Reliable Method for Driving Events Recognition

Dejan Mitrović

***Abstract*—Motor vehicles greatly influence human life but are also a major cause of death and road congestion, which is an obstacle to future economic development. We believe that by learning driving patterns, useful navigation support can be provided for drivers. In this paper, we present a simple and reliable method for the recognition of driving events using hidden Markov models (HMMs), popular stochastic tools for studying time series data. 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.**

***Index Terms*—Driving events, hidden Markov models (HMM), pattern recognition.**

I. Introduction

STEADY development of motor vehicles during the 20th century has greatly influenced human life, providing much greater mobility for people and products and thus enabling advances in many other areas. However, it is not possible to match fast increase in vehicle speed and availability, with appropriate changes in road infrastructure and in human's driving capabilities. On the contrary, road congestion has become an obstacle for future economic development and, even worse, motor vehicle accidents have become one of the major causes of death.

Vehicle driving is an act of operating and directing the course of a vehicle. For many humans, driving is a very easy process, perceived as a natural extension of their capabilities. However, driving is a complex decision-making process due to the multidimensionality of the problem (space-time), the complex relationships between the major entities involved in driving (driver, vehicle, and environment) and the dynamic nature of these entities (for example, the driving capabilities of a person vary depending on their age, the time of day, their mental, physical, and emotional state, etc.).

We use the term "driving event" to refer to any major change in vehicle attitude or speed, such as a left or a right turn, a stop, making a U-turn, etc. From the decision-making point of view, driving events are demanding as they require the driver to act on vehicle controls while, at the same time, the relationship between the vehicle and the environment (road infrastructure) is far more complex than between events. In these circumstances, driving safety margins are very small and even a minor disturbance can cause an accident situation. Accordingly, driving events have a very significant role in driving safety research.

In this paper, we will present a method for driving event recognition using hidden Markov models (HMMs). In Section II, we reflect on the broader scope of our research, followed by the brief discussion of the relevant work. The basic concepts of the HMMs are presented next. The following sections describe the data acquisition hardware and software and discuss the experiments with HMMs and the results obtained. Conclusions are presented in Section VII.

II. Learning Driving Patterns

There are a number of frequently occurring patterns in driving. On the higher level, it is very likely that the vast majority of everyday driving routes will be to the same places (job, children's schools, favorite restaurant) using the same path. Repeating the known route reduces driving strain and is a logical choice for any driver. A route can be represented as a sequence of driving events and the similar pattern of events is repeated every time a driver uses the same route. On the lower level, driving actions also represent patterns. Turns, for example, consist of the following actions: the driver first reduces speed and changes to a lower gear, then rotates the steering wheel, next increases speed, changes up a gear, and finally rotates the steering wheel to the other direction. This set of actions will be repeated every time a driver executes a turn.

We believe that by identifying and learning someone's driving patterns, we can provide useful information which can help the driver to fulfill a range of different driving navigational tasks, particularly guidance tasks. Guidance driving tasks are also known as tactical or intermediate. They involve selecting and executing maneuvers to achieve the short-term objectives [1], resulting in a sequence of driving events.

We believe that if we can reliably recognize driving events, we can build the model of the current situation relevant for vehicle guidance navigation. From this model, and by using the pattern history of driving events, we may predict future driving events [2]. One of the possible applications of predicted driving events is to monitor the driver's behavior by comparing

Manuscript received January 21, 2004; revised June 23, 2004. This work was supported by the Road Safety Trust New Zealand. The Associate Editor for this paper was M. Kuwahara.

The author is with the Trimble Navigation, Christchurch, New Zealand (e-mail: dejan_mitrovic@trimble.com).

Digital Object Identifier 10.1109/TITS.2005.848367

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

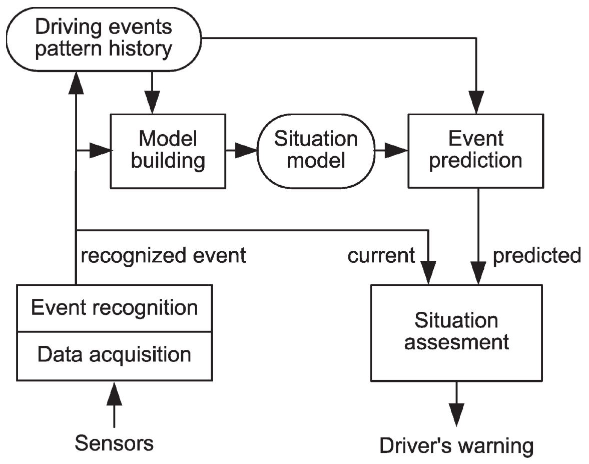

Fig. 1. The framework of the driver warning system based on learning driving patterns.

the actual sensor data with the predicted event. The observed difference between predicted and actual data could be used to detect possible accident situations and to generate appropriate warnings. The framework of the driver warning system based on learning driving patterns is presented in Fig. 1. The "model builder" is a learning module; it uses the pattern history and the currently recognized driving event to make the model of current situation for driver–vehicle–environment system. This model also contains temporal information (unfolding the situation from the recent past) and reliability assessment (how reliably the model represents the actual situation). To be usable, the framework should operate in real time; each new driver's experience will be used to build the situation model, and at the same time, the model will be used to predict the next driving event.

