curve (m)	188.49	471.225
Left-hand curve (m)	92.245	251.32
Right-hand curve (degree)	540	1080
Left-hand curve (degree)	180	720

Figure 4.8: Statistics of the roads

4.3 Data Acquisition

The driving task is simply driving the vehicle along the simulated road using the driving simulator. The driving goal is following center-line of the road while driving not too slow and not too fast. In addition it is told to the subject driver to increase the speed of the car as mush as he can when he is approaching a straight lane. The subject driver has control on only accelerator (gas) pedal and steering wheel. There is no need to change the gear using the gear shifter and clutch pedal simultaneously, and even use the brake pedal. It is assumed the speed of the vehicle is increased by pushing the gas pedal and decreased by releasing the gas pedal. Due to the nature of our driving task, there is no other vehicles which needs the car stopped suddenly, that makes the need

fro brake pedal. In addition there is no intersection to stop behind the line. So without losing the generality, we just ignored the brake pedal, instead by releasing the gas pedal, somehow similar effect, i.e. decreasing the speed, is implemented. When a subject driver seats behind the steering wheel for a first time, he/she drives several trials to get familiar with the road and driving operations. Then during his real performance, some driving signals are collected as follows:

- Time: the time interval in msec.
- Acceleration pedal position: its original value is normalized in a way having 0.0 means no push and having 1.0 means maximum push. This normalized value also is passed to physics engine.
- Steering wheel angle: its original value is normalized in a way having 0.0 means no turn, having -1.0 means maximum left-hand turn, and having +1.0 maximum right-hand turn. This normalized value also is passed to physics engine.
- Velocity: it is calculated through the physics engine.
- Heading: orientation of the car.
- 2d position coordinate: the 2d (X, Z) vector is sampled ans stored individually.

A plot of collected driving signals along the big road is given in Figure 4.9. For being clear, just a range of 2000 samples (x-axis) are highlighted in the plots. The data is sampled inside a loop in the driving simulator program. If in each loop execution the sample is recorded in that case the average sampling rate is about 1 msec. That is too much for us, we actually do not need that kind of fast sampling rate, due to slow re-action of human kind. So we just stored 1 sample from 100 samples, that means we decreased the sampling rate to each 100 msec or 10 hertz. Acceleration pedal position and steering wheel angle directly is read from the interface provided by Irrlicht Engine to manage a joystick. Correspond to the value of steering wheel angle, the vehicle is rotated to meet the new rotation, and in return by calling the getRotation() function the current heading of the vehicle is stored. But the velocity is treated in different manner and by physics engine. We pass the value of acceleration pedal position to physics engine, then according to the physics engine, corresponding velocity is calculated and returned.

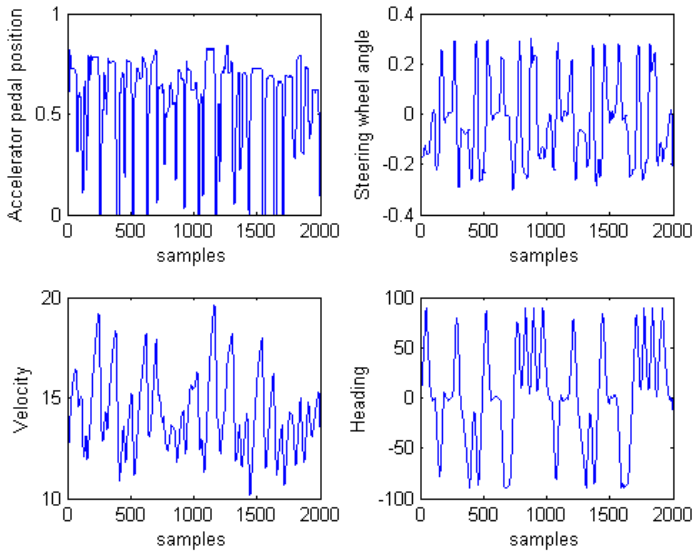


Figure 4.9: Driving signals: Accelerator pedal position, steering wheel angle, velocity, heading.

As it is mentioned, the driving task is repeated continuously without any stop in the beginning of the road, for several times. In case of the expert driver, because we are going to rely on his data, this process is done a few times in different days to collect as much as possible driving signals from his performance. In total 78 trials are stored from his performance in big road. Some statistics about collected driving trials is given in Figure 4.10.

It should be noticed that Logitech G27 has a setting window (Figure 4.11) to configure some parameters of the racing wheel. Amongst them we can name degrees of rotation and centring spring strength. The first parameter is adjusted to 720 degrees and second one to 60%.

4.4 Feature Extraction

One of the hardest tasks of pattern recognition is to identify the features of the pattern that would allow some computer system to recognize it. Therefore, selecting good features in any pattern recognition system plays significant role in system's performance

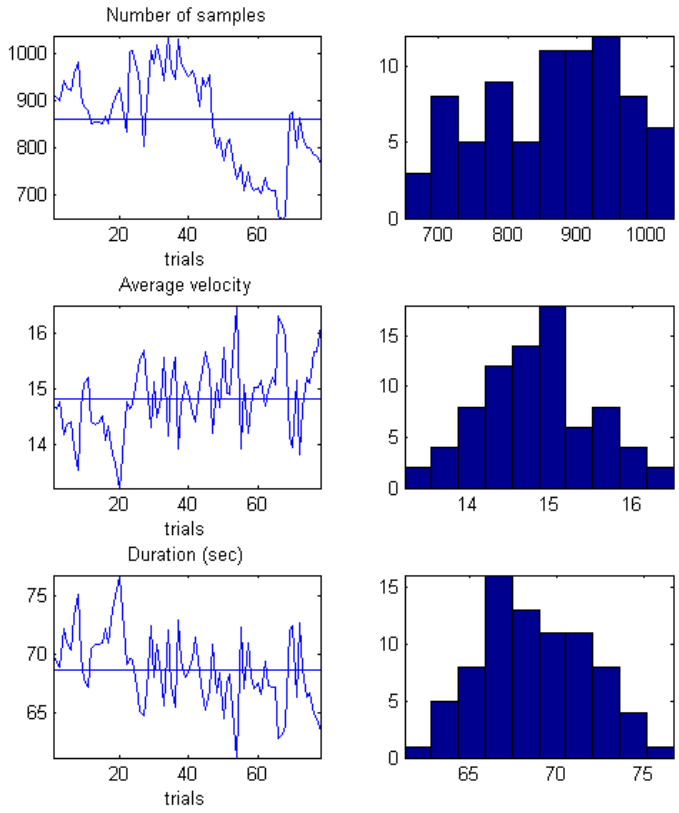


Figure 4.10: Statistics of the 78 trials of the expert driver's performance. Correspond to each plot in left, its histogram is given in right. The horizontal line in plots of left, denoting the average value of the variable.

as it reduces the system's complexity and simultaneously allows high recognition results. Moreover, the features are completely connected to the algorithm selected to perform recognition [34].

Our feature extraction methodology is inspired by what actually is happening in a real driving operation. In a real driving task, the driver should take care of various variables. The source of these variable can be the vehicle itself such as sound of engine, or the surrounding environment, like the shape of road, distance to side lanes, distance to other vehicles. Each of these variables needs an appropriate action. For example by hearing the laud sound of engine, the driver realizes it is time to change the gear or in case of any turning left/right, the relative alarm signal should be turned

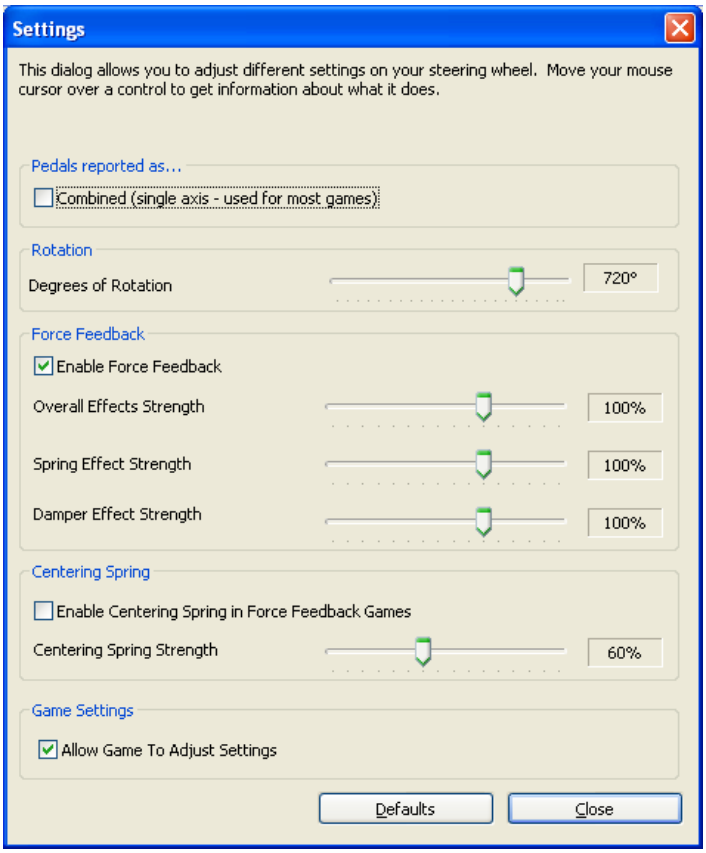


Figure 4.11: G27 Racing Wheel setting

on. In current project, we do not want to focus on these kinds of driving commands, i.e. changing gear, activating alarm signals. The most important behaviour for us in current scope of the project is pedal and wheel operations. We draw our attention to these most important and vital variables.

Intuitively inspired by real driving situation, we believe that the two most important features are the one which affect the velocity and heading. In another sense we suggest that every time a change in velocity and heading is occurred, that means the driver changed his/her driving commands. We can say ”monitoring those changes” is equivalent to ”monitoring the changes in driver’s intentions”. So from a mathematical standpoint, the first-order derivative of mentioned signals, will give us the time-steps in which a change is occurred. In other words we extracted some features from original signals that best reflect the current driving status. As heading is a measure of distance its first-order derivative will give a measure of velocity, we name it *rotational veloc-*

ity (rVel), in the same fashion the first-order derivative of velocity of the vehicle will return acceleration, which we named it *translational acceleration (tAcc)*. By studying these features we are being able to find more about the situations a driver tries to keep or change the driving commands. A plot of extracted features correspond to velocity and heading signals is given in Figure 4.12.

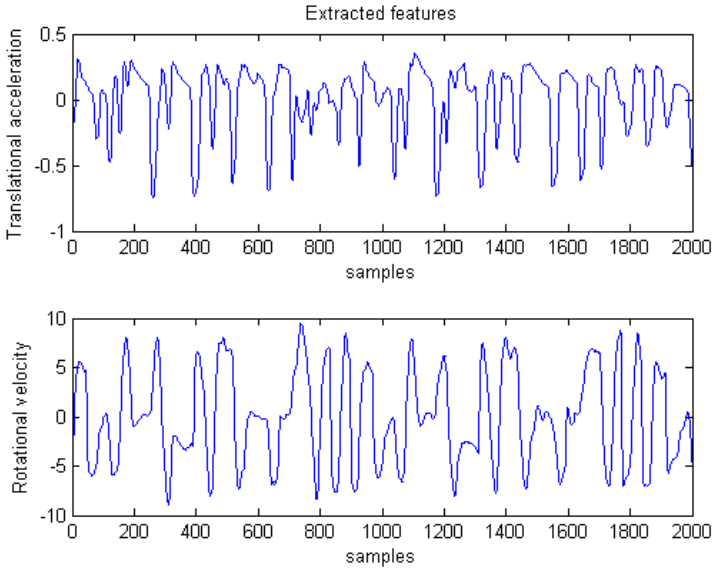


Figure 4.12: Extracted features: translational acceleration, rotational velocity.

To numerically implement the first-order derivative the method of *Central Estimate* is employed 4.13. It is a method of "finite difference" which approximates the tangent to the curve in point t (green line) with something easy, much more convenient in calculation, the secant to the curve. There are three options available around point t which bring three different methods: forward estimate (blue line), backward estimate (red line), and central estimate (purple line). Due to more accurate estimation of the method of central estimate, we used it in our computations. It is implemented through the following formula:

$$\frac{df(t)}{dt} \simeq \frac{f(t+h) - f(t-h)}{2.h}$$

Right after it, a low-pass filter with $RC = 0.1$ is applied, then a triangular smoothing method with smooth width $m = 5$ is applied on the resulted signal. The final

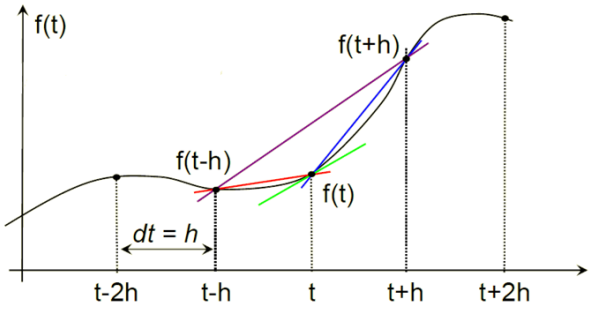


Figure 4.13: The method of ”Central Estimate” to numerically calculated first-order derivative.

outcome, is what we refer them as tAcc and rVel.

Chapter 5

Automatic Segmentation and Partitioning

We extracted two features namely translational accelerations (tAcc) and rotational velocity (rVel) from velocity and heading signals respectively, which best represents driver's intentions in terms of pedal and wheel operation. From the fact that the driver has control on just velocity and heading of the vehicle, we conclude that every time a change in their first-order derivative or better to say extracted features is detected, that means the driver changed his driving operations, in other words, he made different decision than before. So the time-steps equivalent to those changes are of our interest. We detect those time-steps and we suggest that the driver's behaviour between two consecutive steps are constant and remains unchanged. According to them we can divide driving signals into a number of segments. We believe not all segments come from different source, so maybe there are some ways to group similar segments, which we refer to them as partitioning methods.

5.1 Automatic Segmentation

According to the extracted features now we are being able to divide driving data into a number of segments. We suggest that the beginning of a new segment is the time-step that either a major change in velocity or heading is happened. By major change it means the sign of their growth rate is changed. That means we are looking for local extrema of the velocity and heading signals. In another sense, every time one of the extracted features crosses the zero axes, a new segment is begun. By this assumption

we can separately find the segments boundaries according to each feature and then by taking their union, we will have the final segmentation outcome.

For the purpose of implementation, first the time-steps which local extrema of velocity and heading signals, are calculated separately. Those time-steps occur when their first-order derivative cross the x-axis. The result of segmentation based on velocity (or tAcc) is given in Figure 5.1. The same process is applied to heading (or rVel). The result of segmentation based on rVel is given in Figure 5.2. To the plots be clear and easy-to-read, we chose a small range of 500 samples to depict them. The final outcome which is in result of combination of segmentation based on both tAcc and rVel is given in Figure 5.3.

