Expert Systems With Applications 99 (2018) 231–238

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

Evolving cloud-based system for the recognition of drivers' actions

Igor Škrjanc[a,*], Goran Andonovski[a], Agapito Ledezma[b], Oscar Sipele[b], Jose Antonio Iglesias[b], Araceli Sanchis[b]

[a] Faculty of Electrical Engineering, University of Ljubljana, Slovenia
[b] Carlos III University of Madrid, Spain

ARTICLE INFO

Article history:
Received 19 July 2017
Revised 3 November 2017
Accepted 4 November 2017
Available online 6 November 2017

Keywords:
Evolving systems
Beheviour recognition
Expert system
Cluster

ABSTRACT

This paper presents an evolving cloud-based algorithm for the recognition of drivers' actions. The general idea is to detect different manoeuvres by processing the standard signals that are usually measured in a car, such as the speed, the revolutions, the angle of the steering wheel, the position of the pedals, and others, without additional intelligent sensors. The primary goal of this investigation is to propose a concept that can be used to recognise various driver actions. All experiments are performed on a realistic car simulator. The data acquired from the simulator are pre-processed and then used in the evolving cloud-based algorithm to detect the basic elementary actions, which are then combined in a prescribed sequence to create tasks. Finally, the sequences of different tasks form the most complex action, which is called a manoeuvre. As shown in this paper, the evolving cloud-based algorithm can be very efficiently used to recognise the complex driver's action from raw signals obtained by typical car sensors.

1. Introduction

The autonomy of the intelligent systems in certain environments also assumes the understanding of actions of other agents in these environments. This is an essential characteristic of intelligent behaviour and an intelligent system. Today, in the era of big data, ubiquitous computing and cloud computing; many different systems exhibit this kind of behaviour, they can recognise the action of other agents. Activity recognition is a widespread and significant research area in ICT. For this reason, considering the focus of the research and the area of application, the research work in this field is associated with terms such as the recognition of plans (Charniak & Goldman, 1993) (Avrahami-Zilberbrand & Kaminka, 2005), user modelling (Webb, Pazzani, & Billsus, 2001), agents modelling (Steffens, 2004), opponent modelling (Ledezma, Aler, Sanchis, & Borrajo, 2009) (Iglesias, Ledezma, & Sanchis, 2009), intention recognition (Sukthankar, Geib, Hai Bui, Pynadath, & Goldman, 2014), behavioural recognition (Candamo, Shreve, Goldgof, Sapper, & Kasturi, 2010), (Iglesias, Angelov, Ledezma, & Sanchis, 2010), etc. In addition, different applications have been developed in this area, such as those related to people care and health, intelligent personal assistants, intelligent user interfaces, electronic commerce and driver modelling (Chen, Hoey, Nugent, Cook, & Yu, 2012; Ke et al., 2013) which is the focus of this work.

Within the area of driver modelling, the recognition of driving manoeuvres raises multiple applications, such as the identification of deficient states of the driver, such as fatigue, distraction, etc., which can cause unsafe driving, by detecting a deviation from his usual way of driving; the creation of cognitive models that allow the reproduction of the way of executing manoeuvres of a human driver in autonomous vehicles; the determination of possible driving styles (normal, aggressive, etc.) that a driver can adopt in certain situations.

Nowadays, the so-called Advanced Driver Assistance System (ADAS) (Bengler et al., 2014) are available in a wide range of vehicles. These systems, whose main goal is to make driving easier and safer, provide a large variety of functionalities such as lane departure warning system (environments-based ADAS) (Cacciabue, 2007) or driver drowsiness detection (Driver-centred ADAS). In particular, an essential aspect of the driver-based ADAS is to detect what the driver is doing or what she/he is going to do. This detection can be done by direct observation, i.e. head-viewing cameras for eye detection (Liu, Xu, & Fujimura, 2002; Sigari, Fathy, & Soryani, 2013; Zhang, Cheng, Feng, & Zhang, 2008) or for yawning detection (Abtahi, Hariri, & Shirmohammadi, 2011; Tiesheng Wang, Pengfei Shi, Wang, & Shi, 2005) and by indirect observation, i.e. sensors (smartphone) in the car (Castignani, Derrmann, Frank, & Engel, 2017; Cervantes-Villanueva, Carrillo-Zapata, Terroso-Saenz, Valdes-Vela, & Skarmeta, 2016). Once the drivers

* Corresponding author.
E-mail addresses: igor.skrjanc@fe.uni-lj.si (I. Škrjanc), goran.andonovski@fe.uni-lj.si (G. Andonovski), ledezma@inf.uc3m.es (A. Ledezma), bsipele@inf.uc3m.es (O. Sipele), jiglesia@inf.uc3m.es (J.A. Iglesias), masm@inf.uc3m.es (A. Sanchis).

https://doi.org/10.1016/j.eswa.2017.11.008
0957-4174/

behaviour is detected, intelligent systems can work as expert co-pilots (Zamora, Sipele, Ledezma, & Sanchis, 2017) and facilitate driving performance and make it easier.

In our approach, a well-established technique for the approximation of complex and non-linear systems is used. Our approach originates from fuzzy modelling, which is one of the most popular methods. Constructing a Takagi–Sugeno model (Takagi & Sugeno, 1985) requires identifying the membership functions and the local model's parameters. Furthermore, if the system is time-variant, the system should have the ability to describe the different behaviours by evolving the model structure and identifying the parameters on-line. In recent years, some evolving identification techniques have been proposed relying on fuzzy logic (eTS by Angelov and Filev (2004), exTS by Memon, Angelov, and Ahmed (2006), FLEXFIS by Lughofer and Klement (2005), switching eTS by Kalhor, Araabi, and Lucas (2013), etc.), that not adapt only the parameters of the model but also the model structure is changed according to the data received.

The simplest form of fuzzy rule-based (FRB) systems was proposed by Angelov and Yager (2011). This new FRB system, named AnYa, uses a non-parametric, vectorized antecedent part. It is based on data clouds, which are sets of previous data samples close to each other, while the membership functions are calculated using the relative data density of the current data with the existing cloud. The data clouds, as the traditional data clusters, are used for partitioning the problem space to detect different operational (non-linear) conditions. In comparison to the data clusters, the data clouds do not require an explicit definition of the membership function (Gaussian, Triangular, etc.). Moreover, the data clouds do not have or require boundaries, therefore they do not have specific shape. The AnYa method can evolve the structure by adding new data clouds. In recent years, the AnYa FRB system was used to solve different control problems (Andonovski, Angelov, Blažič, & Škrjanc, 2016; Angelov, Škrjanc, & Blažič, 2013; Costa, Škrjanc, Blažič, & Angelov, 2013; Škrjanc, Blažič, & Angelov, 2014) and also for identification of different processes (Ali, Hutchison, Angelov, & Smith, 2012; Blažič, Dovžan, & Škrjanc, 2014; Rosa, Gomide, Dovžan, & Škrjanc, 2014).