Learning from previous driver's experiences has many advantages over conventional systems. It allows adaptation to a particular environment or to the individual driver's style. For example, a set of driving patterns for the same road could be very different in the morning than during the night, and for some purposes, these may be treated as two different driving experiences. Consequently, the situation model should be able to differentiate between these two cases. On the other hand, two segments of a driving route on different geographic locations can be very similar from the guidance perspective, in which case the situation models may be equivalent. Systems that are not based on learning techniques cannot identify and use such similarities.

Recognized driving events could be used for other purposes as well. For example, digital road map matching is a process of matching data from vehicle sensors to the digitized road map stored in the computer in order to recognize the present vehicle location [3]. This technique is often used to augment global positioning system (GPS)-based navigation systems in "urban canyons," where GPS signal is not available, or to provide dead-reckoning-only navigation solution. Reliable recognition of driving events could significantly improve existing map-matching techniques. Additionally, recognized driving events could be used to compare driving behavior over time. Tiredness is a very often cause of traffic accidents. If we are able to reliably recognize driving events, we can evaluate how driver behavior changes over time. We believe that it is possible to detect the moment when a driver's behavior becomes too different from his/her usual behavior before the driver and vehicle pass the safety limits and an accident occurs.

Our goal is to prove that a very simple and inexpensive system can provide useful information to the driver and increase the safety of driving. We believe that comprehensive driver support systems, or autonomous driving systems in the future, must consist of many components, which support different navigation tasks. The framework described above is only one possible component of such a system. For this reason, we believe that complexity and cost of components must be kept at minimum. Our goal is to provide a system which is based on components which are already present in modern cars and standard and well-known and robust data-processing and recognition techniques.

III. Related Work

Navigation on the basis of previous experiences was a research focus in the area of autonomous mobile robots for some time. Balch and Arkin [4] used a grid-based local spatial memory to help their reactive navigation system avoid areas that had already been visited. HMMs were successfully used by Aycard *et al.* [5] to recognize features in a corridor (angles, open doors, etc.) by using data from infrared and ultrasonic sensors mounted on a mobile robot.

Following these results, similar research is done in the area of intelligent transportation systems. Nechyba and Xu [6] use HMMs and data from a driving simulator to compare human control strategies. One model was trained for each driver, and probabilities are cross-evaluated for data generated by all drivers on the same route. System was able to discriminate between various drivers by using a simple similarity measure.

Researchers from the Massachusetts Institute of Technology and the Nissan Cambridge Basic Research Center, Pentland and Liu [7] modeled human driving behavior as a Markov chain of control states, where each state is defined by a dynamic control model. They collected data from the instrumented Nissan 240SX driving simulator such as steering wheel angles and brake and accelerator positions. A number of subjects drove through the simulated world in which they had to execute a vehicle maneuver as a result of the text command presented on a simulator screen. The commands presented were: stop, turn left or right on the next intersection, change line, and pass the car in front. The HMMs were trained to recognize driving events 2 s after the text command was presented to the subject. The goal of the experiment was to predict driver's intentions based on his first actions while executing the command. Authors reported mean recognition accuracy of 95.24%, proving that human driving intentions could be accurately recognized very soon after the beginning of the action.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

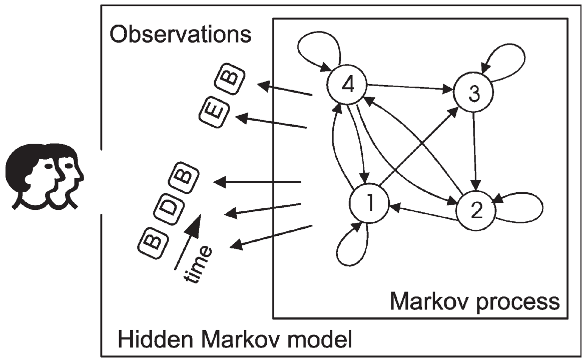

Fig. 2. The basic concepts of hidden Markov models.

IV. Hidden Markov Models

The HMMs are probabilistic tools for studying time series data. They attracted great attention in the pattern recognition research community due to their success in a range of applications, notably in speech recognition. Baum *et al.* [8], [9] published the theory of HMMs in a series of articles in the late 1960s. In this section, we present only the basic concepts. Interested readers can find detailed tutorials on HMMs in [10], [11].