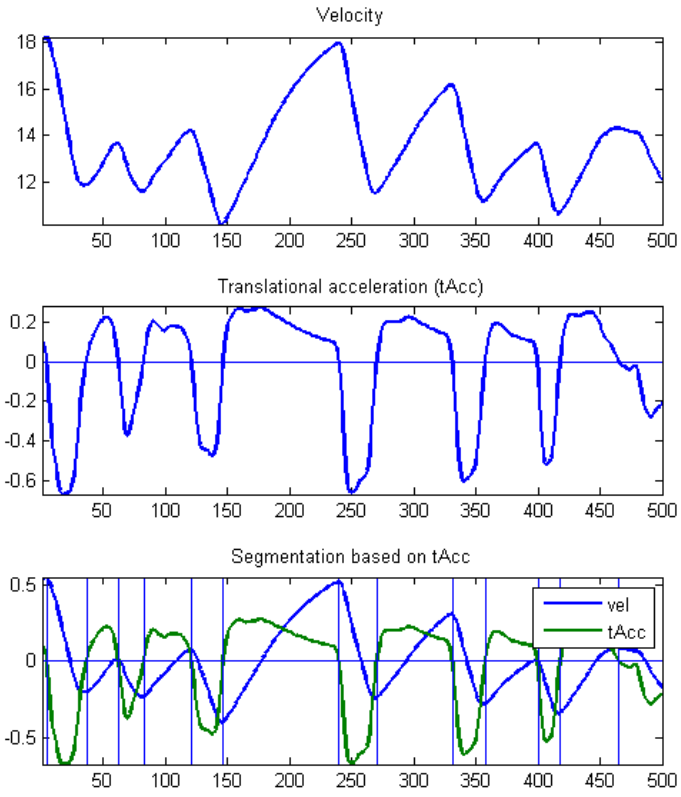


Figure 5.1: Segmentation based on tAcc

It should be noticed that once the boundaries of the segments are determined it can

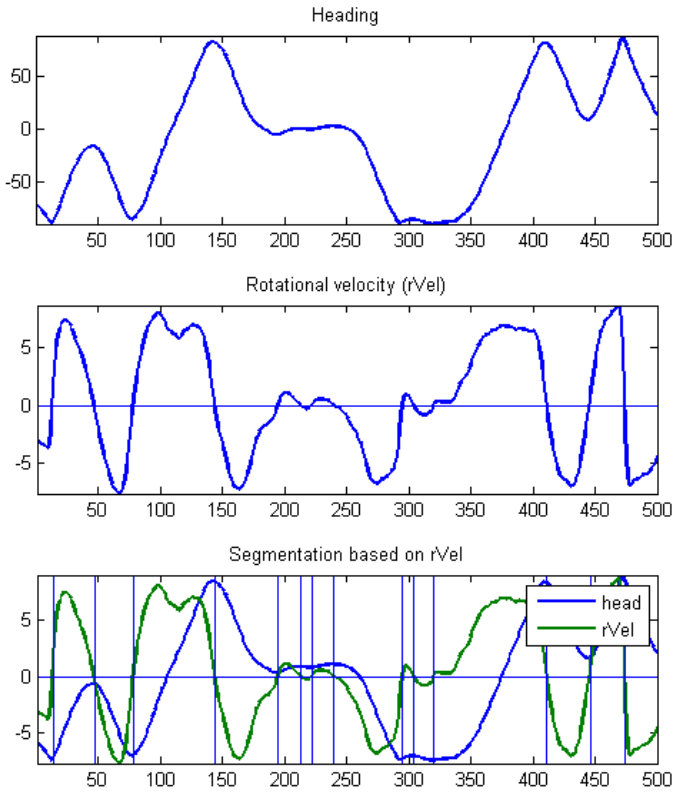


Figure 5.2: Segmentation based on rVel

be applied to all other driving signals. For example we can divide acceleration pedal position, steering wheel angle, and even 2d (X, Z) position coordinate. The result of segmentation on later one is given in Figure 5.4. The red dots in the third plot, show the beginning of each segment. The location of red dots, reveals the moments our driver decided to change his pedal or wheel operations. By looking at the straight lane we can see three red dots in range of about -10 to 30. Maybe one asks why the driver decided to apply different action set in a part of road in which there is nothing important is happening. The answer is even in a straight lane sometimes it happens we rotate the wheel for small degrees to the left or right. Therefore it is detected as a change in rotational velocity.

One important property of the segments is that the sign of tAcc or rVel in each segment is constant and not-changed. We will see how this property will lead us to

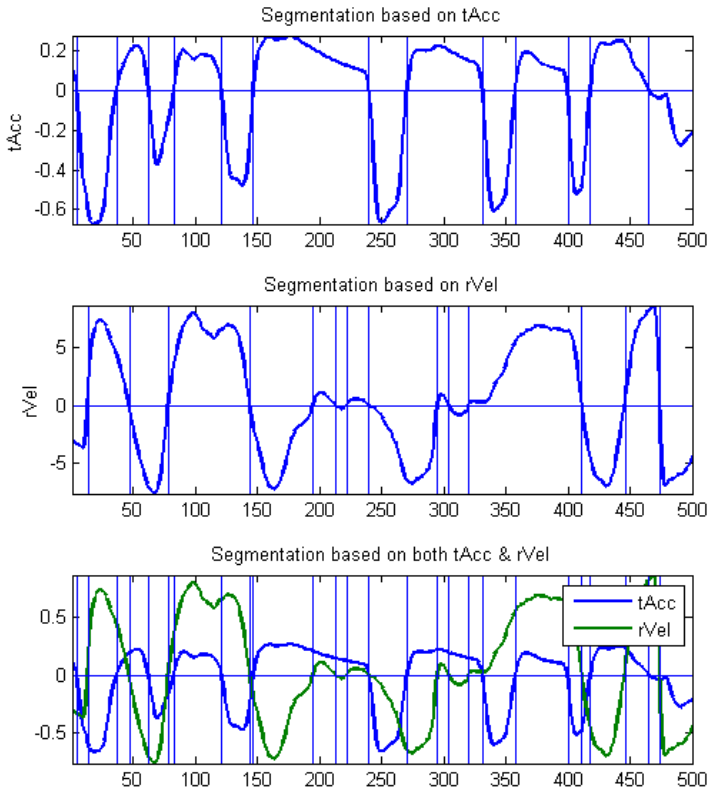


Figure 5.3: Segmentation based on both tAcc-rVel

appropriate partitioning. From Figure 5.3 we can see that it happens a segment in result of one feature is divided into two or more from the result of another feature.

5.2 Partitioning

In previous section it is mentioned that one of the applications of segmentation is grouping. When the segmentation process is done, we have plenty of data segments. We believe there are similarities between them and based on that we can categorize them into a number of classes. From the fact that drivers usually repeat some specific driving habits in various driving situations, we looked for some ideas for grouping similar behaviour data segments. Two questions must be answered here: 1) how many classes do we have, 2) what kind of criterion should be employed to group data

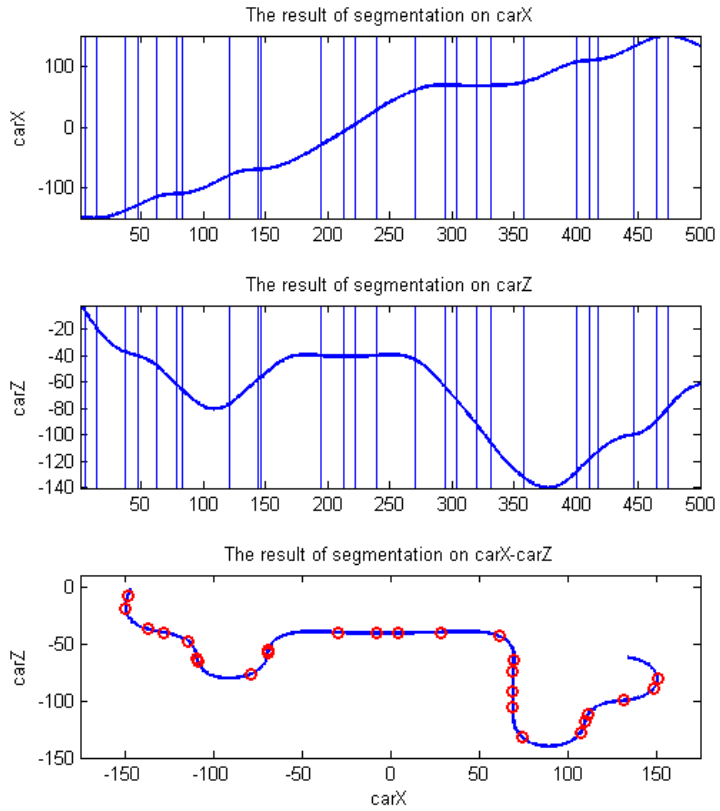


Figure 5.4: The result of segmentation on carX-carZ

segments into classes. As a matter of fact when we proposed the feature extraction algorithm we also thought about the grouping problem. The answers can be taken by studying the 2d feature space. Although we are going to use the result of segmentation for grouping, but we also proposed another approach for grouping without considering segments. In latter case, we define some partitions in the 2d feature space in which data samples are categorized into the partitions not data segments. Considering all of these, three partitioning approaches are examined: threshold-based, GMM-based, and hierarchical (Figure 5.5).

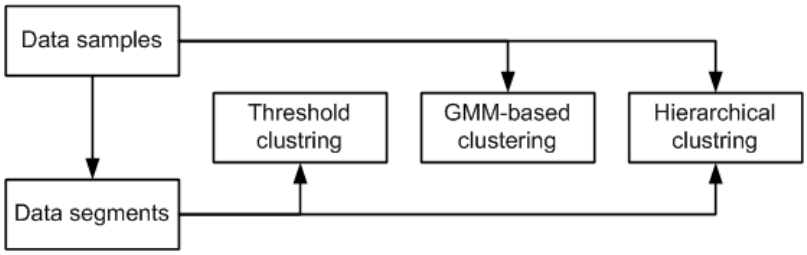


Figure 5.5: Three partitioning approaches: threshold-based, GMM-based, hierarchical.

5.2.1 Threshold partitioning

From the segmentation, we recall that segmentation is done in three steps: 1) finding the local extrema of velocity, 2) finding the local extrema of heading, and 3) taking union between two sets. For segments resulted from step 1, the extracted feature, translational acceleration has either positive or negative values during the whole segment. In the same fashion, for segments resulted from step 2, rotational velocity has either positive or negative values during the segment. Of course the same principal still stands for their union. Here an idea for partitioning comes out: first we can take average amongst all data samples of a data segment, then discretize them into some values. In case of translational acceleration, we can think about having positive acceleration, zero or negative. By defining a threshold we can divide data segments between those three groups. Let's just assign a label to each group as follows: +1, 0, -1 for positive, zero, negative respectively. We can treat rotational velocity in the same way. Having defined three groups for each feature, in total we will have nine different groups in combination (Figure 5.6). Each combined group acts as a class for us. The classes are labelled from 1 to 9 (Figure 5.7).

We may define more than three groups for each class. For example define five regions for tAcc and three regions for rVel, we will have 15 groups in total. It is all up to us and of course depend on the application. For instance when we define five regions for tAcc, that means we have options to define very positive acceleration, normal positive, zero, normal negative, and very negative.

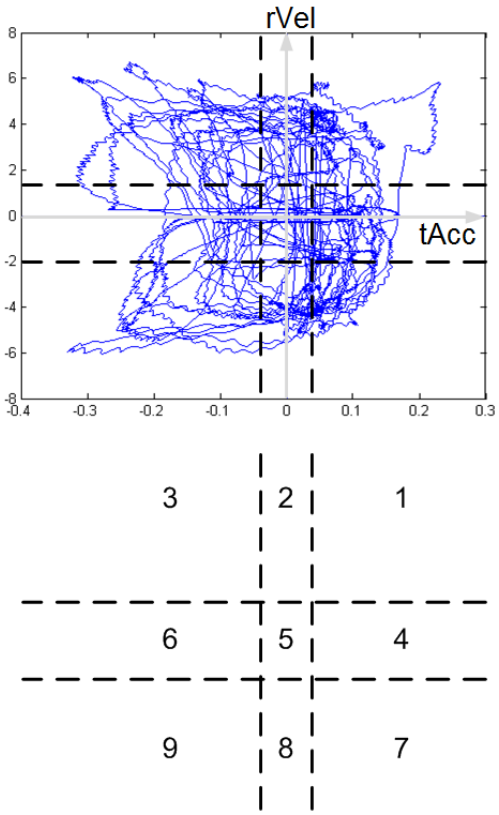


Figure 5.6: Threshold partitioning: partitioning based on discretizing tAcc and rVel into three values, Positive, Zero, Negative. Two thresholds define the range we interpret as Zero. For example in regions 4, 5, 6, tAcc is considered as zero and in 2, 5, 8, rVel is Zero.

		Translational acceleration		
		+1	*0*	*-1*
Rotational velocity	*+1*	1	4	7
	0	2	5	8
	-1	3	6	9

Figure 5.7: Threshold partitioning: labelled classes

5.2.2 GMM-based partitioning

When we talked about segmentation, it was mentioned that it maybe or may not be used for the purpose of grouping. In previous section, we saw how a threshold-based partitioning approach makes use of result of segmentation. In contrast in this section

we are going to review another approach for partitioning data samples instead of data segments.

The idea of GMM-based partitioning comes from the stochastic nature of the problem of interest. In a way it can be considered as an extension to threshold partitioning, instead of defining some threshold values heuristically, we can examine the histogram plots of features individually and see how the data is distributed along each feature axes. Based on their distributions, a Gaussian mixture model (GMM) can be fitted over the data. In the driving application and because the features are extracted from a single 1-dimensional source, a mixture of univariate Gaussian pdfs will be enough for us. By studying histogram plots of tAcc and rVel, we can realize how many Normal pdfs we need to make a mixture.

Figure 5.8 demonstrates a typical example. Rotational velocity is given in y-axes while translational acceleration in x-axes with their corresponding histogram plots. From vertical histogram at least four distinctive Normal distributions can be observed. In the same manner a mixture of two or even three Normal pdfs can be realized for horizontal histogram. That gives us some ideas about the possible sources of each dimension of data. If we agree that rVel comes from a mixture of four Normal sources and tAcc from three sources, in combination we will have 12 classes each one with a distinctive characteristics. In other words we partitioned the 2d feature space into 12 different classes (Figure 5.9).