In our approach, the evolving cloud-based algorithm is used to recognise the driver' action from the signals, which are usually measured in the car. These data are pre-processed and used to form clouds that are typical for different basic elementary actions, which are called atomic actions (*AA*). These atomic actions form more complex prescribed sequences called tasks (*T*). The sequences of different tasks can form a more complex structure, which is called manoeuvre (*M*). As shown in this paper, the evolving cloud-based algorithm can be very efficiently used to recognise a driver's very complex action with the use of the simplest sensors. This paper is organized as follows: after the introduction, the problem description of driver modelling is presented. In the next section, the proposed solution in the form of evolving cloud-based algorithm is given, the pre-processing of the signals, the atomic actions, the tasks and the manoeuvres detection is presented in detail. The experimental results in the learning and in the validation phase are given for the signals obtained on the car simulator. At the end, some conclusions are made.

2. Problem description - driver modelling

Driver behaviour modelling (DBM) is a research problem that has been extensively studied in many different research works Torkkola, Venkatesan, and Liu (2005), Sathyanarayana, Boyraz, and Hansen (2008), Miyajima et al. (2007), Shi et al. (2015) and Nakano, Okuda, Suzuki, Inagaki, and Hayakawa (2009). In Torkkola et al. (2008), a machine learning approach is employed to detect and classify a wide range of driving manoeuvres, to perform a large scale automotive sensor selection and to detect driver inattention by using sensors available in the current vehicle fleet.

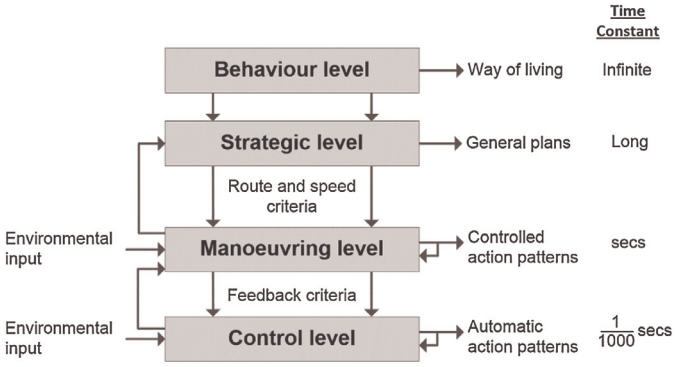

Fig. 1. Different hierarchical levels of the car.

The main goal of DBM is to improve the transport safety and the driving experience. In this sense, there are some surveys that review the literature about this issue. In a recent survey Abuali and Abou-zeid (2016) provide an overview of advances of capabilities and applications, and services of DBM emphasizing research challenges and key future directions. Doshi and Trivedi (2011) survey the field of driver behaviour and intent prediction, with a specific focus on tactical manoeuvres, as opposed to operational or strategic manoeuvres. In Wang, Xi, and Chen (2014) works covering driver skill and different approaches to driver models are presented.

The components and stages involved in driver behaviour modelling are very diverse. In this sense, DBM can be classified as reactive or predictive models taking into account if they classify the observed behaviour or driving manoeuvre in real time or after the action has been conducted. Typically, the predictive models are more difficult to develop than the reactive models. In this research, we present a reactive model. Other aspect in DBM is the hierarchical control model which can be categorized as operational, tactical, and strategic.

In Michon (1985) a hierarchical structure of the road user task is proposed. In that structure, performance is structured at three levels of skills and control which are comparatively loosely coupled: strategic (planning), manoeuvring (tactical) and control (operational). The external outputs of these 3 levels are respectively: general plans, controlled action patterns and automatic action patters. In Panou, Bekiaris, and Papakostopoulos (2007) an adaptation of that hierarchical model is defined and one more level (behaviour level) is included. This level refers to *personal preconditions* and *characteristics*, and has the highest priority because such dispositions heavily influence driving decisions at lower levels. Fig. 1 shows the hierarchical structure of the driving levels.

Other very important aspect are the inputs used in the DBM task. Thus, the vehicle data used can be from the Controller Area Network (CAN), sensors (such radars, lane position sensors, GPS, accelerometers or gyroscopes), or cameras. This research is focused only in the data from CAN.

Due to the complexity of different human activities and the different levels of abstraction that can be used to represent them, it is necessary to establish a taxonomy of activities that allows approaching the task of recognition. Different approaches for creating this taxonomy of activities can be considered (Chaaraoui, Climent-Perez, & Florez-Revuelta, 2012; Moeslund, Hilton, & Kruger, 2006). However, most of these approaches are based on the same idea: defining the taxonomy of actions based on duration, semantic degree and complexity of actions. In this work, an ad hoc taxonomy that consists of different hierarchical levels in a pyramidal form has

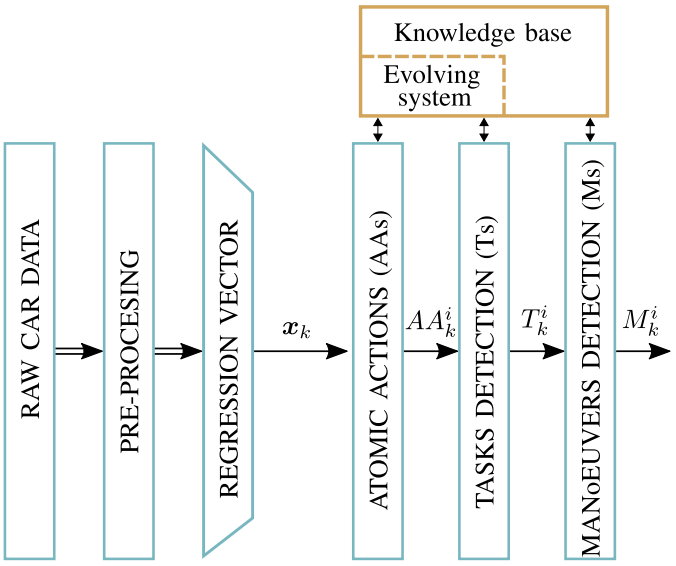

Fig. 2. Data acquisition system and manoeuvres detection.

been used. Each higher level indicates a higher level of complexity, a higher semantic level and a longer time of activity. In addition, activities at higher levels are made up of activities from lower levels.

In addition, modelling driver behaviour can be used in many different applications such as driver training, energy efficiency, crowd-sourced sensing for read conditions and, specially, Driver Assistance Systems (ADAS). The work presented in this research is very related with the object of ADASs, to forecast the behaviour of a vehicle in real time and compensate for dangerous circumstances or events.

Finally, the modelling objectives are also important in the development of these systems. In this research, the objective of the presented research work is to model specific manoeuvres in order to model general driver characterization.

The main idea of our research of driver's behaviour modelling is based on the detection of the different tasks involved in the activity of driving a vehicle and how they interrelate with each other. In this sense, different levels can be distinguished: the lowest level is related to the basic actions that can be performed by the driver, such as stepping on the pedals, turning the wheel, etc. These basic actions are then linked together in a higher level; tasks and manoeuvres that are linked to those decisions taken by the driver during a route (Levermore, Ordys, & Deng, 2014).The proposed taxonomy consists of four types of activities, from the top to the bottom of the hierarchy: manoeuvre, such as for example overtaking, which is composed of tasks, atomic actions or events, task, which is composed of atomic actions or events, such as starting the engine and starting to drive, and atomic action, such as moving up a gear, which are composed of one or more events.