The foundation for the HMM is a stochastic Markov process consisting of a number of states and the transitions between them. At discrete time intervals, the Markov process moves from one state to another (or to the same state) according to a set of transition probabilities. State changes in the underlying Markov process are hidden from the user (not observable). However, the user can observe process changes through a sequence of observations, which are produced at each state change as shown in Fig. 2. Observations could be continuous signals or discrete symbols from the finite alphabet. The stochastic process of observation generation is defined by the output probability distributions for each state. The output probability distribution can be discrete or continuous, depending on the type of observations. Both types have their own advantages, but the low computation cost of discrete models has made them much widely used.

More formally, discrete HMMs can be characterized as follows.

- A set of N distinct states $S = \{S_1, S_2, \dots, S_N\}$, with q_t denoting a state at time t.
- The initial state distribution $\pi = \{\pi_i\}$, where $\pi_i = P[q_1 = S_i]$, $1 \leq i \leq N$.
- The state transition probability distribution $A = \{a_{ij}\}$, where $a_{ij} = P[q_{t+1} = S_j | q_t = S_i]$, $1 \leq i, j \leq N$.
- Each state can produce one of M distinct observation symbols from the set $V = \{V_1, V_2, \dots, V_M\}$.
- The observation symbol probability distribution in state j, $B = \{b_j(k)\}$, where $b_j(k) = P[v_k \text{ at } t | q_t = S_j]$, $1 \leq j \leq N$, $1 \leq k \leq M$.

Therefore, a HMM λ can be specified as $\lambda = (N, M, A, B, \pi)$. The probability that observation sequence $O = O_1 O_2 \cdots O_t$ is generated by the model λ is denoted as $P(O|\lambda)$.

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 [12]. Left-to-right models are used in a number of temporal pattern recognition applications such us speech recognition [10], [11], human gesture recognition [13], and signature verification [12], with great success. As the driving event recognition problem is similar to the abovementioned domains, we restricted our attention to left-to-right models.

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 reestimated.

Evaluation is a procedure for calculating the probability of a particular observation sequence being generated by the model with given parameters $[P(O|\lambda)]$. In our experiments, we used a "forward–backward" algorithm [9] for evaluation. The forward variable α is defined as

$$\alpha_t(i) = P(O_1 O_2 \cdots O_t,\ q_t = S_i \mid \lambda).$$

It is easy to conclude that the observation sequence probability for a given model can be calculated as

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i).$$

The forward variable could be easily calculated using a dynamic programming approach as follows

$$\alpha_1(i) = \pi_i b_i(O_1), \quad 1 \leq i \leq N$$

$$\alpha_{t+1}(i) = \left[\sum_{i=1}^{N} \alpha_t(i) a_{ij}\right] b_i(O_1), \quad 1 \leq j \leq N, \quad 1 \leq t \leq T-1.$$

The observation sequence probability could alternatively be calculated using a backward variable $\beta_t(i) = P(O_{t+1} O_{t+2} \cdots O_T,\ q_t = S_i | \lambda)$, which could be calculated using a similar inductive algorithm as for the forward variable. Note that the evaluation requires only the forward or backward variable to be computed; however, as both variables are required for training, they are usually evaluated together.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

For training, we used the popular Baum–Welch reestimation method. This is an iterative procedure which starts with initial model parameters that are set randomly or uniformly (or by using some other appropriate procedure). Training sequences are evaluated using the forward–backward algorithm, and the calculated forward and backward variables are used to reestimate model parameters using the following formulas

$$\overline{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)}$$

$$\overline{b}_{jk} = \frac{\sum_{\substack{t=1 \\ O_t = v_k}}^{T-1} \alpha_t(j) \beta_t(j)}{\sum_{t=1}^{T-1} \alpha_t(j) \beta_t(j)}.$$

Evaluation and reestimation 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 30 iterations of the Baum–Welch algorithm with different initial model parameters. The model with the highest probability was chosen as the training result.

Data for our HMM experiments consist of a number of short observation sequences (5–30 observations per sequence), which are grouped according to the type of driving event they represent. A small number of observations in the training set, and an observation probability distribution that is far from uniform, lead to typical numerical problems encountered in HMM implementations. The forward and backward variables are computed by multiplying probabilities and in such a circumstance they could easily become smaller than the computer numeric precision. In order to prevent this, forward variables are rescaled after each iteration. The inverse scale factor is applied to backward variables in order to cancel the effect of scaling on reestimation formulas.

Small observation probabilities could cause the observation probability for some observation symbols to become zero. This effectively removes some transitions from the model. Due to the reestimation nature of training procedure, every removal is permanent and cannot be repaired in the next iteration. In order to avoid this problem, a simple flooring method is implemented. If, after reestimation, we have $\overline{b}_{jk} = 0$, we replace it with a small value ε. This simple approach prevents this numerical problem without compromising the calculated probability. More complex algorithms exist [14], but we found flooring quite appropriate for our application.