Now that we know the number of Normal pdfs in each mixture model, we need to optimize their parameters via an appropriate learning process. Usually the method of expectation maximization is used for this purpose. In the end of learning process, we have the optimized Normal probability density functions. From the Figure 5.9 the lower two plots are about learned GMMs. Once the GMMs are learned, we can use them to recognize the belonging source of each data sample. From the 2d feature space, we peak a data sample, it is a 2d scalar-valued vector, we examine its first element against each corresponding Normal pdf, the one with highest probability is the belonging source. In the same way, the belonging source of second element is determined. By their combination we determine the examined data sample belongs to which class. In the case of given example in Figure 5.9, the labelled classes is given in following table:

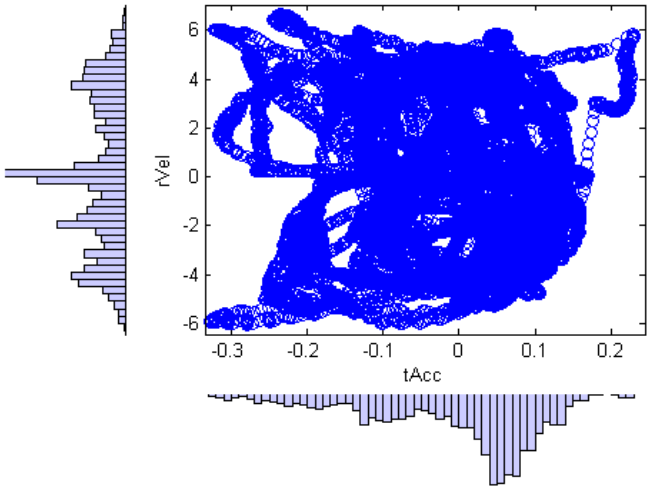


Figure 5.8: Scatter histogram plot.

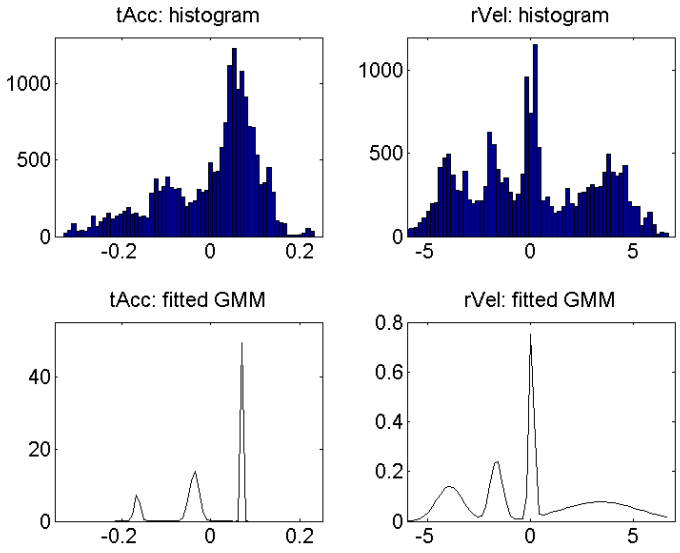


Figure 5.9: GMM-based partitioning: three Normal sources for tAcc and four Normal sources for rVel can be distinguished from each histogram plots.

		Translational acceleration		
		Source 1	*#2*	*#3*
	Source 1	1	5	9
Rotational	*#2*	2	6	10
velocity	*#3*	3	7	11
	#4	4	8	12

Figure 5.10: The labelled classes resulted from GMM-based partitioning.

5.2.3 Hierarchical partitioning

In hierarchical partitioning, we take advantage of both data samples and data segments. Data segments are important for us because they are reflecting the moments that our driver kept his driving decisions unchanged. In each segment, translational acceleration and rotational velocity are either positive or negative. That means in each segment, we are intending to increase or decrease the speed and turn right or left. There is no other case than these four situations. The idea of hierarchical partitioning comes from this concept. Firstly we can partition data segments into these four main classes. Correspond to each of them we can look for more detail sub-classes by studying the distribution of data samples. From top to bottom the result of first step can be called root (first) layer, and other steps, next layers.

An example of hierarchical partitioning is given in Figure 5.11. As we can see, there are two layers, in root layer data segments are divided into four main classes and in second layer gathered data samples are divided again into four other sub-classes. That brings in total 16 classes to our attention. Although it seems there is no big difference compared to other methods but there is huge advantage in terms of efficiency in HMMs training and recognition. As a matter of fact, we train HMMs according to their layer. For example in first layer we train four HMMs, then according to each region in next layer we train another four HMMs. That means in total we will have 20 trained HMMs. This will result in higher correct classification ratio because in each layer we are dealing with just four HMMs at the same time. This will reduce the effect of wrong classification that usually happens if the number of classes are big. Although 20 HMMs are trained in given example, but we need just eight HMMs to find the belonging class.

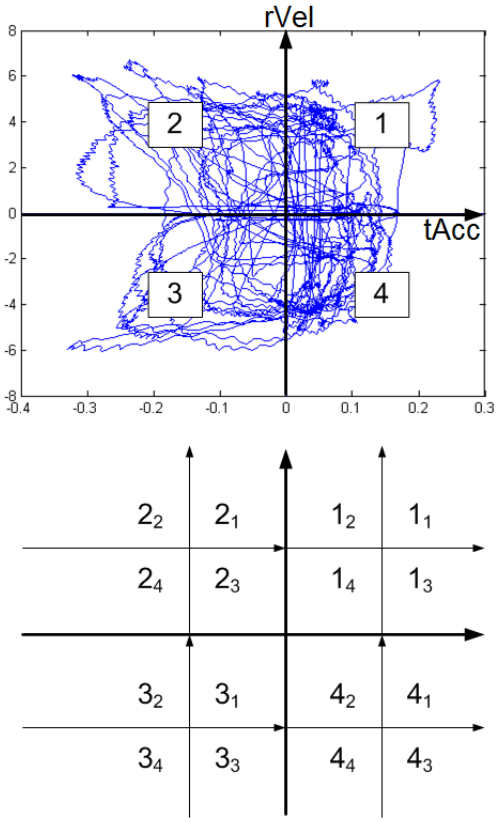


Figure 5.11: Hierarchical partitioning: root layer is divided into four basic classes, and then each one again is divided to another 4 sub-classes in next layer, makes totally 16 classes.

5.3 Conclusion

We need classes for one main reason: we are going to train one HMM per each class. In addition we need classes that have some physical interpretation embedded inside. This latter one is necessary because we want to provide some feedback according to them. From this point of view it is early to decide which partitioning algorithm is best for our purpose. To evaluate them we need to know: 1) is there any physical meaning underlaid beneath classes, 2) how accurate is training and recognition based on classes, 3) how easy or complex is the partitioning method for implementation and how much driving data we need, 4) how does it reflect the relation to the geometry of the road? In other words we want to know is there any relation between classes and specific part of the road such left-hand or right-hand curves? By thoroughly examining the aforemen-

tioned partitioning methods, it is realized that threshold-based partitioning is based on data segments and seems to be easy to implement. Also we can attach some physical unique meaning to each class. GMM-based partitioning is more compliant with the stochastic nature of the data, and it is based on data samples, but it needs a learning session to optimize mixture's parameters. We literally need huge amount of data to best learn the GMMs. In addition partitioning based on GMMs might have some errors like all stochastic methods, in that case it is possible a data sample classified into wrong class. The similar physical meaning to threshold-based partitioning can be extracted from each class. Hierarchical partitioning is based on both data segments and data samples. In first (root) layer, the partitioning is done based on data segments, but in next layer(s) we just rely on data samples and how they are distributed in the selected top-layer class. We need histogram plots for more detail partitioning in layer(s) other than root layer. One can use either GMM-based method or even threshold-based method in bottom layers.

Chapter 6

Driving Behaviour Recognition using HMMs

Having driving data segments/samples partitioned into several classes, distinct Hidden Markov Models can be trained individually on data obtained from each class of a classification problem. If we create a set of those models and specialize each model to recognize each of the separate classes, we can then use the HMMs ability to calculate the likelihood that a given sequence belongs to the model to determine the most likely class of an unknown sequence.

Two basic operations for HMM pattern recognition are training and evaluation. Training is a procedure of calculating the model parameters (namely, state transition and observation symbol probability distribution and initial state distribution) on the basis of a training set of observation sequences in order to maximize $P(O|\lambda)$. Generally, the topology of the model (number of states, number of symbols, and imposed constraints) is known before the training starts. We should also note that the initial state distribution for left-to-right models is fixed and does not need to be re-estimated. Evaluation is a procedure for calculating the probability of a particular observation sequence being generated by the model with given parameters [$P(O|\lambda)$].

6.1 Training

Before start training, we should decide about structure and size of HMMs. Available options are:

- Structure: left-to-right vs. ergodic. There are different HMM architectures, depending on the limitations imposed on the state transition probability matrix A. If we impose constraints: $a_{ij} = 0$ for $j < i$ and $\pi_1 = 1$, we get the so-called left-to-right model (also known as Bakis model). The left-to-right model always starts at the first ("leftmost") state, and transitions are allowed only toward later ("right") states or to the same state. It has been shown that the left-to-right model is better than the general model at capturing dynamic characteristics of data by imposing a temporal order to the model [35]. Left-to-right models are used in a number of temporal pattern recognition applications such us speech recognition [32], [36], human gesture recognition [14], and signature verification [35], with great success. As the driving pattern recognition problem is similar to the above-mentioned domains, we restricted our attention to left-to-right models.

- Type: discrete vs. continuous. This is refer to type of emissions, in other words, how we are going to represent the observations. In discrete case we usually quantize the measured signals into a finite set of symbols. In continuous case we treat observations as continuous signals and fit a mixture of multi-variate Gaussian pdfs for each dimension of data. Clearly our driving signals are continuous, therefore our choice will be naturally continuous HMMs.

- Size: number of hidden states. There is no analytical way to calculate the number of hidden states for a given process. Having some insight into the process that is being modelled can help us make the right decision. In practice usually system designers make a few different attempts at modelling and see how much of a difference it makes. Sometimes, if you start out with too many states, the Baum-Welch algorithm will effectively ignore the unnecessary states (say, by making it highly improbable that other states will transition into them). We found that nine hidden states is optimum choice for our application.

The training begins with concatenating data samples grouped in each class to form training data set. Then we fit a mixture of three univariate Gaussian pdfs over each training data set. This is because we are making use of continuous HMMs. Then the actual training session is started using the method of Baum-Welch. Evaluation and re-estimation steps are repeated until we reach the local probability maxima for $P(O|\lambda)$. In order to find a better global solution, in our experiments, we used 50 iterations of

the BaumWelch algorithm with different initial model parameters. The model with the highest probability was chosen as the training result for each HMM.

It should be noticed that data samples in each class is divided into training data set and testing data set. For this purpose a 2-fold cross validation method is applied to whole data samples within each class.

6.2 Evaluation

The evaluation is done by measuring the performance of trained HMMs is response to testing data set. A rectangular sliding window with the length of 16 and 50% overlapping choose data samples to feed into each HMMs. This is repeated until sliding windows swept the whole range of testing data set. Each HMM outputs a measure of likelihood in response to each window. By aggregating the resulted likelihood measures, the HMMs with highest rank is determined as belonging class.

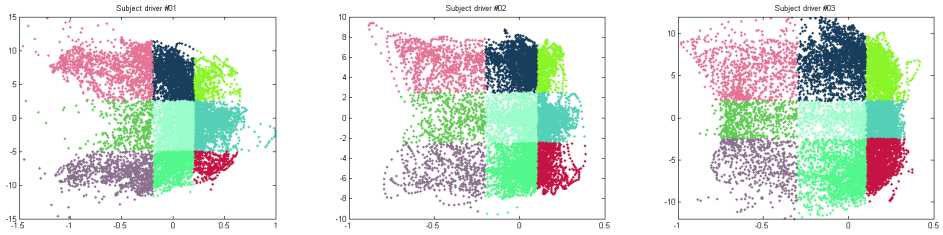
6.3 Results

12 subjects, 10 male and 2 female, are hired to participate in experiments using driving simulator. One of them with 12 year driving experience is chosen as the expert driver. Others ranged from having no driving experience to a few years limited driving experience. All subject's driving data are collected from driving around big road, for at least 25 minutes. In case of expert driver, this process is repeated several time in different days, to collect bigger amount of driving data. In addition the expert driver's performance along with one another driver, is collected during driving small road.

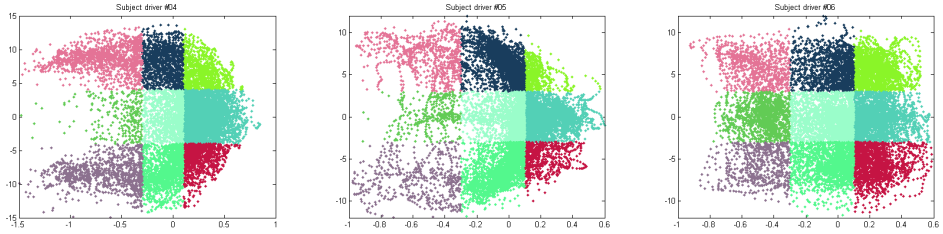
6.3.1 Analysis 1

In this analysis all 12 subject drivers are participated. We decided to apply the threshold-based partitioning method to classify each driver's data into 9 classes. The threshold values for each driver's is determined by studying corresponding plot of 2d feature space, i.e. rVel vs. tAcc (Figure 6.1).