3. Proposed solution

The proposed idea of the evolving cloud-based system to recognise the driver actions is shown as schematic diagram in Fig. 2. In the first block, the acquisition of the raw signals from the simulator is realized. These signals are normalized and pre-processed to obtain mainly the discretized form of them. The discretized signals are then collected to form the regression vector, which is then used in the first evolving cloud-based algorithm to detect the possible atomic actions. In the second block, the tasks are detected and in the third the manoeuvres.

All these parts of the whole system will be discussed in detail in the next subsections.

3.1. Pre-processing

The pre-processing of the raw signals, obtained from the car simulator presented in Vernaza, Ledezma, and Sanchis (2014), is one of the most important parts of the whole algorithm. It is important to eliminate the redundant information. In our case, the stored signals are the following: *speed* (v), *revolutions per minute* (r), *position of the steering wheel* (w), *gas pedal* (g), *brake pedal* (b), *clutch pedal* (p) and *gear switch* (s). These are all the signals which are usually measured in a car nowadays.

The raw signals are pre-processed to obtain the information in the discrete. The signals *speed* (v), *revolutions* (r) and *position of steering wheel* (w) are pre-processed to extract the information about the trend. The speed v is pre-processed as follows:

$$\hat{v}_k = \begin{cases} 1, & \Delta v_k > 0 \\ 0, & \Delta v_k = 0 \\ -1, & \Delta v_k < 0 \end{cases} \tag{1}$$

where subscript k denotes the time instant and $\Delta v_k = v_k - v_{k-1}$. The same pre-processing procedure is used for continuous variables r, and w. The obtained discretized variables are then $\hat{v}$, $\hat{r}$, and $\hat{w}$.

The signals *gas pedal* (g) and *brake pedal* (b) are pre-processed to extract the information about the trend. The signal *gas pedal* is pre-processed using the following function:

$$\hat{g}_k = \begin{cases} 1, & \Delta g_k > 0 \quad or \quad g_k = 1 \\ 0, & \Delta g_k = 0 \\ -1, & \Delta g_k < 0 \end{cases} \tag{2}$$

where $\Delta g_k = g_k - g_{k-1}$. The first row in (2) means that in case the original signal has value 1 ($g_k = 1$), then the discretized signal has also the value $\hat{g}_k = 1$. However, in this case we do not care about the gradient of the signal, but we take in the consideration the maximal value of the signal. The same procedure is also used for the *brake pedal*. The discretized variables are then $\hat{g}$ and $\hat{b}$.

The *clutch pedal* (p) is pre-processed in a way to obtain the information about whether the pedal is pressed or not. The discrete variable $\hat{p}$ is obtained using the following function:

$$\hat{p}_k = \begin{cases} 1, & p_k > 0 \\ 0, & p_k = 0 \end{cases} \tag{3}$$

The last signal is *gear switch* (s) and it is pre-processed using the following procedure. In general the variable s is already discrete ($s = \{1, 2, 3, 4, 5, N, R\}$), where N and P define neutral and reverse, respectively. However, an important situation is when the value changes from lower to higher gear ($\hat{s} = 1$) and when the value changes from higher to lower gear ($\hat{s} = -1$):

$$\hat{s}_k = \begin{cases} 1, & \Delta s_k > 0 \\ -1, & \Delta s_k < 0 \end{cases} \tag{4}$$

3.2. Regression vector

For a certain atomic action the pre-processed signals in discrete form represent an unique combination of values; therefore, the regression vector of these discrete values is formed as follows to define a space of dimension $n = 7$, Eq. (5).

$$\boldsymbol{x}_k = \left[\hat{v}_k,\ \hat{r}_k,\ \hat{w}_k,\ \hat{g}_k,\ \hat{b}_k,\ \hat{p}_k,\ \hat{s}_k\right]^T \tag{5}$$

Regarding the upper pre-processing of the signals, each atomic action represents one of the $3^5 \cdot 2^2$ combinations of possible discrete signal values. These regression vector is then used in the evolving algorithm to detect the actions.

3.3. Evolving system for atomic actions detection

The general form of the evolving algorithm is based on the AnYa rule-based system (Angelov & Yager, 2011). A particular atomic action, AA^i, is represented as a point, or a region of points or cloud, in the data space. With the first data point $\boldsymbol{x}_k$ received, the first atomic action or cloud is defined, AA^0. Further, using the recursive density measure, the next data points $\boldsymbol{x}_k$ are a part of existing clouds or atomic actions or they do not belong to the existing cloud and therefore a new cloud or AA should be added.

The recursive density is calculated as follows in Eq. (6)

$$\gamma_k^{i_A} = \frac{1}{1 + \|\boldsymbol{x}_k - \boldsymbol{\mu}_k^{i_A}\|^2 + \sigma_k^{i_A} - \|\boldsymbol{\mu}_k^{i_A}\|^2} \quad i_A = 1, \ldots, c_A \tag{6}$$

where c_A is the number of known atomic actions, $\boldsymbol{\mu}_k^{i_A}$ and $\sigma_k^{i_A}$ denote mean value vector and mean-square length of the data of ith AA, respectively. Both $\boldsymbol{\mu}_k^{i_A}$ and $\sigma_k^{i_A}$, are calculated recursively using Eqs. (7) and (8).

$$\boldsymbol{\mu}_k^{i_A} = \frac{N^{i_A} - 1}{N^{i_A}} \boldsymbol{\mu}_{k-1}^{i_A} + \frac{1}{N^{i_A}} \boldsymbol{x}_k \tag{7}$$

$$\sigma_k^{i_A} = \frac{N^{i_A} - 1}{N^{i_A}} \sigma_{k-1}^{i_A} + \frac{1}{N^{i_A}} \|\boldsymbol{x}_k\|^2 \tag{8}$$

Therefore, each atomic action has tree features, the mean value $\boldsymbol{\mu}^{i_A}$, the mean-square length σ^{i_A} and the number of points N^{i_A}.

The algorithm starts without any knowledge about the process, and the first data point received is declared as a centre of the first cloud or atomic action. For the next data points received, the local densities are calculated (6). If the maximal value of the calculated local densities $\max(\gamma_k^{i_A})$ between the current data $\boldsymbol{x}_k$ and all existing clouds is lower than the threshold γ_{max}, then a new cloud is defined. Otherwise, the current data is associated to the cloud or atomic action with maximal local density. The evolving procedure for detecting new cloud or atomic actions is described in the pseudo Algorithm 1. In this procedure the parameter γ_{max} influences the number of detected atomic actions. As it is shown in Figs. 5–8 the parameter γ_{max} is set to detect all the atomic action as they are declared by the expert knowledge.

Algorithm 1 Pseudo algorithm for detecting new Atomic Actions AA.