Due to short observation sequences, each training iteration must include all available training sequences, or else many reestimated parameters become zero. Therefore, we use modified reestimation formulas for training by multiple observation sequences [12]

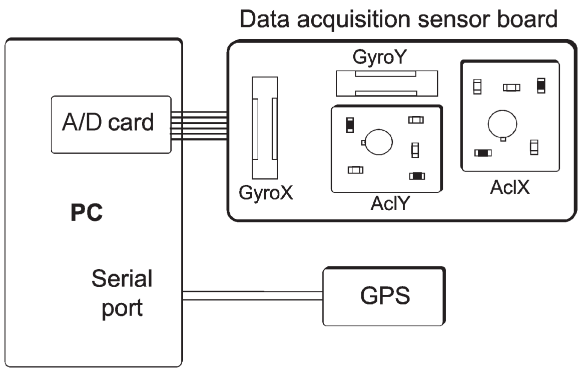

Fig. 3. Data acquisition hardware.

$$\overline{a}_{ij} = \frac{\sum_{k=1}^{Q} \sum_{t=1}^{T-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^{Q} \sum_{t=1}^{T-1} \alpha_t^k(i) \beta_t^k(i)}$$

$$\overline{b}_{jk} = \frac{\sum_{k=1}^{Q} \sum_{\substack{t=1 \\ O_t = v_k}}^{T-1} \alpha_t^k(j) \beta_t^k(j)}{\sum_{k=1}^{Q} \sum_{t=1}^{T-1} \alpha_t^k(j) \beta_t^k(j)}.$$

V. Sensors and Data Acquisition

To provide a basis for our experiments, data acquisition hardware and software were developed. The data acquisition hardware consists of a number of sensors, a data acquisition card, and a portable computer as shown in Fig. 3. The accelerometers are set to measure longitudinal and lateral vehicle acceleration. We use Analog Devices ADXL05 accelerometers and micromachined uniaxial sensors with a movable beam (often used as air-bag sensors). Two gyroscopes are mounted to measure the horizontal rotation (left and right vehicle turns) and the changes in road slope. Gyroscopes and accelerometers are mounted on a separate sensor board that could be easily fixed to the car interior. Sensors provide analog output digitized by an eight-channel, 12-bit data acquisition card mounted inside a desktop personal computer (PC) modified to work in a car. A GPS receiver is used as a velocity sensor to avoid otherwise necessary changes to the vehicle electronics (as we used a private car for experiments). A Trimble Placer 400 GPS receiver is connected to the PC through a serial connection. Positions provided by the GPS receiver are also collected to serve as a reference during data analysis but are not used for driving event recognition.

Following our goal to implement the simplest possible system, we decided not to use gyroscope data for the HMM experiments. Gyroscopes are not often used in vehicle electronics and are much more expensive than accelerometers. As will be proved later, we found speed and acceleration data sufficient for the driving event recognition.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

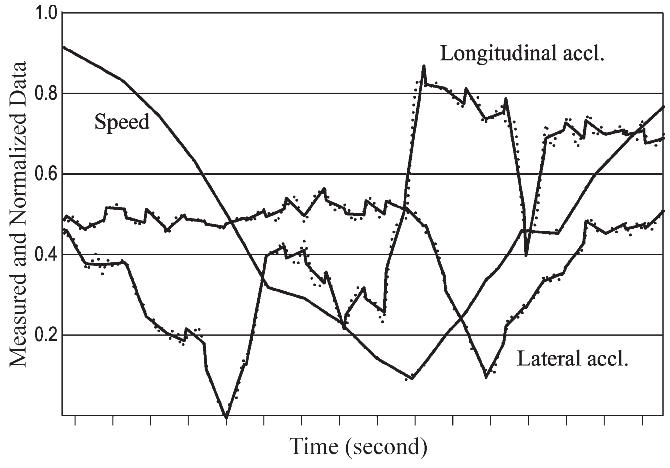

Fig. 4. Data before and after the waveform segmentation.

Analog data from sensors are collected in 50 ms intervals. Velocity data from the GPS receiver are generated every second, and we interpolated it to match data from accelerometers. A range of digital signal-processing techniques is used to prepare data for further processing. Actual data reading from analog-to-digital (A/D) converters is done at a much higher rate, and data are averaged to increase the signal-to-noise ratio. The driving events that we are interested in are very slow ones (even the simplest operation such as a gear change normally takes more than 1 s), and we use the second-order normalized low-pass Butterworth digital filter with a cutoff frequency of 2 Hz to eliminate noise. Data are normalized to range 0–1 to eliminate the impact of data range differences on further processing steps.