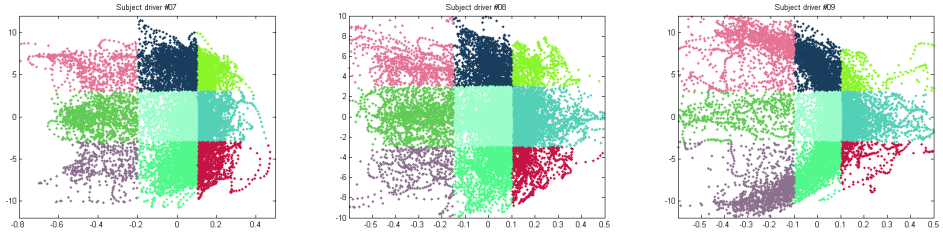
The result of training 9 HMMs for each driver are given in Figure 6.2. For each driver, the best achieved recognition rate during training is given in "Train" row and testing result in "Test" row. Also the average correct classification rates are given in last column. From the given table it can be seen that overally the recognition rates are



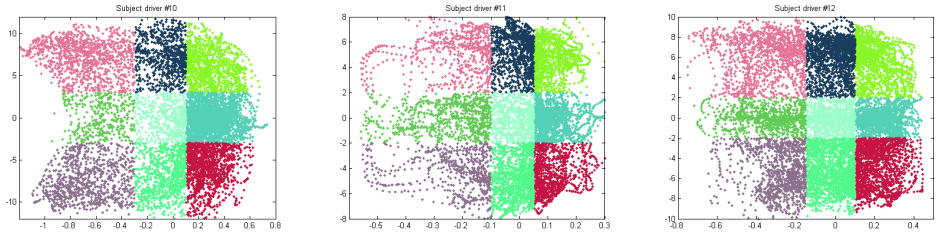
(a) Driver01 (b) Driver02 (c) Driver03



(d) Driver04 (e) Driver05 (f) Driver06



(g) Driver07 (h) Driver08 (i) Driver09



(j) Driver10 (k) Driver11 (l) Driver12

Figure 6.1: Plot of 2d feature vectors for all 12 subject drivers. Driver12 is the expert driver.

very high more than 95% in average. The lowest one belongs to Driver09 and highest one to Driver04. In addition the HMMs trained for Driver12, the expert driver, have shown average correct classification rate about 97

This proves the way we addressed the problem. Considering each class as a driving behaviour, the trained HMMs are being able to recognize them with over 95% accuracy. This is a promising result, because usually in classification problems this kind of high performance cannot be achieved. The main reason behind this is the extracted features are fully adapted for this application.

By matching result of Figure 6.2 and Figure 6.1, it can be seen that the recognition rate for drivers with widely spread plots are lower than compacted ones. There is one physical reason for this: when the subject driver try to push the gas pedal to the maximum, the speed is increased too much, in result he/she needs to release the gas pedal completely to reduce the speed quickly to avoid any collision. This is an immature behaviour and you will not see these kinds of actions from an expert driver.

Driver#	Rate	HMM#1	#2	#3	#4	#5	#6	#7	#8	#9	Avg
01	Train	1.0000	0.9934	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**0.9993**
	Test	1.0000	0.8874	1.0000	0.9857	0.9948	0.8824	1.0000	0.9615	1.0000	**0.9680**
02	Train	1.0000	0.9510	1.0000	0.9504	0.9835	0.9221	1.0000	1.0000	1.0000	**0.9786**
	Test	0.9701	0.9161	1.0000	0.9220	0.9091	0.8312	1.0000	1.0000	1.0000	**0.9498**
03	Train	1.0000	1.0000	1.0000	1.0000	1.0000	0.9851	1.0000	1.0000	1.0000	**0.9983**
	Test	1.0000	0.9536	0.9392	0.9921	0.9111	0.8134	1.0000	1.0000	1.0000	**0.9566**
04	Train	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	**1.0000**
	Test	1.0000	1.0000	0.9714	1.0000	0.8857	1.0000	1.0000	1.0000	1.0000	**0.9841**
05	Train	0.7429	0.8919	0.7955	0.9939	0.9178	1.0000	1.0000	1.0000	1.0000	**0.9269**
	Test	0.6857	0.8829	0.8409	0.9939	0.7877	1.0000	1.0000	1.0000	1.0000	**0.9101**
06	Train	1.0000	1.0000	0.9908	1.0000	0.9945	0.9429	1.0000	1.0000	1.0000	**0.9920**
	Test	0.9920	1.0000	0.9725	1.0000	0.9615	0.9429	1.0000	0.9512	1.0000	**0.9800**
07	Train	0.9759	0.9677	0.9155	0.9708	0.9730	1.0000	1.0000	1.0000	1.0000	**0.9781**
	Test	0.9756	0.9742	0.9014	0.9197	0.8198	0.9388	1.0000	1.0000	1.0000	**0.9477**
08	Train	0.7083	1.0000	0.9623	1.0000	0.8384	1.0000	1.0000	1.0000	1.0000	**0.9454**
	Test	0.4792	0.9896	0.9623	1.0000	0.7475	0.9917	0.9815	1.0000	1.0000	**0.9058**
09	Train	0.9259	0.8389	0.8462	1.0000	0.8223	0.8227	1.0000	1.0000	1.0000	**0.9173**
	Test	0.8519	0.8556	0.8077	0.9860	0.7314	0.7801	0.9770	1.0000	0.9610	**0.8834**
10	Train	0.9747	1.0000	0.9643	0.9800	1.0000	0.9643	1.0000	1.0000	1.0000	**0.9870**
	Test	0.9744	0.9517	1.0000	0.9600	1.0000	0.9286	1.0000	1.0000	1.0000	**0.9794**
11	Train	0.5976	0.8915	0.9710	1.0000	0.9672	1.0000	1.0000	1.0000	0.9722	**0.9333**
	Test	0.5976	0.9147	0.8406	0.8507	0.9672	0.8804	1.0000	1.0000	0.9167	**0.8853**
12	Train	1.0000	1.0000	1.0000	0.9898	1.0000	0.9920	1.0000	1.0000	0.9412	**0.9914**
	Test	0.8989	0.9855	0.9820	0.9592	1.0000	0.9680	1.0000	1.0000	0.9706	**0.9738**

Figure 6.2: The result of training 9 HMMs for each driver.

6.3.2 Analysis 2

In this analysis we investigated what will happen if we use the results of partitioning based on extracted features as a criterion to categorize other pairs of driving signals, specificity [pedal, wheel] and [vel, head], into determined classes. To this end, first

according to threshold-based partitioning, the feature space is divided into six classes. Then it is used to partition driving signals in [vel, head] space and [pedal, wheel] space. Each time six HMMs are trained based on the partitioned data, and then tested against them. In this analysis only the Driver12's data in used.

The threshold-based partitioning is done by taking a quick look at scatter histogram plot of feature space. It gives us some insight about how to choose the threshold values. From the Figure 6.3 we can clearly observe that the y-axis (rVel) can be divided into three horizontal regions while it is not so clear about the x-axis (tAcc). But we made a decision to simply consider two vertical regions, either positive or negative. So in total we are going to have six classes (Figure 6.4).

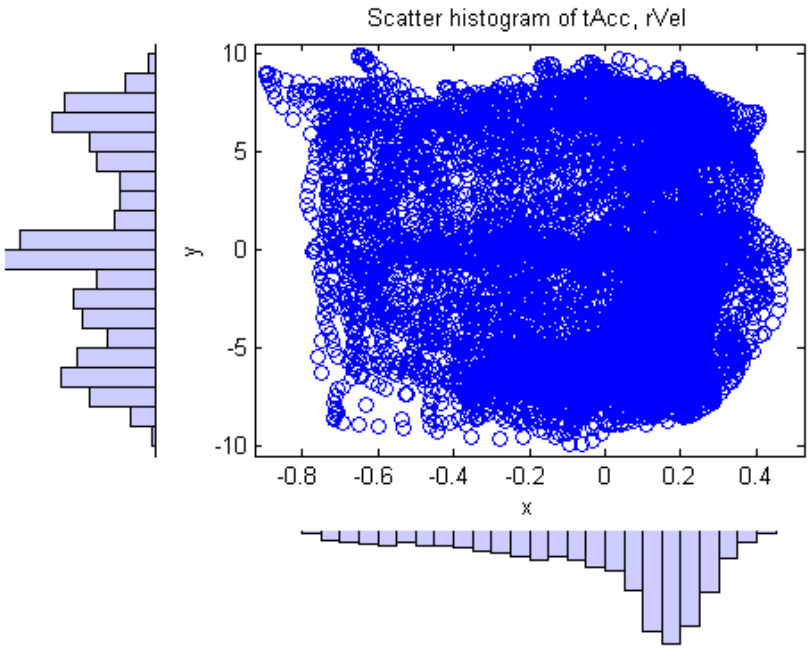


Figure 6.3: Scatter histogram of rVel vs. tAcc.

The result of training and testing is given in Figure 6.5. As it can be seen, a very high classification percentile, about 95% is achieved. This proves the capability of HMMs to study time-series data. The training is repeated only 10 times, and from the given figure, it was not actually necessary. We could have stopped earlier for

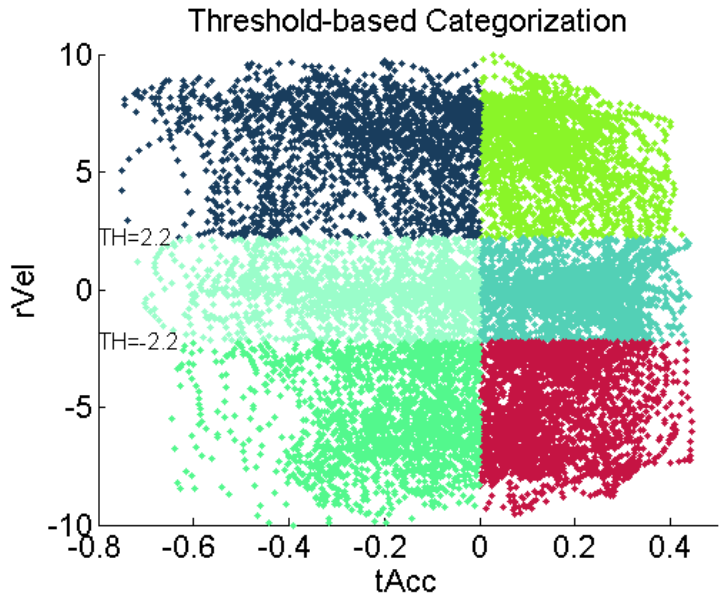


Figure 6.4: rVel vs. tAcc, classes labelled by colors.

example in trial three or four. Amongst six HMMs only the third one showed a lower performance. In that class, the tAcc in positive while rVel is smaller than a negative threshold, the green color region in Figure 6.4.

Training						
Trial #	HMM #1	#2	#3	#4	#5	#6
1	0.9920	0.9921	0.5352	0.9535	1.0000	0.9140
2	1.0000	0.9921	0.5070	0.9651	1.0000	0.9032
3	0.9440	0.9921	0.8099	1.0000	1.0000	1.0000
4	0.9520	0.9843	0.7324	0.9767	1.0000	0.9570
5	0.8880	1.0000	0.7958	0.9302	1.0000	0.7634
6	0.9760	0.9921	0.7958	0.9302	1.0000	1.0000
7	0.9680	0.9843	0.7887	1.0000	1.0000	0.8710
8	0.8560	0.9843	0.8028	0.5465	1.0000	0.8817
9	0.9600	0.9843	0.7887	0.9884	1.0000	0.9785
10	0.9520	1.0000	0.5704	0.9884	1.0000	1.0000
Best	1.0000	1.0000	0.8099	1.0000	1.0000	1.0000
	Expected best average classification percentile					**0.9683**
Testing						
	1.0000	1.0000	0.7042	1.0000	1.0000	1.0000
	Average correct classification percentile					**0.9507**

Figure 6.5: Training and testing results based on tAcc and rVel.

We wanted to evaluate the performance of training HMMs using some variable other than extracted features to see how careful our features are. For this purpose we trained the HMMs once using pair of accelerator pedal position and steering wheel angle and later using velocity and heading. It should be noticed that the partitioning criteria is still based on tAcc and rVel. The results of partitioning which are labelled by colors are given in Figure 6.6 and Figure 6.7 respectively. As we can see there is no specific connected region can be distinguished, it more seems data samples are randomly selected in the relevant 2d space.

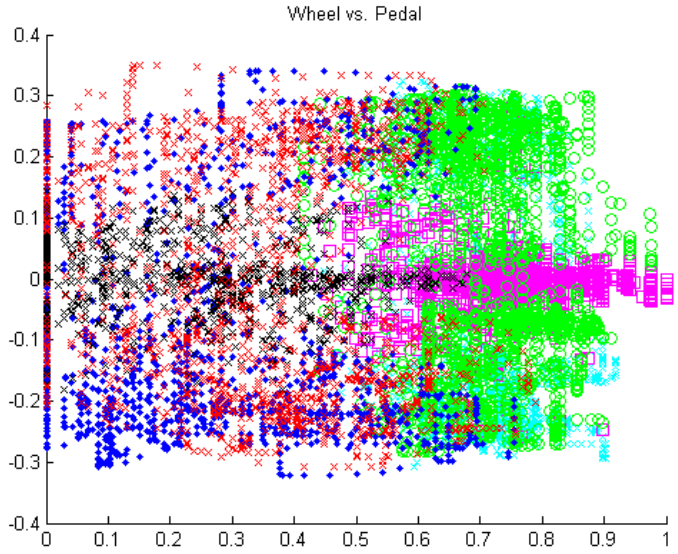


Figure 6.6: Steering wheel vs. Accelerator pedal, classes labelled by colors.

The training and testing results are given in Figure 6.8 and Figure 6.9. In case of [pedal, wheel] the average percentile is about 73% and in case of [vel, head], even worse about 64%. This proves that we made a right decision to choose the extracted features for the purpose of training and testing. One can claim if the partitioning criterion is modified to meet the new variables, we might have better classification performance. The answer is positive, maybe we could have better classification but instead we lose the beauty of having physical interpretation in our classes. It is worth to emphasise again that we are not looking just to identify and recognize some driving behaviours, it is important to us the behaviours have some physical meaning attached.

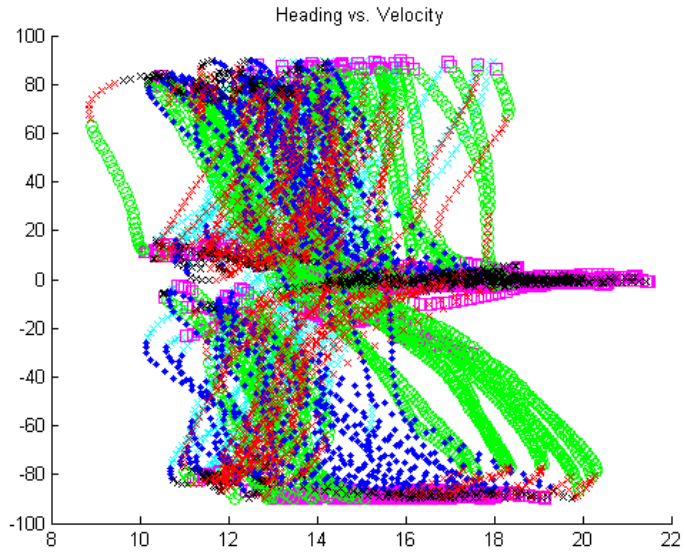


Figure 6.7: Heading vs. Velocity, classes labelled by colors.