1: Initialize (Evolving parameter): $\gamma_{max} = 0.85$.
2: **repeat** ▷ *k is running index*
3: **Input**: Measurement from the car
4: Pre-processing: (1), (2) and (3)
5: Regression vector: $\boldsymbol{x}_k$.
6: Compute: $\gamma_k^{i_A}, \quad i_A = 1, \ldots, c_A$.
7: **if** k==1 **then**
8: Init.: $c_A \leftarrow 1$
9: Init.: $\mu_k^{c_A} \leftarrow \boldsymbol{x}_k$, $\sigma_k^{c_A} \leftarrow \|\boldsymbol{x}_k\|^2$, $N_k^{c_A} \leftarrow 1$.
10: Define: $AA_k^{c_A} \in \{\mu_k^{c_A}, \sigma_k^{c_A}, N_k^{c_A}\}$
11: **else**
12: **if** $\max_{i_A}(\gamma_k^{i_A}) < \gamma_{max}$ **then** ▷ *Evolving law*
13: *New atomic action is detected.*
14: Increment: c_A.
15: Init.: $\mu_k^{c_A} \leftarrow \boldsymbol{x}_k$, $\sigma_k^{c_A} \leftarrow \|\boldsymbol{x}_k\|^2$, $N_k^{c_A} \leftarrow 1$.
16: Define: $AA^{c_A} \in \{\mu_k^{c_A}, \sigma_k^{c_A}, N_k^{c_A}\}$.
17: **else**
18: Associate sample $\mathbf{x}_k$ with AA $(max_{i_A} \gamma_k^{i_A})$
19: Update $\mu_k^{i_A}$, $\sigma_k^{i_A}$ and $N_k^{i_A}$ for that AA
20: **end if**
21: **end if**
22: **Output**: Current atomic action's centre $(\mu_k^{i_A})$.
23: **until** End of data stream.

3.4. Task detection

A certain set of atomic actions centres forms a task. Therefore, a particular group of atomic actions defines the task as higher level of abstraction. In this case the order of atomic action is not important. The mechanism of tasks detection is highly similar to the detection of atomic actions given in Algorithm 1 and is presented in the pseudo-code Algorithm 2. Both algorithms are practically the same, with a few differences. For example, the input data into Algorithm 2 is the centre of the current active atomic action or cloud $\mu_k^{i_A}$, with the maximal local density. Next, the adding mechanism (see line 10 in Algorithm 2) is extended with outliers protection mechanism (at least n_{add} samples has to pass from the last added task k_{add}). Therefore, the evolving mechanism for detecting tasks contains two parameters, γ_{max} and n_{add}. The value of the density threshold γ_{max} should be set in the range $\gamma_{max} \in [0\,1]$. Values closer to 1 generate a lof of tasks, while values closer to 0 generate just few tasks. The second parameter n_{add} serves as protective mechanism and helps to the newly added tasks to acquire enough information about the current state of the system. To summarize, Algorithm 2 receives atomic action's centre as input and returns a task's index as output.

Algorithm 2 Pseudo algorithm for detecting Tasks Ts.

1: Initialize (Evolving parameters): $\gamma_{max} = 0.25$, $n_{add} = 10$.
2: **repeat** ▷ *k is running index*
3: **Input**: Current AA centre $(\mu_k^{i_A})$.
4: Compute: $\gamma_k^{i_T}, \quad i_T = 1, \ldots, c_T$.
5: **if** k==1 **then**
6: Init.: $c_T \leftarrow 1$
7: Init.: $\mu_k^{c_T} \leftarrow \mu_k^{i_A}$, $\sigma_k^{c_T} \leftarrow \|\mu_k^{i_A}\|^2$, $N_k^{c_T} \leftarrow 1$.
8: Define: $T_k^{c_T} \in (\mu_k^{c_T}, \sigma_k^{c_T}, N_k^{c_T})$
9: **else**
10: **if** $(\max_{i_T}(\gamma_k^{i_T}) < \gamma_{max}) \wedge (k > k_{add} + n_{add})$ **then**
11: *New Task is detected.*
12: Increment: c_T,
13: Init.: $\mu_k^{c_T} \leftarrow \boldsymbol{x}_k$, $\sigma_k^{c_T} \leftarrow \|\boldsymbol{x}_k\|^2$, $N_k^{c_T} \leftarrow 1$.
14: Define: $T^{c_T} \in (\mu_k^{c_T}, \sigma_k^{c_T}, N_k^{c_T})$
15: **else**
16: Associate current AA centre $(\mu_k^{i_A})$ with T $(\max_{i_T}(\gamma_k^{i_T}))$
17: Update $\mu_k^{i_T}$, $\sigma_k^{i_T}$ and $N_k^{i_T}$ for that $T_s^{i_T}$
18: **end if** ▷ *End of the evolving law for Tasks*
19: **end if**
20: **Output**: Current Task's index (i_T).
21: **until** End of data stream.

3.5. Manoeuvre detection

The manoeuvres are defined as a specific sequence of different tasks. For example, the ith manoeuvre M^i is defined as a vector of tasks $M^i = \left[T^1, T^3, T^2, T^4, T^3\right]^T$. The pseudo Algorithm 3 presents the procedure of detecting the current task sequence, where δ_T is the holding parameter. This parameter can be understand as low-pass filter, therefore, the idea behind this parameter is to filter out all fast changes between different tasks. In Algorithm 3, $last_{i_T}$ is index of the last task in the sequence, κ is counting index, i_T is index of the current task, M_k is the manoeuvre vector that contains current sequence of detected tasks' indexes. In the learning phase the output of the Algorithm 3 is the sequence M^{i_M}

Algorithm 3 Pseudo algorithm for detecting Manoeuvres M_s.

Initialize parameter: $\delta_T = 7$.
Set values: $last_{i_T} = -1$, $\kappa = 0$.
Input: Current Task's index (i_T).
repeat ▷ *stream of Tasks' indexes* i_T
  if $i_T \neq last_{i_T}$ **then**
    Increment: κ
    if $\kappa > \delta_T$ **then**
      $M_k \leftarrow \left[M_{k-1} \mid i_T\right]$
      $\kappa \leftarrow 0$
      $last_{i_T} \leftarrow i_T$
    else
      $M_k \leftarrow \left[M_{k-1}\right]$
    end if
  else
    $\kappa \leftarrow 0$
    $M_k \leftarrow \left[M_{k-1}\right]$
  end if
  Output: Current task sequence M_k.
until End of data stream.

($i_M = 1, \ldots c_M$) which is stored to the knowledge base. In the validation phase the output of the Algorithm 3 is the current manoeuvre sequence M_k which need to be compared with the knowledge base of learned manoeuvres' sequences from the learning phase. The comparison of the current sequence M_k and the learned manoeuvres' sequences is realized by using the Hamming distance. The Hamming distance $d_H(x, y)$ between two vectors $x, y \in F(n)$ is the number of coefficients in which they differ. The current sequence M_k is classified as a known manoeuvre if the following inequality is fulfilled:

$$\left(\min_{i_M=1,\ldots,c_M} \frac{d_H(M_k, M^{i_M})}{len(M_k)}\right) < m_f \tag{9}$$

where $m_f = 0.35$ is user defined threshold that defines the maximal ratio of difference and should be defined by trial and error.