Common driving events last for a few seconds, and the data acquisition system produces a large quantity of data (60 data items/s). In order to reduce the amount of data and simplify pattern recognition, we use a waveform segmentation technique. Normalized acceleration and speed data are split into segments (frames) of predefined length. Frames overlap each other in order to emphasize the influence of the global trend in the collected data. The least squares method for linear data approximation is used to calculate frame parameters: average (mean) value and slope (change in values). As a result, each frame is represented by five parameters: average speed, average lateral acceleration, average longitudinal acceleration, lateral acceleration slope, and longitudinal acceleration slope. Fig. 4 displays the segmented output of data collected from sensors during a right turn on an intersection. It is easy to note that all important waveform features are preserved, while the number of required data values is reduced by a factor of six (for frame size of 0.5 s).

In order to use discrete HMMs, we need to translate continuous frame parameters into symbols from a small predefined set. Vector quantization is a standard procedure for joint quantization of a set of continuous amplitude signals into one of the codebook (discrete amplitude) values [15]. It is widely used in data compression, telecommunication, and speech coding. The design of the codebook is a process of determining the set of vectors that minimize the quantization error for a given data

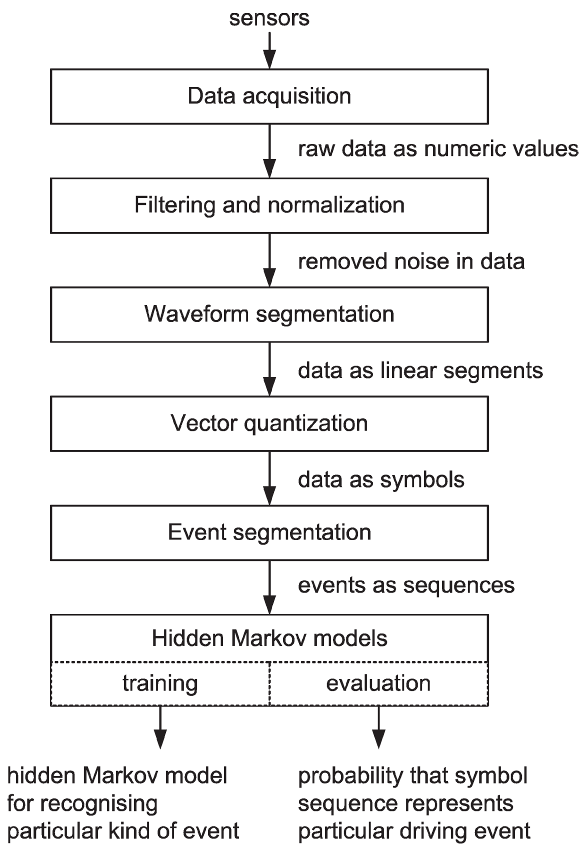

Fig. 5. The flow chart of training and recognition procedures.

set. We use a well-known iterative clustering algorithm known as the k-means algorithm. As the k-means algorithm is known to converge to the local minimum of the quantization error, we repeat it many times with different random values for the initial code vectors in order to find the best possible global solution. Input into the vector quantifier are vectors consisting of five parameters describing each frame, while output is a stream of codebook indexes which represent observation symbols for HMMs.

Selection of the codebook size is a tradeoff between smaller quantization error (for a larger codebook size) and easier HMM operations (for a smaller codebook size). We conducted a number of tests with various codebook sizes in the range of 12–36. As could be expected, an increase in codebook size reduced quantization error. However, due to HMM's stochastic nature and the numerical problems described above, HMMs trained with larger sets of observation symbols had a worse recognition rate than HMMs trained with smaller sets. The codebook size of 16 is selected as a convenient compromise. If we denote codebook indexes with uppercase letters `A`–`P`, sequences `KKICCAALL` and `KJJJJIBCCALHL` represent data from two right-turn events, encoded with the selected codebook. Observation sequences like these are used for HMM training and evaluation.

The abovedescribed data-processing steps and the recognition steps described in Section VI are summarized in the flow chart diagram shown in Fig. 5.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