Training						
Trial #	HMM #1	#2	#3	#4	#5	#6
1	0.7040	1.0000	0.5915	0.6512	1.0000	0.5054
2	0.6480	0.9764	0.7465	0.6628	0.9808	0.5269
3	0.6960	1.0000	0.5423	0.6977	1.0000	0.5054
4	0.7040	1.0000	0.4437	0.6628	1.0000	0.4731
5	0.6880	1.0000	0.4648	0.6744	1.0000	0.6022
6	0.6960	0.9764	0.4930	0.6628	0.8846	0.5376
7	0.6880	1.0000	0.5282	0.6860	1.0000	0.5161
8	0.6480	1.0000	0.5986	0.8023	0.9808	0.4409
9	0.7040	1.0000	0.4577	0.6860	0.9808	0.5376
10	0.7120	0.9921	0.4437	0.6512	0.9808	0.4839
Best	0.7120	1.0000	0.7465	0.8023	1.0000	0.6022
	Expected best average classification percentile					**0.8105**
Testing						
	0.6800	0.6850	0.7606	0.7209	1.0000	0.5269
	Average correct classification percentile					**0.7289**

Figure 6.8: Training and testing results based on accelerator pedal and steering wheel.

In summary, one immediate conclusion of this analysis is the efficacy of extracted features to recognize different driving behaviours. Its accuracy is almost about 95% by considering that it is affected by just one of six HMMs, specificity HMM 3. The

Training						
Trial #	HMM #1	#2	#3	#4	#5	#6
1	0.2240	0.6063	0.5493	0.6279	0.4423	0.4301
2	0.3280	0.5591	0.8169	0.4419	0.7500	0.2903
3	0.2720	0.5984	0.3873	0.8140	0.3462	0.5161
4	0.3520	0.6142	0.9577	0.6395	0.4808	0.1935
5	0.8800	0.7323	0.4014	0.5000	0.3654	0.5914
6	0.4640	0.5748	0.7817	0.5581	0.2692	0.3763
7	0.4560	0.7874	0.5704	0.4884	0.4231	0.3656
8	0.2960	0.4646	0.6831	0.4884	0.5769	0.4516
9	0.3920	0.4567	0.6549	0.2907	0.4808	0.4086
10	0.1680	0.7008	0.3521	0.5233	0.4615	0.6237
Best	0.8800	0.7874	0.9577	0.8140	0.7500	0.6237
	Expected best average classification percentile					**0.8021**
Testing						
	0.4160	0.7953	0.7465	0.6628	0.7692	0.4409
	Average correct classification percentile					**0.6385**

Figure 6.9: Training and testing results based on velocity and heading.

performance of HMM 3 in recognition of [tAcc, rVel] vectors are not so high while the rest of HMMs are pretty surprising. Another result is that, [vel, head] (denoted by vh), are not good choice for training HMMs while the partitioning is done based on ”tr”. They all almost show low average classification rate. But ”pw”, [pedal, wheel], are in between. Their performance is not as good as ”tr” but not as bad as ”vh”.

6.3.3 Analysis 3

We examined the partitioning, training and recognition all based on velocity and heading as chosen features. We actually wanted to show that it might be possible to get high recognition rate in case of using other variables than extracted features but as mentioned before we lose having physical meaning in classes.

The evaluation is done based on Driver12's data. In Figure 6.10 the scatter plot of vel-head is given. We divided the vel-head space into nine classes according to threshold-based partitioning (Figure 6.11). The training and testing result is given in Figure 6.12. As it can be seen again we can get a high recognition rate about 95%. This proves the efficacy of partitioning criterion, i.e. partitioning feature space into a number of classes. Although theses classes can be considered as driving behaviours, but they do not have useful information to be used for providing feedback. In addition

they are completely dependant to absolute values while in [tAcc, rVel] space, we detect the rates (changes over time) not absolute values.

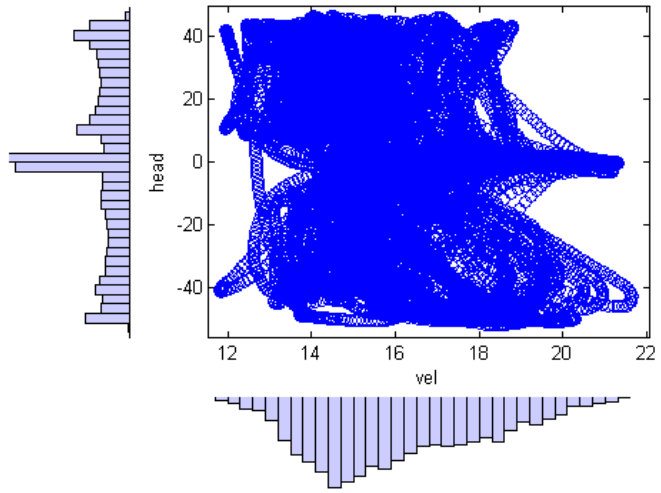


Figure 6.10: Scatter histogram plot of head vs. vel.

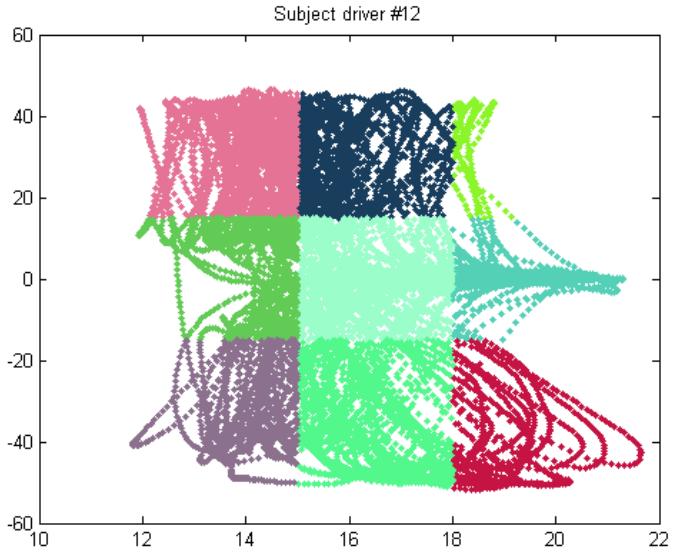


Figure 6.11: The result of partitioning the vel-head space into nine classes.

Training									
Trial #	HMM #1	#2	#3	#4	#5	#6	#7	#8	#9
1	1.0000	1.0000	0.5714	1.0000	0.9636	1.0000	0.9730	1.0000	1.0000
2	0.7778	1.0000	0.9388	0.9574	1.0000	0.9739	1.0000	0.9780	0.9531
3	0.5556	0.9794	0.8571	1.0000	0.9939	0.9739	1.0000	0.9121	1.0000
4	0.6667	0.9485	0.6327	0.9149	0.9879	0.9130	0.9730	0.8022	1.0000
5	0.1111	1.0000	0.5306	0.8404	0.9758	1.0000	1.0000	1.0000	0.9844
6	0.4444	0.9485	0.9796	0.9574	0.9879	1.0000	0.9797	0.9670	0.9844
7	0.2222	0.9794	0.5714	1.0000	0.9879	1.0000	0.9730	1.0000	0.9219
8	0.7778	0.9897	0.6531	1.0000	0.9818	1.0000	0.9797	0.8571	0.9844
9	0.4444	1.0000	1.0000	1.0000	0.9818	1.0000	0.9797	0.9890	1.0000
10	0.1111	1.0000	0.9796	0.9681	1.0000	0.9826	0.9932	0.7912	1.0000
Best	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	Expected best average correct classification rate								**1.0000**
Testing									
	1.0000	1.0000	1.0000	0.8723	1.0000	1.0000	1.0000	0.9890	1.0000
	Average correct classification rate								**0.9846**

Figure 6.12: The result of training and recognition based on vel-head.

6.3.4 Analysis 4

This time we hired Driver11 and Driver12 to drive both small and big roads. We wanted to see if we can train HMMs using expert driver's data and test against other driver's data, even worse, test against driving data collected from other road. For this end, first we collect expert driver's data on big road. Then according to a threshold-based partitioning, the feature space is divided into six classes. Then we collect his data on small road. Later and for the purpose of comparison, we collect Driver11's data on both small and big road. Same as analysis 2, we also provide the result of partitioning in feature space, but training and testing in two other spaces, pw: [pedal, wheel] and vh: [vel, head].

The comparative table of recognition results is given in Figure 6.13. The first table just summarised the results of training and testing given before. It worth to recall again, all HMMs are trained according to driving data collected during a driving of Driver12 (our expert driver) on big road. His data are collected during different session on different days. Each session includes about 10-12 trials. In total 78 effective trials are collected from the performance of Driver12. Second table shows the recognition result according to testing data of same driver in small road. This time the correct classification rate is about 86%. It is not bad, although it is not as good as the big road, 95%, but it gives an impression of possibility for generalizing the approach. In

third table, the data of Driver11 on big road used for evaluation. The rate is about 87%, that means the HMMs which are trained by another driver, can recognize other driver's behaviours with 87% accuracy. In fourth table, the data of Driver11 on small road used for testing. And in last table, the whole data of Driver12 (78 trials) are concatenate together to be used as testing data set. It can be surprisingly seen that not only the trained HMMs are capable of recognizing the same driver's behaviours on different roads but also they can be used to recognize driving behaviours of another driver in different roads too. This can give us an idea of how to compare an expert driver's performance with a novice driver's. That kind of information can be used in providing some corrective feedbacks.

Driver 1, big road							
	HMM #1	#2	#3	#4	#5	#6	Avg (%)
tr	1.0000	1.0000	0.7042	1.0000	1.0000	1.0000	**0.9507**
pw	0.6800	0.6850	0.7606	0.7209	1.0000	0.5269	**0.7289**
vh	0.4160	0.7953	0.7465	0.6628	0.7692	0.4409	**0.6385**
Driver 1, small road							
tr	1.0000	1.0000	0.2766	0.9250	1.0000	1.0000	**0.8669**
pw	0.1746	0.8904	0.8085	0.3000	1.0000	0.8605	**0.6723**
vh	0.2540	0.4795	0.5319	0.3750	0.2000	0.4186	**0.3765**
Driver 2, big road							
tr	1.0000	0.9735	0.6154	0.7432	0.9459	1.0000	**0.8797**
pw	0.4083	0.5033	0.1154	0.5541	1.0000	0.6220	**0.5339**
vh	0.2667	0.0662	0.6308	0.2568	0.0405	0.1220	**0.2305**
Driver 2, small road							
tr	0.9909	0.9130	0.5410	0.7714	0.8600	0.9596	**0.8393**
pw	0.5000	0.4022	0.3934	0.2429	0.9000	1.0000	**0.5731**
vh	0.3273	0.0435	0.5738	0.2429	0.1000	0.2525	**0.2567**
Driver 1, big road, whole 78 trials							
tr	0.9947	1.0000	0.6727	0.9458	0.9950	0.9984	**0.9344**
pw	0.6270	0.7328	0.6949	0.6579	0.9975	0.4334	**0.6906**
vh	0.4153	0.6450	0.6264	0.5248	0.4296	0.2841	**0.4875**

Figure 6.13: Comparative table of recognition results. tr: tAcc, rVel; pw: pedal, wheel; vh: velocity, heading.

6.3.5 Analysis 5

In this section the result of GMM-based partitioning is given. In GMM-based partitioning the the probabilistic nature of the data are taken into account. For the given example and from the scatter histogram plot, Figure 6.14, we can realize a mixture of four Normal pdfs for rVel and a mixture of three Normal pdfs for tAcc (Figure 6.15). Having a narrow component in GMM makes the classified region to be too small. In addition when the components are too separated and poorly overlapped, it might result in classification error. As we expected, this error can be seen in Figure 6.16 as disconnected regions: the green region is divided by both a narrow vertical dark green and a pink horizontal regions. It can be solved by tweaking the variance of correspond mixture component. We did this and the new result is given in Figure 6.17 in which 12 connected regions, labelled by different colors, can be easily recognized.

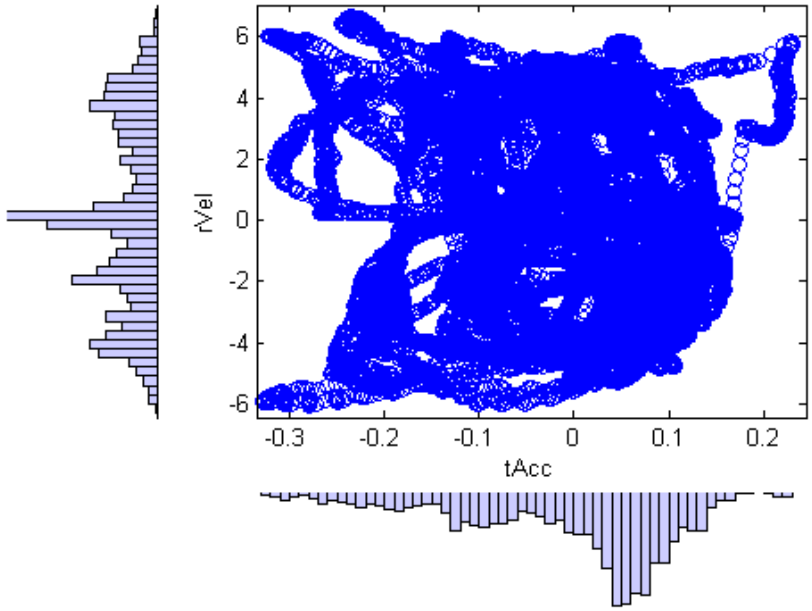


Figure 6.14: Scatter histogram of rVel and tAcc.

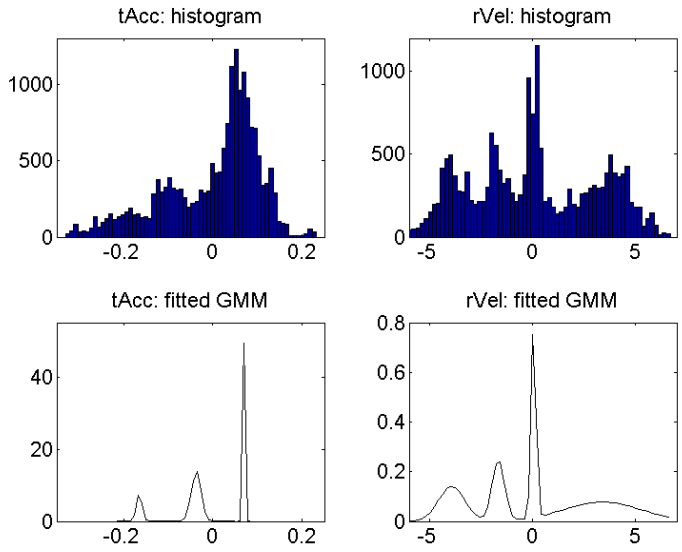


Figure 6.15: Optimized GMMs correspond to each feature, tAcc and rVel.