4. Experimental results

All the data, from, the data of events, atomic actions and the corresponding manoeuvres, that are used in this experimental section, have been obtained through a driving simulation system. This system is based on the advanced systems of STISIM Drive®, (Systems Technology, 2016), and it incorporates driving devices that simulate the behaviour of those installed in a real vehicle.

Considering the vehicle controls, the simulation system incorporates a steering wheel that simulates the power steering system and pedals like those of a real vehicle as can be seen in Fig. 3. This provides a realistic simulation experience during the data collection.

The methodology that has been used in the process of collecting data on atomic actions and manoeuvres, consists of the following phases:

1. Designing a simulated driving scenario. The scenario must be designed to use the elements such as intersections, traffic lights and slow vehicles, that allows the visual identification of a set of manoeuvres M. In addition, the execution of the same manoeuvre is mutually exclusive. Fig. 4 shows a sequence of images in which an overtaking manoeuvre is executed by the driver in the designed driving scenario.
2. Data collection. By using the designed scenario, the data related to driving controls (i.e. the raw signals) are collected in real time. The videos of the execution of the driving are also stored. As result of the scenario execution, a data file is obtained. In

Fig. 3. Simulator system.

Fig. 4. Overtaking manoeuvre.

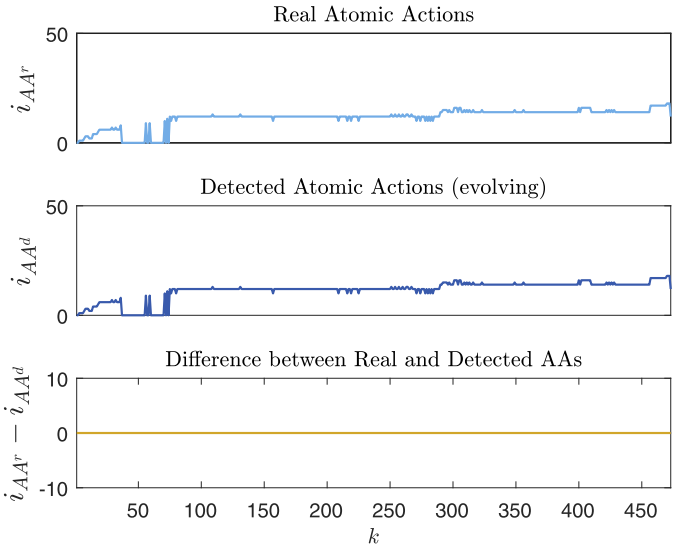

Fig. 5. *Learning phase.* (Overtaking manoeuvre). Operator defined *AAs* (upper plot), detected *AA* using the evolving mechanism (middle plot) and difference between them (bottom plot).

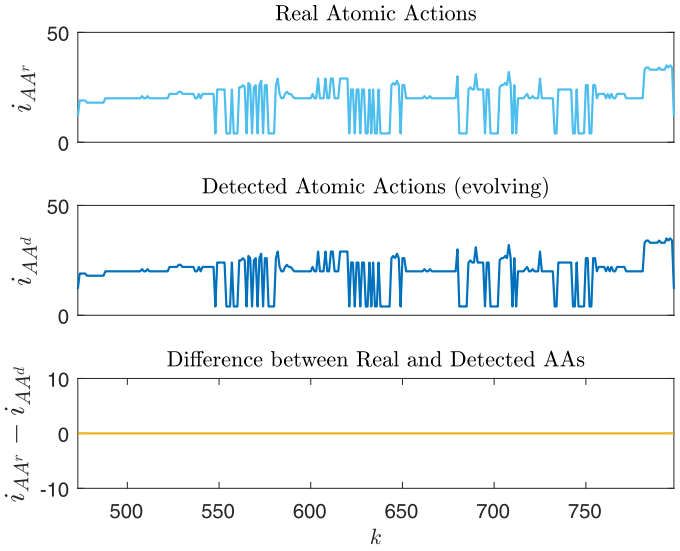

Fig. 6. *Learning phase.* (Stopping manoeuvre). Operator defined *AA* (upper plot), detected *AA* using the evolving mechanism (middle plot) and difference between them (bottom plot).

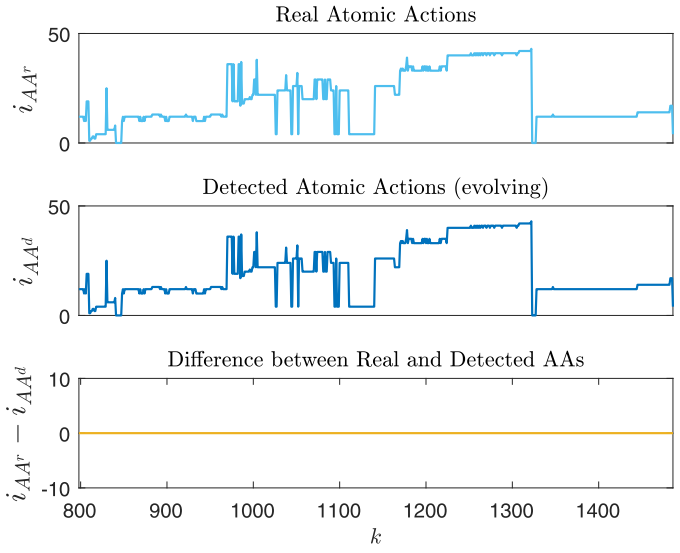

Fig. 7. *Learning phase.* (Manoeuvre stopping at traffic light). Operator defined *AA* (upper plot), detected *AA* using the evolving mechanism (middle plot) and difference between them (bottom plot).

this file, each line (i.e. instance) is identified with a time stamp and containing the set w, where $st \in \{S\}$ is each of the signals generated by the driving controls at that time instant.

3. Visual identification of manoeuvres. Using the recorded videos of each simulation, a ground truth task has been performed to identify the time gaps in which the driver performs a specific manoeuvre, $mi = 0, 1 \mid mi \in M$.
4. Fusion of raw data and manoeuvre tags. The instances, that belong to the time gaps of the manoeuvre set identified, are searched in the file. As result of this process, a new file is obtained. Each instance in the file is composed of the set of signals obtained in the Phase 2, and the set of tags M obtained in the Phase 3.