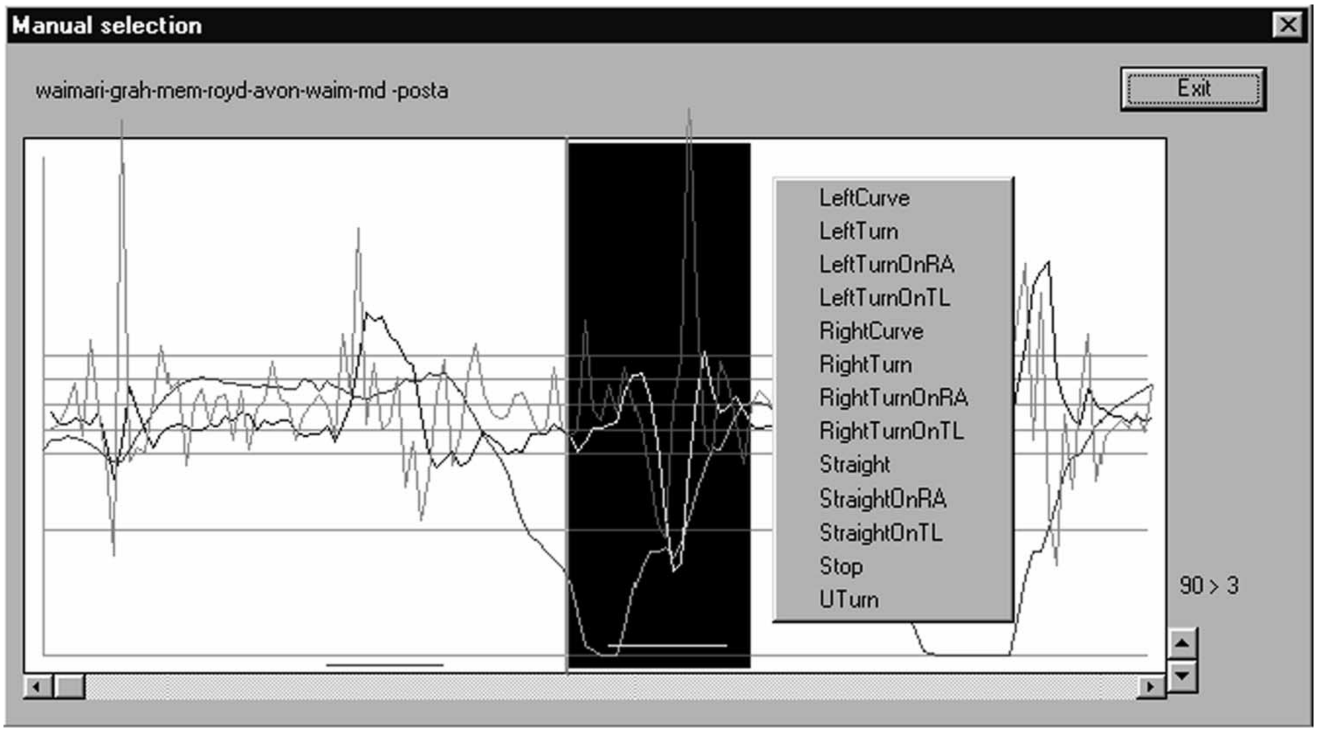

Fig. 6. A program for manual data selection and marking.

VI. Driving Event Recognition

For our experiments with HMMs, we used data from 22 test drives around northwest Christchurch, New Zealand, undertaken during late 1999. No particular routes for test drives were selected, but we tried to incorporate a variety of urban road features which could be found in a medium-sized New Zealand town. Some of the routes, or parts of routes, are repeated in order to provide data for our later experiments with driving event prediction.

Training and testing data for experiments were manually selected from test drive data. A program has been developed for manual selection of data representing a driving event and marking it as a particular type of event. A screenshot of this program is depicted in Fig. 6. Segmented values for speed and lateral and longitudinal acceleration are graphically presented. The black background represents data currently selected by the user. Our experience showed that we were able to select a driving event and determine its type from data displayed on the screen in most cases as patterns in the data are clearly visible and easy to recognize. In rare circumstances, we had to consult the route built from position data collected from the GPS receiver and displayed over the street map in the separate software.

A pop-up menu for event marking is visible on the right from the selected data. The user marks selected data by choosing one of the driving events from the pop-up menu. This way we selected and marked 263 events. We experimented with seven most common types of driving events for our experiments: driving a vehicle along left and right curves, turning a vehicle left and right on intersections, with and without roundabouts, and driving straight across an intersection with a roundabout. Roundabouts (rotary intersections) are very common traffic features in New Zealand urban areas. In total, there were 238 events of these types as shown in Table I. The same program collected additional statistical data, such as duration of the events and minimal and maximal values for data from each sensor. These data will be used for situation model and for event prediction.

We constructed seven HMM models—one for each type of driving event we wished to recognize. As mentioned above, we used left-to-right discrete HMMs. We made small preliminary tests with various model sizes (from five to eight states) in order to find an optimal model size for each driving event. If a number of HMM states are too small, the false rejection error rate increases, as the model is not flexible enough to cover all events of a particular type. On the other hand, a model with a large number of states leads to a larger error rate for false acceptance. Each model (each size for every type) was trained by all events for the given type. Then observation probabilities are found for each observation sequence from the same set. We selected the model size that had the smallest variance in output probabilities, effectively minimizing false rejections. In all cases, except for left turns, we selected HMMs with six states; for left turns, we selected the model size of five.