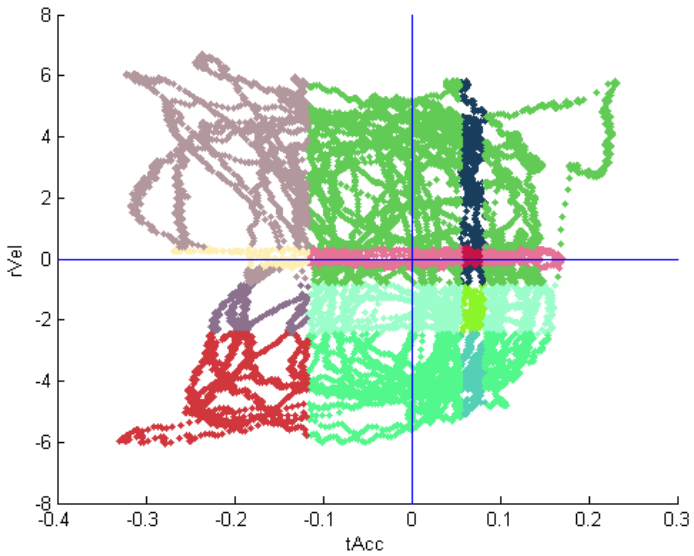


Figure 6.16: The result of GMM-based partitioning. As we expected, some of data are wrongly classified. For example green zone is either vertically and horizontally divided.

Now we have 12 classes, again we train one HMM for each class. The result of training and testing based on Driver12's driving data is given in Figure 6.18. For most

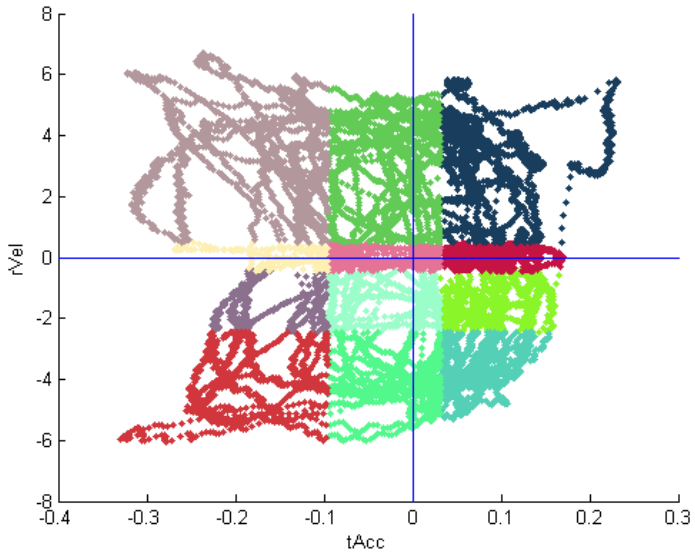


Figure 6.17: The result of GMM-based partitioning. Some of variances are tweaked a bit, so there is no wrong classification.

of classes we have a high accuracy but in few HMMs we do not. For example HMM 5 and 6 showed minimum performance. But in total, its average correct classification percentile is about 71%. The main reason for this low classification rate is lack of enough training data set. It is clearly lower than threshold-based partitioning but as we will see in next analysis, it can be used in hierarchical structure with small number of classes. In some regions, i.e. yellow, pink, and red, the number of partitioned data samples is significantly less than others. This lack of enough training data set, affect the training performance.

6.3.6 Analysis 6

In this analysis we are going to evaluate the the outcome of hierarchical partitioning. When hierarchical partitioning is used for dividing the two dimensional feature space, in first layer (root layer) we have only four classes (Figure 6.19). It can be considered as a special case of threshold-based partitioning with x-axis and y-axis as the thresholds. According to each class one HMM is trained. From the given results in analysis 1, we expect a high classification percentile. We were right about this because according to Figure 6.20 in average we got 95% correct classification rate. In next layer, we

Training												
Trial #	HMM #1	#2	#3	#4	#5	#6	#7	#8	#9	#10	#11	#12
1	0.6714	0.8198	0.9653	0.5449	0.4898	0.5109	0.8621	0.3511	0.6429	0.5000	0.6800	0.7823
2	0.6571	0.9640	0.5896	0.6292	0.4082	0.2065	0.7931	0.4504	0.6786	1.0000	0.8400	0.6855
3	0.6857	0.9459	0.5607	0.5955	0.4898	0.1739	0.7931	0.3969	0.7500	0.6500	0.8000	0.6694
4	0.6786	0.8288	0.5549	0.5506	0.3469	0.5543	0.7931	0.4733	0.6786	0.4667	0.8400	0.6935
5	0.6571	0.9640	0.9653	0.5225	0.4898	0.5326	0.8621	0.4504	0.7143	0.5500	0.8000	0.7097
6	0.6786	0.9099	0.5838	0.5506	0.2857	0.2500	0.8966	0.9084	0.7143	0.7167	0.5600	0.6371
7	0.6429	0.7658	0.4624	0.6517	0.5102	0.5435	0.7931	0.3969	0.5357	0.5500	0.8000	0.7984
8	0.6714	0.9099	0.6474	0.6517	0.3878	0.4130	0.8276	0.3435	0.7857	0.6667	0.7600	0.7823
9	0.6429	0.9550	0.7919	0.6348	0.3673	0.1522	0.8966	0.4733	0.6786	0.8667	0.7600	0.7016
10	0.6929	0.9189	0.9422	0.5449	0.5102	0.3587	0.7931	0.3435	0.5714	0.6500	1.0000	0.7661
Best	0.6929	0.9640	0.9653	0.6517	0.5102	0.5543	0.8966	0.9084	0.7857	1.0000	1.0000	0.7984
	Expected best average correct classification rate											**0.8106**
Testing												
	0.6857	0.7207	0.8092	0.3933	0.3673	0.6739	0.8966	0.8626	0.8929	0.6000	1.0000	0.6452
	Average correct classification rate											**0.7123**

Figure 6.18: The table of recognition result according to GMM-based partitioning.

can treat the selected class as a new 2d feature space and repeat the same procedure we followed in root layer. It might be helpful to look at their histogram plot to see how the data is distributed over each main class (Figure 6.21). Either threshold-based partitioning method, or GMM-based, or even another hierarchical method can be hired again to proceed. As this case is going to be similar to previously given examples, we just proceed one more layer for the zone (a) in Figure 6.21. From the histogram, we decided to divide that space horizontally and vertically into half once more to have four smaller classes. The corresponding partitioning result is given in Figure 6.22 and the training and testing result in Figure 6.23. As it can be seen we could achieve 100% accuracy in recognizing correct classes in second layer. Recalling the accuracy of root layer is 0.9688 and second layer is 1.0000, the combined accuracy will be remain 0.9688.

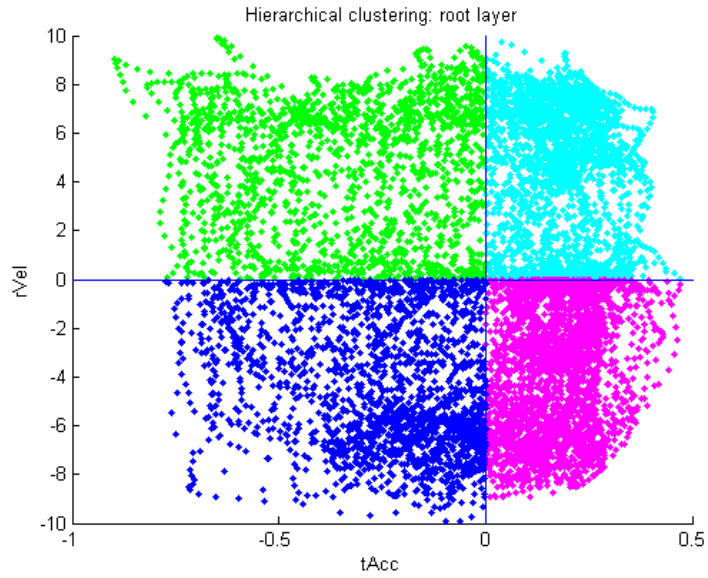
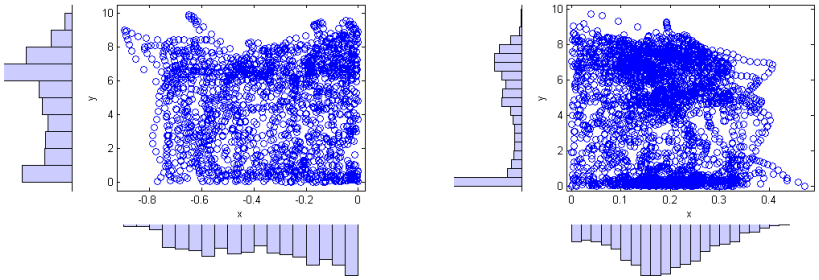


Figure 6.19: The result of hierarchical partitioning in root layer.

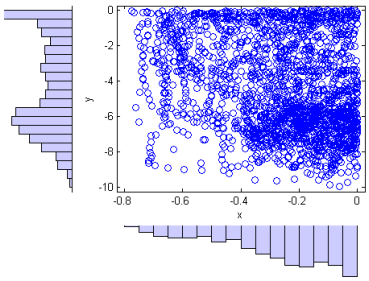
Training				
Trial #	HMM #1	#2	#3	#4
1	0.5054	0.5877	1.0000	1.0000
2	0.9837	0.7109	0.7091	1.0000
3	0.8478	0.5924	0.8636	1.0000
4	0.7772	0.9905	0.9909	0.9672
5	0.6359	0.8531	0.9909	1.0000
6	0.7337	0.3839	0.9455	1.0000
7	0.4022	0.4408	1.0000	1.0000
8	0.7065	0.5877	0.9455	1.0000
9	0.9076	0.6588	0.9909	1.0000
10	0.4293	0.7346	1.0000	0.9508
Best	0.9837	0.9905	1.0000	1.0000
	Expected best average (%)			**0.9936**
Testing				
	0.9130	0.9621	1.0000	1.0000
			Average (%)	**0.9688**

Figure 6.20: The table of recognition result according to hierarchical partitioning in root layer.

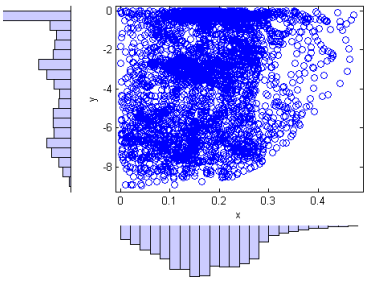


(a)

(b)



(c)



(d)

Figure 6.21: Scatter histogram plot of data in zone 1, 2, 3, 4 correspond to root layer.

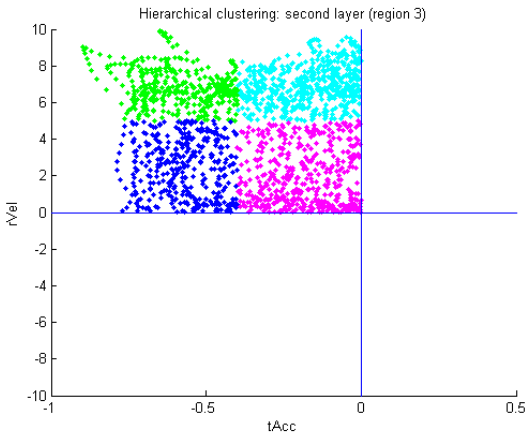


Figure 6.22: Hierarchical partitioning: second layer (region 3) is divided into 4 smaller sub-classes.

Training				
Trial #	HMM #1	#2	#3	#4
1	1.0000	0.7500	1.0000	1.0000
2	1.0000	0.9643	1.0000	0.9091
3	1.0000	1.0000	0.9524	0.6818
4	1.0000	0.9643	1.0000	1.0000
5	1.0000	0.7857	0.9524	1.0000
6	1.0000	1.0000	1.0000	1.0000
7	1.0000	0.8571	1.0000	1.0000
8	1.0000	0.9286	0.9524	1.0000
9	1.0000	0.7857	1.0000	1.0000
10	1.0000	1.0000	0.9524	1.0000
Best	1.0000	1.0000	1.0000	1.0000
	Expected best average			**1.0000**
Testing				
	1.0000	1.0000	1.0000	1.0000
			Average	**1.0000**

Figure 6.23: Training and testing result for the smaller sub-classes in second layer of the hierarchical partitioning.

Chapter 7

Feedback Hints

As it is mentioned previously our journey is started to look for some answers for a main problem: how we can capture and transfer driving skills from an expert driver to a novice trainee. Until now we introduced some methods and algorithms to segment and partition driving data samples and recognize driving behaviours. In this chapter we introduce two approaches to make use of results of driving behaviour recognition to provide some feedbacks: 1) road-dependent, and 2) road-independent. In former one and according to the result of segmentation and partitioning on an expert driver's driving features, we know which element of road belongs to which class. By element, it means the smallest entity of the road which can be 2d position vector. This is why we call it road-dependent because it relies on geometry information from the road. We might call the outcome as reference trajectory. By this way we are being able to compare the novice driver's performance in each road element with the reference trajectory, if they belong to the same class, there is no need for any feedback, if not some feedbacks can be generated according to the nature of classes. We had this view from the very beginning of the project which it was the reason we looked for classes having physical meaning embedded. In road-independent approach we remove the dependency to road position information by extracting two other features from road geometry, namely dov (depth of view) and coh (change of heading). We believe that both feature sets, [tAcc, rVel] from driving signals, and [dov, coh] from road geometry information, referring to a unique process. So if we train separate HMMs based on each feature set, they must be show similar recognition results. The second feature set can be extracted for each road in the same fashion we did for first feature set. That means there is no need to rely on a specific road (position) information in advance and

it can be measured appropriately. We train two sets of HMMs based on feature sets according to the expert driver's driving data. Then during novice driver's performance the recognition is done in both two feature spaces, if the belonging class resulted from each set of HMMs is the same, there is no need for feedback, if not a feedback can be provided appropriately.

7.1 Road-dependent Method

To evaluate the capability of road-dependent method, we assume an example of having four classes. Our expert driver, drove the simulated road for several trials and best trajectory is constructed by the method of aggregation on belonging classes of each road element (Figure 7.1). In Figures 7.2 and 7.3 the result of best trajectory on controlling variables, velocity and heading, is given. Driver09 is selected as our novice driver. Her performance in terms of average correct classification rate was the lowest amongst other participant. The comparative results are given in Figure 7.4. In the third plot of given figure, yellow dots show the similarities while gray dots show the differences. The total number of data samples in best trajectory is 878. The novice driver in only 331 points performed similarly to the expert driver while in 547 did differently. Because we divided the feature space into 4 classes, according to each class from expert driver, there are three wrong classes for novice driver. So in total all difference can be summarized into 12 categories (Figure 7.5).