In this section four different manoeuvres were studied: overtaking, stopping, stopping at the traffic light and keeping the safety distance. For the learning phase the original data from the simulator were used and for the validation phase the same data were

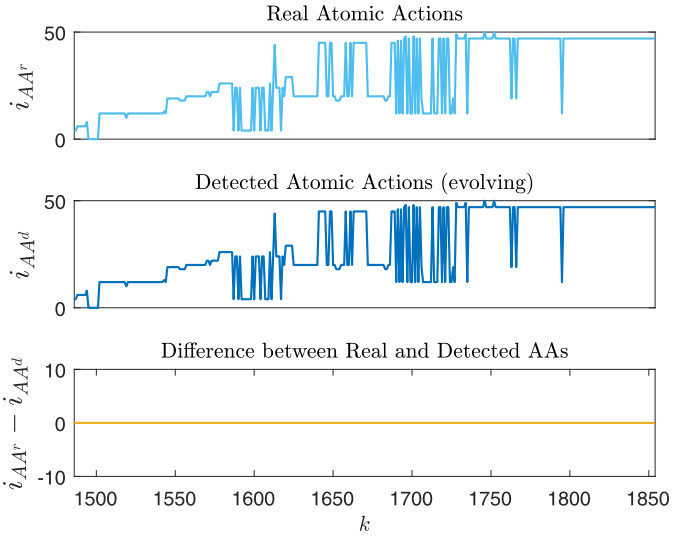

Fig. 8. *Learning phase.* (Manoeuvre keeping safety distance). Operator defined *AA* (upper plot), detected *AA* using the evolving mechanism (middle plot) and difference between them (bottom plot).

modified by adding some artificially generated values. The modification of the original data was done with mainly extending and repeating each data sample for a random number of times to extend a certain action in time.

4.1. Learning phase

In the learning phase of the whole system the prototypes for manoeuvres should be defined. This is done through atomic actions and tasks detection. In this section the results of learning phase will be shown.

4.1.1. Atomic action detection

The raw data from the simulator were pre-processed as proposed in Eqs. (1)–(4). Next, the regression vector is formed as given in Eq. (5). Finally, by using Algorithm 1 the atomic actions are detected in an on-line manner, starting with no clouds and adding the new when the observed sample does not belong to the existing clouds (atomic actions) according to the predefined density criteria.

In Figs. 5–8 the real (predefined-by-the-operator) and the automatically detected atomic actions are presented. Each figure represents the data of the specific manoeuvre. In these figures, on the Y axis the indexes i_{AA} of atomic actions are shown and not their centres, because the of the dimensionality. With the first data received at the beginning of Fig. 5, the first atomic action is defined. The knowledge accumulated, in the form of detected clouds, from the overtaking data in Fig. 5 is used for the next data set, and so on, to accumulate all possible atomic actions. The proposed evolving cloud-based system is capable of detecting all possible atomic actions, without any error (difference between predefined and detected atomic actions). The total number of all detected atomic actions is $c_A = 51$.

4.1.2. Task detection

Once the atomic actions are detected, the next step is to detect tasks. Algorithms 1 and 2 are running simultaneously. Similar atomic actions, i.e. these which are close to each other, are grouped together as described in Section 3.4 and the tasks are detected using the evolving mechanism described in Algorithm 2). In Fig. 9 the detected atomic actions and tasks are presented for all four manoeuvres. The manoeuvres are: overtaking, stopping, stopping at

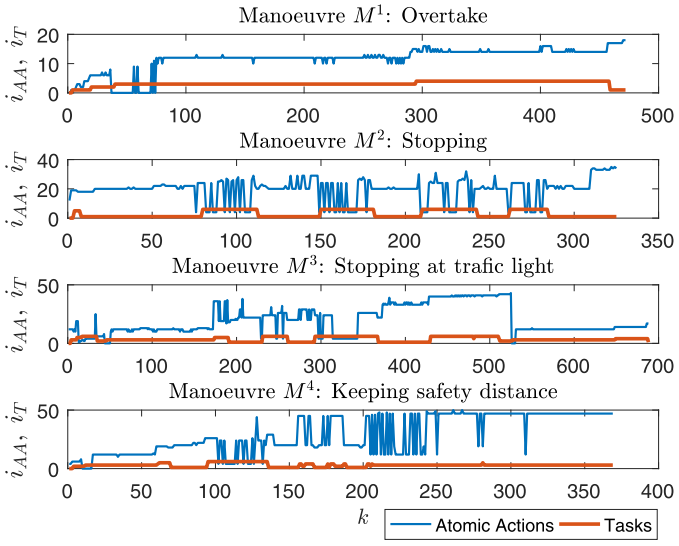

Fig. 9. *Learning phase.* Detected tasks' indexes (red line) for all four manoeuvres. Blue line notes the atomic actions' indexes. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

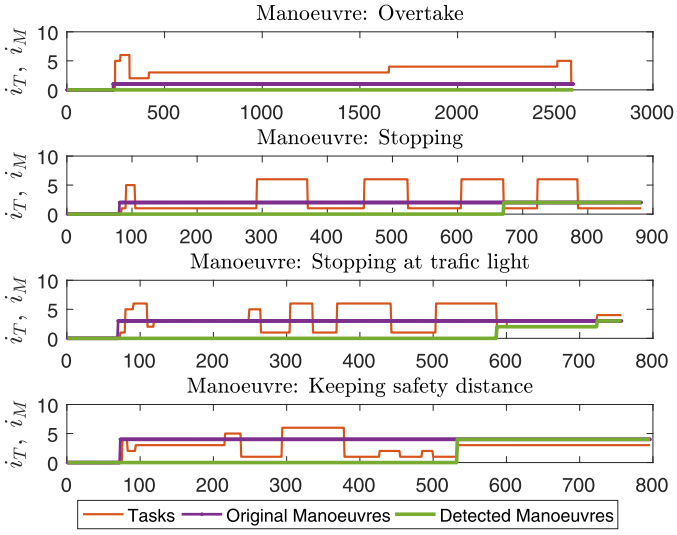

Fig. 10. *Validation phase.* Detected Tasks' sequences (red lines) and corresponding manoeuvres detected (green lines). The purple line denotes the original/known manoeuvres index. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

traffic light and keeping safety distance. The total number of tasks detected is $c_T = 6$.

4.1.3. Manoeuvre detection

Finally, using Algorithm 3, the sequences of tasks are detected for each data file separately. In fact, the algorithm's input are task's indexes ($i_T = 1, \ldots, c_T$) while the output is the sequence of these tasks. For each manoeuvre (data file) the following sequences of tasks are detected:

$$M^1 = \left[T^1\ T^2\ T^3\ T^4\ T^5\right]$$
$$M^2 = \left[T^6\ T^1\ T^6\ T^1\ T^6\ T^1\ T^6\ T^1\right]$$
$$M^3 = \left[T^5\ T^6\ T^2\ T^3\ T^5\ T^1\ T^6\ T^1\ T^6\ T^1\ T^6\ T^2\ T^3\ T^4\right]$$
$$M^4 = \left[T^2\ T^3\ T^5\ T^1\ T^6\ T^1\ T^4\ T^1\ T^2\ T^1\ T^3\right]$$

where M^1 denotes overtaking, M^2 stopping, M^3 stopping at traffic light and M^4 keeping safety distance manoeuvre.

To summarize the learning phase of the proposed procedure, we detected 51atomic actions, 10 tasks and 4 manoeuvres. These data are stored in the knowledge base and it is further used for validation purposes. The detected prototypes are then used in the validation phase to find the similar sequences in the validation data sets.