Each model was trained by observation sequences from the training set for the particular type of event. Training sets consist of representative subsets of events (approximately 30% of original sets). After training all seven models, we started evaluating the sequences from the complete set. For each observation

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

TABLE I
RESULTS OF THE DRIVING EVENT RECOGNITION BY HIDDEN MARKOV MODELS

	Evaluated sequences, the total number of sequences for the particular type of events						
Recognized as	RC, 41	RRA, 16	RT, 36	LC, 56	LRA, 20	LT, 59	SRA, 10
Right curve (RC)	41	0	0	4	0	0	0
Right on roundabout (RRA)	0	16	0	0	0	0	0
Right turn (RT)	0	0	36	0	0	0	0
Left curve (LC)	0	0	0	52	0	0	0
Left on roundabout (LRA)	0	0	0	0	11	18	0
Left turn (LT)	0	0	0	0	9	41	0
Straight on roundabout (SRA)	0	0	0	0	0	0	10

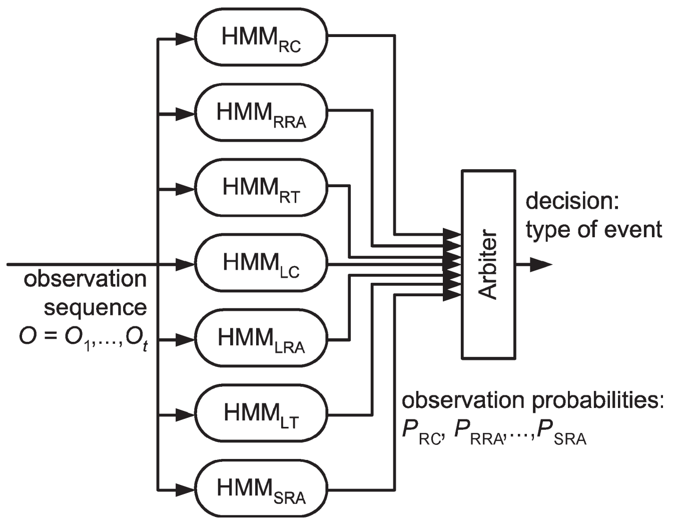

Fig. 7. Parallel evaluation of observation sequences.

sequence, we computed the observation probabilities for all models. The highest ($P_{\max}$) of these probabilities determines the driving event type to which a sequence belongs, as shown in Fig. 7. The results of sequence evaluation are displayed in Table I. Each row in the table represents the number of sequences correctly recognized as events of a given type. In an ideal case, all cells in the table should be zero, except for the diagonal values which should be equal to the number of events of the particular type.

The system successfully recognized all events except the four cases (out of 56) of left-curve events recognized as right-curve events. We suspect that a coarse vector quantization grid, which does not discriminate well between small lateral acceleration on the left and right sides, causes the above problem. Applying a weighting function to the lateral acceleration during vector quantization could easily rectify this problem. We decided not to compromise the generality of the recognition process.

Furthermore, in 29 cases (out of 79 left-turn and left-turn-on-roundabout sequences), the system could not distinguish between left turns and left turns on a roundabout. However, the geometry of intersections and the traffic rules for left turns on intersections, with or without roundabouts, are the same. Hence, this result is expected and we believe that these two event types should be regarded as one.

If we exclude the above problem, we can conclude that even with a very limited set of sensors, the system correctly recognized 234 of 238 (or 98.3%) driving events. The difference between the highest ($P_{\max}$) and the next highest ($P_{\max 2}$) probability calculated for observation sequence was large. For example, the average difference (calculated as $\sum[(P_{\max} - P_{\max 2})/P_{\max}]/N$ for right turns with and without roundabouts (which are reasonably similar events) was above 48%. That means that the probability for the correct model was almost twice the probability for the next best model. This proves that the presented model for driving event recognition is not only very accurate but also reliable and robust.

We believe that recognition rate achieved is sufficient for applications described above. We expect that recognition rate will further increase as the new training sequences become available, providing a greater variety of training data for each driving event. Once the system contains sufficiently trained models for all types of driving events, only very rare events will not be recognized. These unrecognized events may indicate dangerous or other unexpected situation. However, dangerous situations could be much more reliably detected when recognized and unrecognized driving events are analyzed in a context of a situation model as described above. If required, the recognition rate could be improved by adding new sensors (using GPS positions, for example) or by employing the domain specific knowledge in vector quantization (as discussed above).

The recognition process consists of a few steps; however, it is designed to reduce data and maintain the simplicity of required numerical operations. The processing power required for real-time recognition is far lower than power available in modern CPUs. The training of the HMMs in real-time environment could be more demanding due to its iterative nature. However, training does not have to be executed after each new event, and the amount of training data could be kept at reasonable amount by selecting representative subset of events.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.

VII. Conclusion

In this paper, we presented a framework for driver navigation support on the basis of learning from experience, a method for driving event recognition, and results of experiments with driving event recognition using HMMs. Learning on the basis of previous driver's experiences allows adaptation to particular environment or individual driver's style. We presented a few possible applications of driving event recognition.

We developed data acquisition hardware consisting of accelerometers, gyroscopes, and a GPS receiver. Data are collected and processed to remove noise and reduce the data volume. Vector quantization is used to translate analog signal sensors into discrete symbols from the finite set. After a number of tests, the vector quantization codebook size of 16 was selected as a compromise between the quantization error and HMM recognition performances.