7.2 Road-independent Method

For the purpose of driving training system, the road-dependent method is fits well. Because in the system we have a few simulated driving roads with various difficulties, and according to an expert driver's behaviours, we already what the best trajectories are in all pre-defined roads. So there is no need to explore and identify the environment. But let's draw our attention to this situation: we intend to extract driving behaviours of an expert driver given a general road, then extend them to every other roads. In another

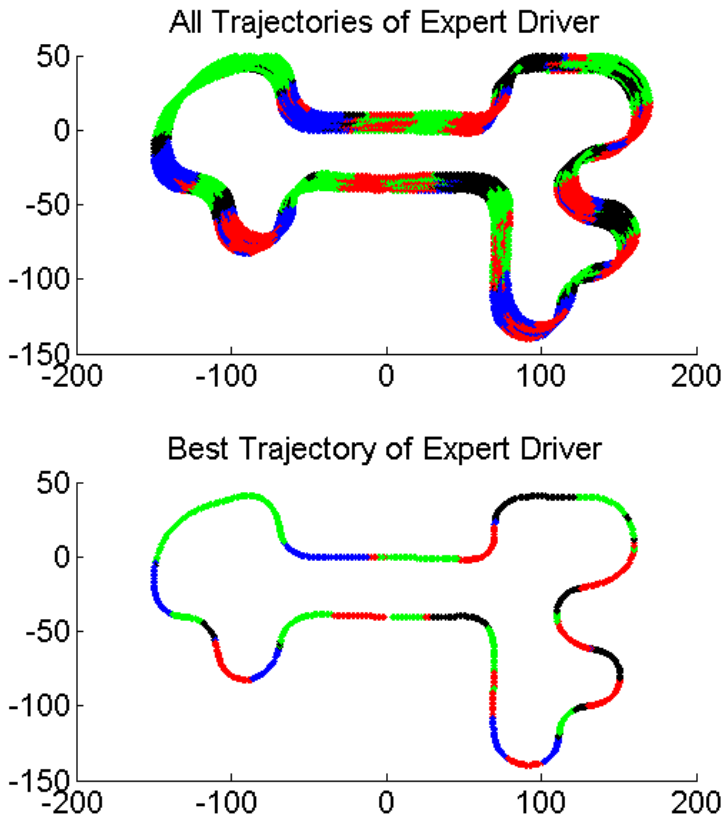


Figure 7.1: The plot of all and best trajectories of the expert driver.

sense, we do not want to collect driving behaviours for every single road, instead we want to generalize them from a somehow complete road to others.

In a real driving task, the driver observes several environmental variables and makes appropriate decisions according to them. We suggest that the effect of two variables are higher than others: depth of view (dov), change of heading (coh). dov is the distance we can see in front of the vehicle within our field of view. cov is an estimation of how we should turn left or right. A plot representing dov and coh is given in Figure 7.6.

As a matter of fact we are looking for feature from the environment or specificity from the road, which are consistent with our extracted features from the controlling variables. dov is a measure of distance and coh is a measure of angle. We suggest they are good choice against tAcc and rVel. Our idea to make use of them is given in the

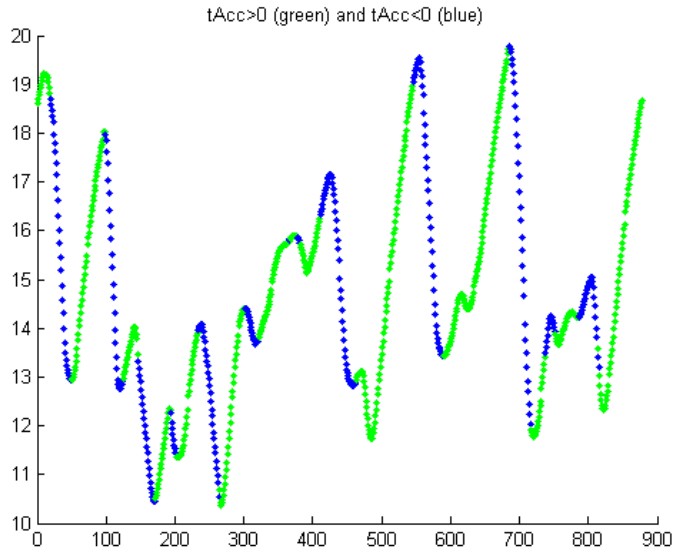


Figure 7.2: The plot of velocity according to best trajectory. Green dots show increasing velocity situation while blue ones decreasing.

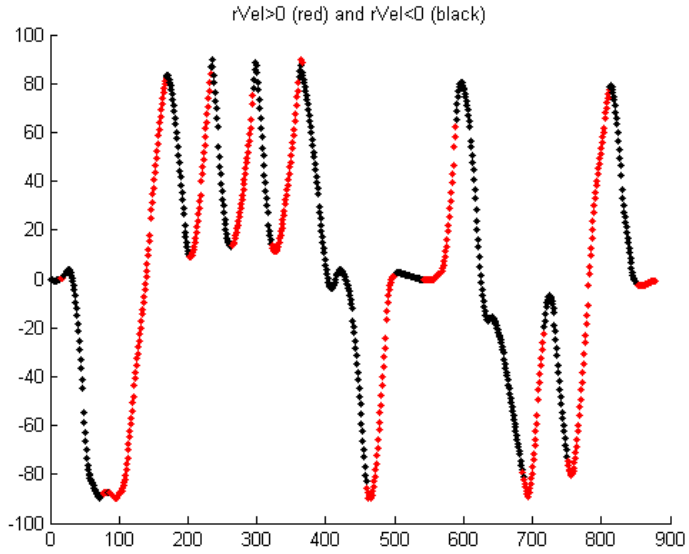


Figure 7.3: The plot of heading according to best trajectory. Black dots show turning to right situation while red ones to left.

following block-diagram (Figure 7.7).

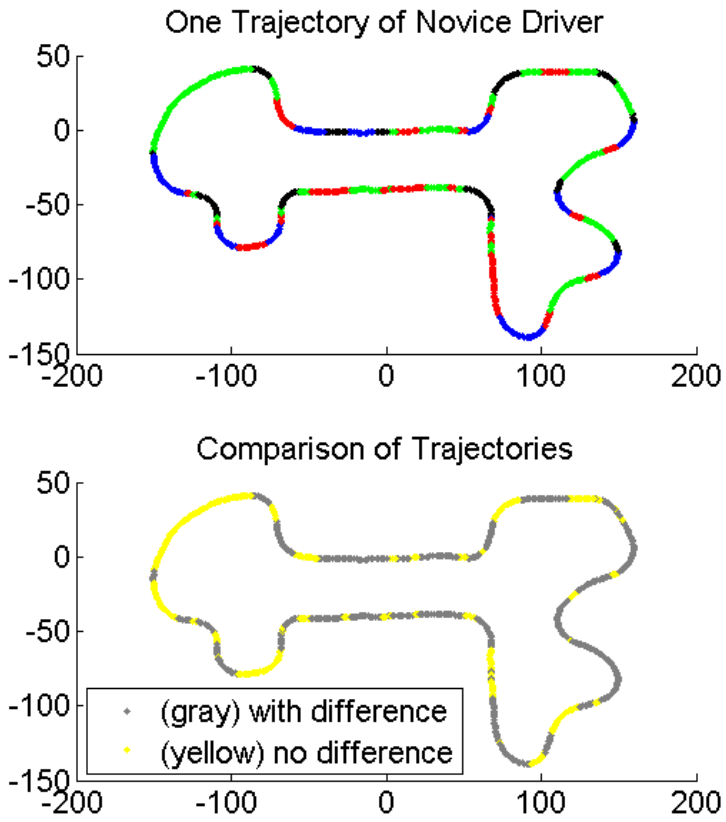


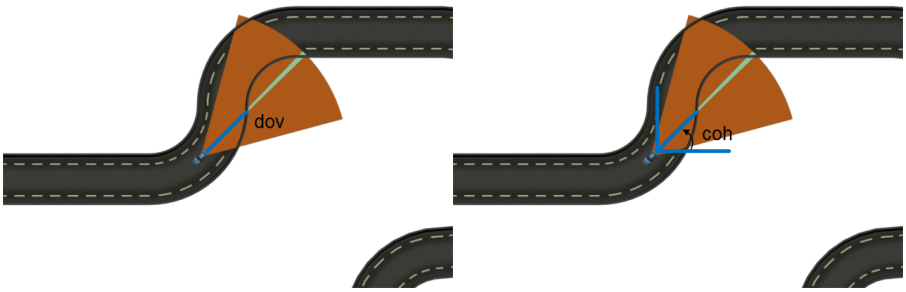
Figure 7.4: The plot of comparison between expert and novice drivers' performances.

The key-idea is we construct two models for a given unique driving process. One model is build based on observations of driving data, i.e. tAcc abd rVel, and another model is build according to observations from the road, i.e. dov and coh. We believe that these two models which are two different representations of one process, should behave in similar manner. To be more specific, we assume the 2d space of tAcc-rVel is divided into four classes. We train four HMMs based on the partitioned driving data. The same partitioning criterion is applied on the 2d space of dov-coh, the four other HMMs are trained according to their partitioning result. Because these two sets of HMMs, are modelled one process in different ways, they should show similar performance in recognizing correct belonging classes.

The evaluation result is given in Figure 7.8. From the table, the correct classification rate for [tAcc, rVel] is about 95% and for [dov, coh] is about 80%. In the same

Recognized Class		Feedback Hint
Expert	Novice	
1	2	turn right
	3	increase speed
	4	turn right while increase speed
2	1	turn left
	3	turn left while increase speed
	4	increase speed
3	1	decrease speed
	2	turn left while decrease speed
	4	turn left
4	1	turn right while decrease speed
	2	decrease speed
	3	turn right

Figure 7.5: Possible feedback hint in case of difference detection.



(a) dov (b) coh

Figure 7.6: The meaning of dov and coh.

way we provided feedback hints in previous section, we can do here by comparing the resultant belonging classes. Although the 80% rate is not perfect but it reveals that we are in right direction and proposed system has the potential of generalization. It is all about finding some features from road which are consistent with the features from driving. The improvement of this method is left for the future work.

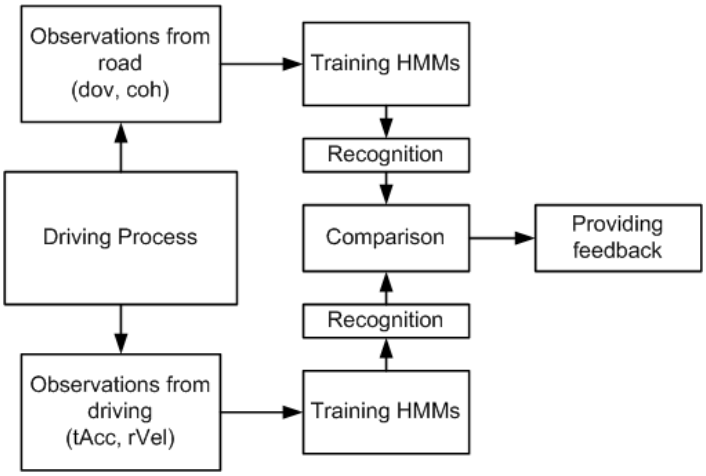


Figure 7.7: The building block diagram of the road-independent method for providing feedback.

Recognition result					
	HMM#1	#2	#3	#4	Avg
[tAcc, rVel]	0.9286	0.8913	1.0000	1.0000	**0.9550**
[dov, coh]	0.8651	0.7880	0.8667	0.6889	**0.8022**

Figure 7.8: The recognition result for road-independent method.

Chapter 8

Conclusions

The current work is an attempt to answer a fundamental question: is there any way to capture and transfer driving skills from an expert driver to a novice trainee? In another sense we want to know 1) how an experienced driver behaves in various driving situations, 2) how we can represent those behavioural skills and lastly 3) how we can make use of them to provide some appropriate multi-modal feedback to the trainee during driving training session. In this project we focused on first and second problem but always we had this in mind that we should introduce a framework which can be easily expanded to cover the third problem.

One way to address the problems of interest is looking at them as a modelling problem. If we can construct a model which represents the way human *reacts* in response to some input *actions*, we already have the answers. In case of driving application, we drew our attention to driver's decisions or driving rules and how we can define and recognize them. In fact, the way a driver makes driving decisions is a kind of realization of driver's driving behaviours and skills. So we suggest that by studying driver's decisions we can collect some useful insights about driving skills. As a matter of fact, driving behaviours are spread over and embedded inside driving signals. Driving signals might come from different sources, for example pedal and wheel, velocity and heading, and some environmental circumstances. Different kinds of driving behaviours can be extracted from these driving signals.

As a first step, we needed to collect some driving signals. An experimental set-up based on Logitech G27 Racing Wheel in developed for this purpose. Some driving signals including acceleration pedal position, steering wheel angle, velocity and heading of vehicle, are stored during driving the simulated task by subject driver for further

analysis. Inspired by real driving situation, two features are extracted by taking first-order derivative of velocity and heading signals, we named them *translational acceleration* (tAcc) and *rotational velocity* (rVel). In our defined 2d feature space, rVel placed in y-axis and tAcc in x-axis. In fact the key-idea of the main contribution was the way we treated this feature space. Later an automatic segmentation method is developed to divide driving signals into a number of segments. It is based on detecting local extrema of velocity and heading signals ans setting them as boundaries of segment. From the fact that not all of these segments comes from different sources, we looked for some appropriate partitioning methods to group similar segments. Later we realized that it is not necessary to to do partitioning based on data segments, we can do it on data samples either. In this way we proposed three different methods, threshold-based, GMM-based and hierarchical partitioning. The main idea behind them is dividing the 2d feature space in a way that data samples in each partition have similar physical interpretation. We suggested that each class is an abstract of a driving behaviour. So for each class one continuous HMM is trained and later is used as recognizer in evaluation process. To evaluate optimized HMMs, the method of sliding windows over testing data set is hired. The high average correct classification percentile, between 85% to 95% depends on the partitioning criteria, reveals the effectiveness of proposed approach in modelling and recognizing driving behaviours. We are being able to only to recognize the original driver's behaviours but also we can use them for recognizing driving behaviours of another driver.