4.2. Validation phase

As explained above, for the validation phase the modified data were used. The modification of the original data was done by repeating each data point for a randomly chosen number of samples (between 0 and 10). During the learning phase we have acquired knowledge (AAs and Ts centres, and Ms sequences) which is used in the validation phase. Using Algorithms 1 and 2 the atomic actions and tasks were detected, respectively. During the validation phase the evolving mechanism is frozen and Algorithms 1 and 2 serve just as classification tool (without adding new data clouds). It should be mentioned that all the algorithms are performed sequentially in each step. Simultaneously, using the criteria (9), the currently detected task sequence (Algorithm 3) is compared with the known manoeuvres from the knowledge base (M^1, M^2, M^3, M^4).

Each sub-plot in Fig. 10 represents one particular manoeuvre. Each manoeuvre sequence from M^1 to M^4 is indexed (and plotted) with the value from 1 to 4, respectively. The red lines present the detected tasks, the known manoeuvres are plotted with a purple solid line while the detected manoeuvres with a green line. At the moment when the green line rises from zero to some value the criteria (9) is fulfilled or in other words a manoeuvre is detected. In Fig. 10, we expected to detect manoeuvre M^1 in the first sub-plot, M^2 in the second sub-plot, etc. With the exception of the first sub-plot in Fig. 10 where no manoeuvre was detected in other tree sub-plots (from second to fourth) the manoeuvres were successfully detected.

Detecting manoeuvres is directly influenced by the parameter $m_f = 0.35$ (sensitivity parameter). If we increase this value (increase the sensitivity) we can also detect the first manoeuvre from the first data file. However, in this case, we detect too many false manoeuvres in the other three data files.

5. Conclusion

In this paper, a new concept of driver's actions recognition system is presented. It is based on an evolving cloud-based algorithm. The proposed system is capable of recognising different manoeuvres by processing the standard signals which are usually measured in the car, and no new sensors are needed. This, of course, makes the implementation much simpler and inexpensive. The main idea of a concept is proven and fulfilled by all means. The concept yields very promising results and can be used very efficiently to recognise the complex drives action by using the simplest sensors in the car. Moreover, the evolving nature (self-learning) of the proposed procedure presents a big advantage for solving detection problems. This property allows us to detect new stages and to extend the knowledge base.

Detecting what the user is doing is only a part of the task that should be done by an expert co-pilot. For this reason, one of the proposed future works is related with the use of fuzzy rule-based (FRB) systems to develop driver's evolving models in order to suit the driving style of the driver. On the other hand, the integration of *new* data from different sensors, such as cameras or proximity sensors, can be very useful. It is also important to consider that the driver can perform a wide variety of tasks concurrently and each

driver can execute those tasks following a different sequence of actions. The use of wearable sensors, which are widely used nowadays, should facilitate driver data collection and therefore, driver behaviour modelling using FRB systems.

Acknowledgements

This work has been supported by the Program Chair of Excellence of Universidad Carlos III de Madrid and Bank of Santander and the Spanish Ministry of Economy, Industry and Competitiveness, projects TRA2015-63708-R and TRA2016-78886-C3-1-R.

References

Abtahi, S., Hariri, B., & Shirmohammadi, S. (2011). Driver drowsiness monitoring based on yawning detection. In *Instrumentation and measurement technology conference (I2MTC)* (pp. 7–10).

Abuali, N., & Abou-zeid, H. (2016). Driver behavior modeling : Developments and future directions. *International Journal of Vehicular Technology, 2016*, 1–12.

Ali, A., Hutchison, D., Angelov, P., & Smith, P. (2012). Adaptive resilience for computer networks: Using online fuzzy learning. 4th international congress on ultra modern telecommunications and control systems and workshops (ICUMT), (pp. 772–778).

Andonovski, G., Angelov, P., Blažič, S., & Škrjanc, I. (2016). A practical implementation of robust evolving cloud-based controller with normalized data space for heat-exchanger plant. *Applied Soft Computing, 48*, 29–38.

Angelov, P., Škrjanc, I., & Blažič, S. (2013). Robust evolving cloud-based controller for a hydraulic plant. In *2013 IEEE conference on evolving and adaptive intelligent systems (EAIS)* (pp. 1–8).

Angelov, P., & Yager, R. (2011). Simplified fuzzy rule-based systems using non-parametric antecedents and relative data density. In *Symposium series on computational intelligence (IEEE SSCI 2011) - IEEE workshop on evolving and adaptive intelligent systems (eais 2011)* (pp. 62–69).

Angelov, P. P., & Filev, D. P. (2004). An approach to online identification of Takagi–Sugeno fuzzy models. *IEEE Transactions on Systems, Man, and Cybernetics, 34*(1), 484–498.

Avrahami-Zilberbrand, D., & Kaminka, G. A. (2005). Fast and complete symbolic plan recognition. In *International joint conference on artificial intelligence (IJCAI)* (pp. 653–658). Edinburgh, Scotland: Morgan Kaufmann Publishers Inc.

Bengler, K., Dietmayer, K., Farber, B., Maurer, M., Stiller, C., & Winner, H. (2014). Three decades of driver assistance systems: Review and future perspectives. *IEEE Intelligent Transportation Systems Magazine, 6*(4), 6–22.

Blažič, S., Dovžan, D., & Škrjanc, I. (2014). Cloud-based identification of an evolving system with supervisory mechanisms. In *2014 IEEE international symposium on intelligent control (ISIC),Antibes, France* (pp. 1906–1911).

Cacciabue, C. (2007). *Modelling driver behaviourin automotive environments*. Springer-Verlag London Limited.

Candamo, J., Shreve, M., Goldgof, D. B., Sapper, D. B., & Kasturi, R. (2010). Understanding transit scenes: A survey on human behavior-recognition algorithms. *IEEE Transactions on Intelligent Transportation Systems, 11*(1), 206–224.

Castignani, G., Derrmann, T., Frank, R., & Engel, T. (2017). Smartphone-based adaptive driving maneuver detection: A large-scale evaluation study. *IEEE Transactions on Intelligent Transportation Systems, 18*(9), 2330–2339.

Cervantes-Villanueva, J., Carrillo-Zapata, D., Terroso-Saenz, F., Valdes-Vela, M., & Skarmeta, A. F. (2016). Vehicle maneuver detection with accelerometer-based classification. *Sensors (Switzerland), 16*(10), 1–23.

Chaaraoui, A. A., Climent-Perez, P., & Florez-Revuelta, F. (2012). A review on vision techniques applied to human behaviour analysis for ambient-assisted living. *Expert Systems with Applications, 39*(12), 10873–10888.

Charniak, E., & Goldman, R. P. (1993). A Bayesian model of plan recognition. *Artificial Intelligence, 64*(1), 53–79.

Chen, L., Hoey, J., Nugent, C. D., Cook, D. J., & Yu, Z. (2012). Sensor-based activity recognition. *IEEE Transactions on Systems, Man and Cybernetics Part C: Applications and Reviews, 42*(6), 790–808.

Costa, B. S. J., Škrjanc, I., Blažič, S., & Angelov, P. (2013). A practical implementation of self-evolving cloud-based control of a pilot plant. In *IEEE international conference on cybernetics, CYBCONF* (pp. 7–12).

Doshi, A., & Trivedi, M. M. (2011). Tactical driver behavior prediction and intent inference: A review. In *IEEE conference on intelligent transportation systems, Washington, DC, USA* (pp. 1892–1897).