For driving event recognition, we used discrete left-to-right HMMs as they better capture the dynamic characteristics of data than a general HMM. For each driving event, we selected the most appropriate number of states and trained an HMM with selected training data. We evaluated each observation sequence by all HMMs. The highest probability determined which driving event was recognized. This approach further improved the recognition rate of our system. Only 4 of 238 (1.7%) test events were wrongly recognized. The probabilities for correct event type were much higher than for incorrect event type, proving the method's reliability.

The experiment presented in this paper is, on the first glance, very similar to the one described in [7]. However, there are some significant differences. Firstly, Pentland and Liu predict the current driving intention isolated from the other events. Their system does not incorporate contextual information, which is necessary to properly make decisions in a complex dynamic environment such as driving. Our general framework, shown in Fig. 1, provides a higher level of situation awareness. We recognize driving events to build the situation model of the driver–vehicle–environment system. Such model could be used in a number of different applications as discussed above. Furthermore, in our experiments, we use sensors to measure data from a real vehicle in normal driving situations, while data used in [7] came from instrumented commands in a driving simulator. Therefore, Pentland and Liu's model is limited to driver's intentions and actions and does not take into account the actual responses from the vehicle and environment.

In presented experiments, we used a very simple and inexpensive set of sensors. We also used well-known preprocessing and prediction techniques. As a proof of the generality of the proposed recognition method, the program we developed for training and evaluation of HMMs was also successfully used for recognition of sport events from video streams [16].

Our future research will focus on the procedure for event prediction from the situation model and the development of situation assessment module to compare predicted and actual events. We will also work toward providing continuous real-time training and evaluation of models. In the long term, we plan to integrate this system with other components of a comprehensive driver's support system.

References

[1] J. A. Michon, "A critical view of driver behavior models: What do we know, what should we do?" in *Human Behavior and Traffic Safety*, L. Evans and R. Schwing, Eds. New York: Plenum, 1985, pp. 485–520.

[2] D. Mitrović, "Experiments in vehicle movements prediction," in *Proc. Australian Road Safety Conf.*, 1999, pp. 519–526.

[3] Y. Zhao, *Vehicle Location and Navigation Systems*. Boston, MA: Artech House, 1997.

[4] T. Balch and R. Arkin, "Avoiding the past: A simple but effective strategy for reactive navigation," in *Proc. IEEE Int. Conf. Robots Automation*, Atlanta, GA, 1993, pp. 678–684.

[5] O. Aycard, F. Charpillet, D. Foht and J. F. Mari, "Place learning and recognition using hidden Markov models," in *Proc. IEEE Int. Robots Syst. 1997*, Grenoble, France, pp. 1741–1746.

[6] M. C. Nechyba and Y. Xu, "Stochastic similarity for validating human control strategy models," *IEEE Trans. Robot. Autom.*, vol. 14, pp. 437–451, Jun. 1998.

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

[8] L. E. Baum and T. Petrie, "Statistical interference for probabilistic functions of finite state Markov chains," *Ann. Math. Stat.*, vol. 37, pp. 1554–1563, 1966.

[9] L. E. Baum and J. A. Egon, "An inequality with applications to statistical estimation for probabilistic functions of a Markov process and to a model for ecology," *Bull. Amer. Meteorol. Soc.*, vol. 73, pp. 360–363, 1967.

[10] L. R. Rabiner, "A tutorial on hidden Markov models and selected applications in speech recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257–286, Feb. 1989.

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

[12] L. Yang, B. K. Widjaja and R. Prasad, "Application of hidden Markov models for signature verification," *Pattern Recognit.*, vol. 28, pp. 161–169, Feb. 1995.

[13] J. Yang, Y. Xu and C. S. Chen, "Human action learning via hidden Markov model," *IEEE Trans. Syst., Man, Cybern.*, vol. 27, pp. 34–44, Jan. 1997.

[14] J. Dai, "Robust estimation of HMM parameters using fuzzy vector quantization and Parzen's window," *Pattern Recognit.*, vol. 28, pp. 53–57, Jan. 1995.

[15] J. Makhoul, S. Roucos and H. Gish, "Vector quantization in speech coding," *Proc. IEEE*, vol. 73, no. 11, pp. 1551–1567, Nov. 1985.

[16] M. Petkovic and W. Jonker, *Content-Based Video Retrieval, A Database Perspective*. Boston, MA: Kluwer, 2003.

Dejan Mitrović received the M.Sc. degree in computer science in 1993 from the University of Niš, Niš, Yugoslavia, and the Ph.D. degree in computer science in 2004 from the University of Canterbury, Christchurch, New Zealand.

He is a Senior Software Engineer at Trimble Navigation. His research interests include spatial data processing, machine learning, and driver modeling.

Authorized licensed use limited to: yuncheng jiang. Downloaded on July 12,2021 at 06:48:06 UTC from IEEE Xplore. Restrictions apply.