Understanding Driving Activity Using Ensemble Methods

Kari Torkkola, Mike Gardner, Chris Schreiner, Keshu Zhang, Bob Leivian, Harry Zhang, and John Summers

Motorola Labs, Tempe, AZ 85282, USA, kari.torkkola@motorola.com

1 Introduction

Motivation for the use of statistical machine learning techniques in the automotive domain arises from our development of context aware intelligent driver assistance systems, specifically, Driver Workload Management systems. Such systems integrate, prioritize, and manage information from the roadway, vehicle, cockpit, driver, infotainment devices, and then deliver it through a multimodal user interface. This could include incoming cell phone calls, email, navigation information, fuel level, and oil pressure to name a very few. In essence, the workload manager attempts to get the right information to the driver at the right time and in the right way in order that driver performance is optimized and distraction is minimized.

In order to do its job, the workload manager system needs to track the wider driving context including the state of the roadway, traffic conditions, and the driver. Current automobiles have a large number of embedded sensors, many of which produce data that are available through the car data bus. The state of many on-board and carried-in devices in the cockpit is also available. New advanced sensors, such as video-based lane departure warning systems and radar-based collision warning systems are currently being deployed in high end car models. All of these could be used to define the driving context [17]. But a number of questions arise:

- What are the range of typical driving maneuvers, as well as near misses and accidents, and how do drivers navigate them?
- Under what conditions does driver performance degrade or driver distraction increase?
- What are the optimal set of sensors and algorithms that can recognize each of these driving conditions near the theoretical limit for accuracy, and what is the sensor set that are accurate yet cost effective?

There are at least two approaches to address these questions. The first is more or less heuristic: experts offer informed opinions on the various matters and sets of sensors are selected accordingly with algorithms coded using rules of thumb. Individual aspects of the resulting system are tested by using narrow human factors testing controlling all but a few variables. Hundreds of iterations must be completed in order to test the impact on driver performance of the large combinations of driving states, sensor sets, and various algorithms.

The second approach, which we advocate in this chapter and in [28], involves using statistical machine learning techniques that enable the creation of new human factors approaches. Rather than running large numbers of narrowly scoped human factors testing with only a few variables, we chose to invent a "Hypervariate Human Factors Test Methodology" that uses broad naturalistic driving experiences that capture wide conditions resulting in hundreds of variables, but decipherable with machine learning techniques. Rather than pre-selecting our sensor sets we chose to collect data from every sensor we could think of and some not yet invented that might remotely be useful by creating behavioral models overlaid with our vehicle system. Furthermore, all sensor outputs were expanded by standard mathematical transforms that emphasized various aspects of the sensor signals. Again, data relationships are discoverable with machine learning. The final data set consisted of hundreds of driving hours with thousands of variable data outputs which would have been nearly impossible to annotate without machine learning techniques.

In this chapter we describe three major efforts that have employed our machine learning approach. First, we discuss how we have utilized our machine learning approach to detect and classify a wide range of driving maneuvers, and describe a semi-automatic data annotation tool we have created to support our modeling effort. Second, we perform a large scale automotive sensor selection study towards intelligent driver assistance systems. Finally, we turn our attention to creating a system that detects driver inattention by using sensors that are available in the current vehicle fleet (including forwarding looking radar and video-based lane departure system) instead of head and eye tracking systems.

This approach resulted in the creation of two generations of our workload manager system called Driver Advocate, Driver Advocate that was based on data rather than just expert opinions. The described techniques helped reduce the research cycle times while resulting in broader insight. There was rigorous quantification of theoretical sensor subsystem performance limits and optimal subsystem choices given economic price points. The resulting system performance specs and architecture design created a workload manager that had a positive impact on driver performance [23, 33].

2 Modeling Naturalistic Driving

Having the ability to detect driving maneuvers can be of great benefit in determining a driver's current workload state. For instance, a driving workload manager may decide to delay presenting the driver with non-critical information if the driver was in the middle of a complex driving maneuver. In this section we describe our data-driven approach to classifying driving maneuvers.