Iglesias, J. A., Angelov, P., Ledezma, A., & Sanchis, A. (2010). Evolving classification of agents' behaviors: A general approach. *Evolving Systems, 1*(3), 161–171.

Iglesias, J. A., Ledezma, A., & Sanchis, A. (2009). Caos coach 2006 simulation team: An opponent modelling approach. *Computing and Informatics, 28*(1), 57–80.

Kalhor, A., Araabi, B. N., & Lucas, C. (2013). Evolving Takagi-Sugeno fuzzy model based on switching to neighboring models. *Applied Soft Computing Journal, 13*(2), 939–946.

Ke, S.-R., Thuc, H., Lee, Y.-J., Hwang, J.-N., Yoo, J.-H., & Choi, K.-H. (2013). A review on video-based human activity recognition. *Computers, 2*(2), 88–131.

Ledezma, A., Aler, R., Sanchis, A., & Borrajo, D. (2009). OMBO: An opponent modeling approach. *AI Communications, 22*(1), 21–35.

Levermore, T., Ordys, A., & Deng, J. (2014). A review of driver modelling. In *2014 UKACC international conference on control, Loughborough, U.K, July* (pp. 296–300).

Liu, X., Xu, F., & Fujimura, K. (2002). Real-time eye detection and tracking for driver observation under various light conditions. In *Intelligent vehicle symposium* (pp. 344–351).

Lughofer, E., & Klement, E. P. (2005). FLEXFIS: A variant for incremental learning of Takagi-Sugeno fuzzy systems. In *Proceedings of the 2005 IEEE international conference on fuzzy systems* (pp. 915–920).

Memon, M. A., Angelov, P., & Ahmed, H. (2006). An approach to real-time color-based object tracking. In *Proceedings of the 2006 international symposium on evolving fuzzy systems, EFS'06* (pp. 86–91).

Michon, J. A. (1985). A critical view of driver behavior models: What do we know, what should we do? In *Human behavior and traffic safety* (pp. 485–520). London: Plenum press, New York.

Miyajima, C., Nishiwaki, Y., Ozawa, K., Wakita, T., Itou, K., Takeda, K., & Itakura, F. (2007). Driver modeling based on driving behavior and its evaluation in driver identification. *Proceedings of the IEEE, 95*(2), 427–437.

Moeslund, T. B., Hilton, A., & Kruger, V. (2006). A survey of advances in vision-based human motion capture and analysis. *Computer Vision and Image Understanding, 104*(2–3 SPEC. ISS.), 90–126.

Nakano, A., Okuda, H., Suzuki, T., Inagaki, S., & Hayakawa, S. (2009). Symbolic modeling of driving behavior based on hierarchical mode segmentation and formal grammar. In *2009 IEEE/RSJ international conference on intelligent robots and systems, St. Louis, USA* (pp. 5516–5521).

Panou, M., Bekiaris, E., & Papakostopoulos, V. (2007). Modelling driver behaviour in european and international projects. In P. C. Cacciabue (Ed.), *Modelling driver behaviour in automotive environments: Critical issues in driver interactions with intelligent transport systems* (pp. 3–25). London: Springer London.

Rosa, R., Gomide, F., Dovžan, D., & Škrjanc, I. (2014). Evolving neural network with extreme learning for system modeling. *Evolving and Adaptive Intelligent Systems (EAIS), 2014 IEEE Conference on*, 1–7.

Sathyanarayana, A., Boyraz, P., & Hansen, J. H. (2008). Driver behavior analysis and route recognition by Hidden Markov Models. In *2008 IEEE international conference on vehicular electronics and safety, Columbus, OH, USA* (pp. 276–281).

Shi, B., Xu, L., Hu, J., Tang, Y., Jiang, H., Meng, W., & Liu, H. (2015). Evaluating driving styles by normalizing driving behavior based on personalized driver modeling. *IEEE Transactions on Systems, Man, and Cybernetics: Systems, 45*(12), 1502–1508.

Sigari, M. H., Fathy, M., & Soryani, M. (2013). A driver face monitoring system for fatigue and distraction detection. *International Journal of Vehicular Technology, 2013*, 1–11.

Škrjanc, I., Blažič, S., & Angelov, P. (2014). Robust evolving cloud-based PID control adjusted by gradient learning method. *2014 IEEE Conference on Evolving and Adaptive Intelligent Systems (EAIS)*, 1–8.

Steffens, T. (2004). Feature-based declarative opponent-modelling. In D. Polani, B. Browning, A. Bonarini, & K. Yoshida (Eds.), *Robocup 2003: Robot soccer world cup VII* (pp. 125–136). Springer, Berlin, Heidelberg.

Sukthankar, G., Geib, C., Hai Bui, H., Pynadath, D., & Goldman, R. P. (2014). An Introduction to Plan, Activity, and Intent Recognition. In G. Sukthankar, R. P. Goldman, C. Geib, D. V. Pynadath, & H. H. Bui (Eds.), *Plan, Activity, and Intent Recognition* p. 22. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Systems Technology (2016). Car driving simulator & simulation software. http://www.stisimdrive.com/products/simulation-systems/m100-series.

Takagi, T., & Sugeno, M. (1985). Fuzzy identification of systems and its applications to modeling and control. *IEEE Transactions on Systems, Man and Cybernetics, SMC-15*(1), 116–132.

Tiesheng Wang, Pengfei Shi, Wang, T., & Shi, P. (2005). Yawning detection for determining driver drowsiness. In *Proceedings of international workshop on VLSI design and video technology* (pp. 373–376).

Torkkola, K., Gardner, M., Schreiner, C., Zhang, K., Leivian, B., & Zhang, H. (2008). Computational intelligence in automotive applications. In D. Prokhorov (Ed.), *Computational intelligence in automotive applications. studies in computational intelligence: Vol. 132* (pp. 59–77).

Torkkola, K., Venkatesan, S., & Liu, H. (2005). Sensor sequence modeling for driving sequence modeling for driving. In *Flairs conference, July* (pp. 721–727). AAAI Press.

Vernaza, A., Ledezma, A., & Sanchis, A. (2014). Simul-A2: Agent-based simulator for evaluate ADA systems. In *17th international conference on information fusion (fusion), Salamanca, Spain* (pp. 1–7).

Wang, W., Xi, J., & Chen, H. (2014). Modeling and recognizing driver behavior based on driving data: A survey. *Mathematical Problems in Engineering, 2014*, 1–20.

Webb, G., Pazzani, M., & Billsus, D. (2001). Machine learning for user modeling. *User Modeling and User-Adapted Interaction, 11*(1), 19–29.

Zamora, V., Sipele, O., Ledezma, A., & Sanchis, A. (2017). Intelligent agents for supporting driving tasks: An ontology-based alarms system. In *Proceedings of the 3rd international conference on vehicle technology and intelligent transport systems* (pp. 165–172). Porto, Portugal: SciTePress.

Zhang, G., Cheng, B., Feng, R., & Zhang, X. (2008). A real-time adaptive learning method for driver eye detection. In *2008 digital image computing: techniques and applications* (pp. 300–304).