According to the proposed series of methods, we have shown that we are being able to identify and recognize driving behaviours in the form of some classes in the time-domain feature space. The feature space based on translational acceleration and rotational velocity has shown the capability to be divided into a number of classes having unique physical meaning attached. By this configuration, the trained hidden Markov models according to each class, have shown a high accuracy in recognition correct belonging class. The average correct classification rate over than 95% is enough promising to make use of the proposed algorithms applicable. As an immediate application, we did examine their performance in providing some feedbacks to trainees. Through the road-dependent and road-independent methods, this capability is evaluated. From the former method we can exactly compare novice and expert drivers performance in each sample point, if the belonging classes are different an appropriate feedback hint can be provided. In contrast, the latter method, remove the dependency to the road and

generalize the approach. From the proposed method we can achieve 95% accuracy to recognize driving behaviours and 80% accuracy to recognize the road features. We just wanted to show the potential of driving behaviours for providing feedback and further improvements are left for future work.

The first priority for feature work is to make use of current achievement to address the third problem, transferring driving skills to novice trainee by the way of comparing his performance to an expert driver's and providing some multi-modal appropriate feedbacks. In addition we are willing to examine various training methods such as guidance and disturbance to measure the capability of identified driving behaviours in this area.

요 약 문

이 작품에서 우리는 숨겨진 마르코프 모델 (HMMs)를 사용하여 인간의 운전 행동을 모델링 문제를 해결. 그것은 전문 드라이버에서 연수 초보자에게 운전 기술을 캡처 및 전송을 향해 더 큰 목적의 일부입니다. 우리는 운전 행동이 규칙을 만드는 운전자의 결정 결과에 믿습니다. 그래서 우리는 운전자의 의사 결정을 식별하고 인식하는 우리의주의를 그린 다른 의미에서 운전 시간 시리즈 신호를 사용하여 규칙을 운전. 이를 위해, 먼저 상용 레이싱 휠에 따라 운전 시뮬레이터는 원하는 운전 작업을 시뮬레이트하기 위해 개발되고 있습니다. 가속 페달 위치, 핸들 각도, 속도와 차량의 제목을 포함하여 필요한 운전 신호는 운전 시뮬레이터를 사용하여 수집하고 있습니다. 그러면 드라이버가 그들을 통제하는 유일한 변수는 속도와 방향있다는 사실에서 영감을, 첫 주문 유도체는 운전 패턴의 가장 중요한 두 가지 특징으로 압축이 풀립니다. 같은 영감에 따라 우리는 세그먼트의 번호로 지역 최소 및 제어 변수와 분할 데이터 샘플의 최대 점을 감지하기 위해 자동으로 세분화 방법을 개발했습니다. 우리는 각 세그먼트 동안 제안, 드라이버는 페달과 바퀴 작업이 변하지 유지합니다. 모든 데이터 세그먼트는 서로 다른 소스에서 온 아니라, 그룹과 유사 세그먼트 몇 가지 기준이있을 수 있습니다. 이 라인에서 우리는 모든 클래스의 숫자에 2 차원 기능 공간을 나누지에서 유래, 모바일지도 기반 임계값 기반 및 계층, 세 파티션 방법을 제안했습니다. 우리의 신념에서 데이터 클래스가 동작을 운전 시간 도메인

실현하고 있습니다. 각 클래스에 따르면, 한 음의 매개 변수가 운전 동작을 인식 나중에 사용할 수 있도록 최적화되어 있습니다. 달성 높은 평균 올바른 분류 율은, 파티션 기준에 따라 % 95 % 85 사이, 분류 및 구동 동작을 인식에 제안된 접근 방법의 효능을 나타낸다. 마지막으로 우리는 몇 가지 피드백을 제공하기 위해 전문가와 초보자 드라이버 '공연을 비교하는 행동 인식을 사용했습니다. 두 가지 방법 길이 하나에 의존하고 도로 독립이 엔드에 대한 제안 아르 또. 평가 결과는 제안된 방법의 적용을 입증.

Bibliography

[1] P. Boyraz, X. Yang, A. Sathyanarayana, and J. Hansen, "Driver behaviour modeling using hybrid dynamic systems for driver-aware active vehicle safety," in *21st International Technical Conference on Enhanced Safety for Vehicle*, 2009.

[2] X. Zhang and F. Naghdy, "Human motion recognition through fuzzy hidden markov model," in *Proceedings of the 2005 International Conference on Computational Intelligence for Modeling, Control and Automation, and International Conference on Intelligent Agents, Web Technologies and Internet Commerce (CIMCA-IAWTIC05)*, 2005.

[3] R. Hess and Modjtahedzadeh, "A control theoretic model of driver steering behavior," *IEEE Control Systems Magazine*, pp. 3–8, 1990.

[4] R. Hess, "A model based theory for analyzing human control behavior," *Advances in Man-Machine System Research, W.B.House (ed.)*, vol. vol.2, London, JAI Press, pp. 129–175, 1985.

[5] D. McRuer and D. Weir, "Theory of manual vehicular control," *Ergonom*, vol. vol. 12, pp. 599–633, 1969.

[6] C. MacAdam, "Application of an optimal preview control for simulation of closed-loop automobile driving," *IEEE Trans. Syst. Man. Cybrn.,*, vol. SMC-11, pp. 393–399, 1981.

[7] D. McRuer and E. Krendel, "Mathematical models of human pilot behavior," *AGARDograph*, vol. 188, p. 1974.

[8] T. Pilutti and A. Ulsoy, "Identification of driver state for lane keeping tasks," *IEEE Trans. on Syst., Man, Cybern., Part A: Syst. And Humans*, vol. 29, No. 5, 1999.

[9] I. Lubashevsky, P. Wagner, and R. Mahnke, "Bounded rational driver models," *European Physical*, vol. 31, p. 243, 2002.

[10] L. Chen and A. G. Ulsoy, "Identification of a driver steering model, and model uncertainty from driver simulator data," *Journal of Dynamic Systems, Meausrement and Control*, vol. 123, pp. 623–629, 2001.

[11] M. Nechyba and Y. Xu, "Human control strategy: Abstraction, verification and replication," *IEEE Control Systems Magazine*, vol. 17, no. 1, pp. 48–61, October 1997.

[12] M. Nechyba and Y. Xu, "On discontinuous human control strategies," in *Proc. IEEE Int. Conference on Robotics and Automation*, vol. 3, May 1998, pp. 2237 – 2243.

[13] A. Pentland and A. Liu, "Modeling and prediction of human behavior," *Neural Computation*, vol. 11, pp. 229–242, 1999.

[14] J. Yang, Y. Xu, and C. S. Chen, "Human action learning via hidden markov model," in *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans, Vol. 27, No. 1*, 1997.

[15] D. Mitrovic, "Reliable method for driving events recognition," *IEEE Transactions on Intelligent Transportation Systems*, vol. VOL. 6, NO. 2, p. 8, 2005.

[16] K. Torkkola, S. Venkatesan, and H. Liu, "Sensor sequence modeling for driving," *Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference*, p. 6, 2005.

[17] P. Boyraz, M. Acar, and D. Kerr, "Signal modelling and hidden markov models for driving manoeuvre recognition and driver fault diagnosis in an urban road scenario," in *Proceedings of the 2007 IEEE Intelligent Vehicles Symposium*, 2007.

[18] A. Sathyanarayana, P. Boyraz, and J. Hansen, "Driver behaviour analysis and route recognition by hidden markov models," in *IEEE International Conference on Vehicular Electronics and Safety*, 2008.

[19] J. Michon, "Explanatory pitfalls and rulebased driver models," *Accident Analysis and Prevention*, vol. 21, No.4, pp. 341–353, 1989.

[20] D. D. Salvucci, "Modeling driver behavior in a cognitive architecture," *Human Factors*, vol. 48, pp. 362–380, 2006.

[21] U. Kiencke, R. Majjad, and S. Kramer, "Modeling and performance analysis of a hybrid driver model," *Control Engineering Practice*, vol. 7, pp. 985–991, 1999.

[22] J.-H. Kim, S. Hayakawa, T. Suzuki, K. Hayashi, S. Okuma, N. Tsuchida, M. Shimizu, and S. Kido, "Modeling of drivers collision avoidance maneuver based on controller switching model," in *IEEE Transactions on Systems, Man, and CyberneticsPart B: Cybernetics, Vol. 35, No. 6*, 2005.

[23] T. Akita, S. Inagaki, T. Suzuki, S. Hayakawa, and N. Tsuchida, "Hybrid system modeling of human driver in the vehicle following task," in *SICE Annual Conference*, 2007.

[24] K. Inata, P. Raksincharoensak, and M. Nagai, "Driver behavior modeling based on database of personal mobility driving urban area," in *Int. Conf. On Control Automation and Systems*, 2008.

[25] L. R. Rabiner and B. H. Juang, "An introduction to hidden markov models," *IEEE ASSP Magazine*, p. 13, 1986.

[26] W. Takano, A. Matsushita, K. Iwao, and Y. Nakamura, "Recognition of human driving behaviors based on stochastic symbolization of time series signal," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, p. 6, 2008.

[27] S. Calinon and A. Billard, "Stochastic gesture production and recognition model for a humanoid robot," in *Proceedings of 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2004.

[28] C. S. Hundtofte, G. D. Hager, and A. M. Okamura, "Building a task language for segmentation and recognition of user input to cooperative manipulation systems," in *Proceedings of the 10th Symp. On Haptic Interfaces For Virtual Envir. & Teleoperator Systs. (HAPTICS'02)*, 2002.

[29] T. E. Murphy, C. M. Vignes, D. D. Yuh, and A. M. Okamura, "Automatic motion recognition and skill evaluation for dynamic tasks," in *Eurohaptics*, 2003.

[30] U. Galassi, “Structured hidden markov model: A general tool for modeling process behavior,” Ph.D. dissertation, Dipartimento di Informatica, Universit‘a degli Studi di Torino, 2008.

[31] S. Buechler, “Statistical models in r,” Department of Mathematics, University of Notre Dame, Tech. Rep., 2007.

[32] L. R. Rabiner, “A tutorial on hidden markov models and selected applications in speech recognition,” in *Proceedings of the IEEE, Vol. 77, No. 2*, 1989.

[33] C. Miyajima, Y. Nishiwaki, K. Ozawa, T. Wakita, K. Itou, and K. Takeda, “Driver modeling based on driving behavior and its evaluation in driver identification,” in *Proceedings of the IEEE*, 2007.

[34] E. B. Pizzolato, M. dos Santos Anjo, and G. C. Pedroso, “Automatic recognition of finger spelling for libras based on a two-layer architecture,” in *Proceedings of the 2010 ACM Symposium on Applied Computing*, ser. SAC ’10. New York, NY, USA: ACM, 2010, pp. 969–973. [Online]. Available: http://doi.acm.org/10.1145/1774088.1774290

[35] L. Yang, B. K. Widjaja, and R. Prasad, “Application of hidden markov models for signature verification,” *Pattern Recognition*, vol. 28, pp. 161–169, 1995.

[36] J. R. Deller, J. G. Proakis, and H. L. Hansen, *Discrete-Time Processing of Speech Signals*. New York: Macmillan, 1993.

Acknowledgements

I would like to express my deep-felt gratitude to my advisor, Prof. Seoungmoon Choi of the Computer Science and Engineering Department at Pohang University of Science and Technology, for his advice, encouragement, enduring patience and constant support. He was never ceasing in his belief in me (though I was often doubting in my own abilities), always providing clear explanations when I was (hopelessly) lost, constantly driving me with energy when I was tired, and always giving me his time, in spite of anything else that was going on. It was a great pleasure working with and learning from my advisor and committee members, in addition to several other outstanding professors, staff, and students colleagues at POSTECH. In addition I would like to thanks my lab-mates who helped me a lot in this journey. I would also like to extend my sincere appreciation for continuous support from my parents and family.

CURRICULUM VITAE

Name: Reza Haghighi Osgouei
Date of Birth: 1977.04.14
Place of Birth: Urmieh, West Azerbayjan, Iran.
Address: No. 115, Haptics and Virtual Reality Laboratory (HVRlab), Science Bld. 4, Pohang University of Science and Technology (POSTECH), Pohang, Gyungbuk, 790-784, Republic of Korea.

EDUCATION

1994-1999: B.Sc. in Electrical Engineering: Control Systems, Sahand University of Technology, Iran.

2000-2003: M.Sc. in Electrical Engineering: Control Systems (non-degree graduate study), Amirkabir University of Technology (Tehran Polytechnic), Iran.
Thesis title: Real-time collision detection of mobile robot during tele-operation using haptic interface through virtual environment.

2010-2012: M.Sc. in Computer Science and Engineering, POSTECH, Korea
Thesis title: Driver Behaviour Recognition Using Hidden Markov Models, Advisor: Prof. Seungmoon Choi.
Honours: *Summa Cum Luade*.

AFFILIATION

1. Institute of Electrical and Electronics Engineers (IEEE)
2. Korean Institute of Information Scientists and Engineers (KIISE)
3. Iranian Society of Instrument & Control Engineers (ISICE)
4. Iranian Association of Electrical and Electronics Engineers (IAEEE)
5. Iranian National Electro-technical Committee (INEC)

ENGLISH LANGUAGE PROFICIENCY

10 Dec 2011: IELTS Academic Overall Band Score: 7.5.

ADDITIONAL TRAININGS

Jan 2008:	ISO 9001:2000 Requirements & Documentation, URS.
Jan 2008:	ISO 9001:2000 Internal Audit, URS.
May 2007:	Strategic Planning, Experts Group.
Aug 2005:	Keyence PLC (BASIC, ADVANCE, PROFESSIONAL), Barg Computer Co.
Jul 2005:	SIMATIC WinCC & S7, NEDA Training Centre.

PUBLICATIONS

1. Reza Haghighi Osgouei, Seungmoon Choi, “Driving Pattern Recognition using Hidden Markov Models and GMM-based Clustering”, In Proceedings of HCI Korea, Republic of Korea, Jan 11-13, 2012.
2. Reza Haghighi Osgouei, Seungmoon Choi, “Modeling and Recognition of Human Driving Behavior using Hidden Markov Models”, In Proceedings of 38th KIISE Conference, vol. 38, no. 2(B), pp. 323-326, Republic of Korea, 2011 (winner of the best student paper award).
3. Reza Haghighi Osgouei, “Haptic Tele-Driving of Non-autonomous Mobile Robot Using a Haptic Interface”, IEEE International Symposium on Computational Intelligence in Robotics and Automation, CIRA 2009, Daejeon, Republic of Korea (South Korea), 2009.
4. Reza Haghighi Osgouei, H. A. Talebi, “Improvement of Human-Machine Interaction using Haptic Interfaces”, 5th Conference on Intelligence Systems, Ferdowsi University of Mashhad, Iran, 2003, (In Persian).
5. Reza Haghighi Osgouei, “Haptic Feedback usage in Education”, 8th Iranian Computer Society Annual Conference, Ferdowsi University of Mashhad, Iran, 2002, (In Persian).