There are two approaches to collecting large databases of driving sensor data from various driving situations. One can outfit a fleet of cars with sensors and data collection equipment, as has been done in the NHTSA 100-car study [18]. This has the advantage of being as naturalistic as possible. However, the disadvantage is that potentially interesting driving situations will be extremely rare in the collected data. Realistic driving simulators provide much more controlled environments for experimentation and permit the creation of many interesting driving situations within a reasonable time frame. Furthermore, in a driving simulator, it is possible to simulate a large number of potential advanced sensors that would be yet too expensive or impossible to install in a real car. This will also enable us to study what sensors really are necessary for any particular task and what kind of signal processing of those sensors is needed in order to create adequate driving situation models based on those sensors.

We collect data in a driving simulator lab, which is an instrumented car in a surround video virtual world with full visual and audio simulation (although no motion or G-force simulation) of various roads, traffic and pedestrian activity. The driving simulator consists of a fixed based car surrounded by five front and three rear screens (Fig. 1). All driver controls such as the steering wheel, brake, and accelerator are monitored and affect the motion through the virtual world in real-time. Various hydraulics and motors provide realistic force feedback to driver controls to mimic actual driving.

The basic driving simulator software is a commercial product with a set of simulated sensors that, at the behavioral level, simulate a rich set of current and near future on-board sensors (http://www.drivesafety.com). This set consists of a radar for locating other traffic, a GPS system for position information, a camera system for lane positioning and lane marking detection, and a mapping data base for road names, directions, locations of points of interest, etc. There is also a complete car status system for determining the state of engine parameters (coolant temperature, oil pressure, etc.) and driving controls (transmission gear selection, steering angle, gas pedal, brake pedal, turn signal, window and seat belt status, etc.). The simulator setup also has several video cameras, microphones, and infrared eye tracking sensors to record all driver actions during the drive in synchrony with all the sensor output and simulator tracking variables. The Seeing Machines eye tracking system is used to automatically acquire a driver's head and eye movements (http://www.seeingmachines.com). Because such eye tracking systems are not installed in current vehicles, head and eye movement variables do not enter into the machine learning algorithms as input. The 117 head and eye-tracker variables are recorded as two versions, real-time and filtered. Including both

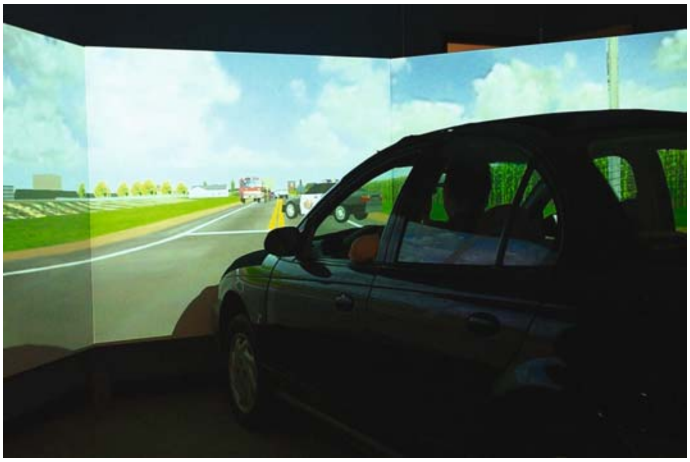


Fig. 1. The driving simulator

versions, there are altogether 476 variables describing an extensive scope of driving data – information about the auto, the driver, the environment, and associated conditions. An additional screen of video is digitally captured in MPEG4 format, consisting of a quad combiner providing four different views of the driver and environment. Combined, these produce around 400 MB of data for each 10 min of drive time. Thus we are faced with processing massive data sets of mixed type; there are both numerical and categorical variables, and multimedia, if counting also the video and audio.

3 Database Creation

We describe now our approach to collecting and constructing a database of naturalistic driving data in the driving simulator. We concentrate on the machine learning aspect; making the database usable as the basis for learning driving/driver situation classifiers and detectors. Note that the same database can be (and will be) used in driver behavioral studies, too.

3.1 Experiment Design

Thirty-six participants took part in this study, with each participant completing about ten 1-hour driving sessions. In each session, after receiving practice drives to become accustomed to the simulated driving environment, participants were given a task for which they have to drive to a specific location. These drives were designed to be as natural and familiar for the participants as possible, and the simulated world replicated the local metropolitan area as much as possible, so that participants did not need navigation aids to drive to their destinations. The driving world was modeled on the local Phoenix, AZ topology. Signage corresponded to local street and Interstate names and numbers. The topography corresponded as closely as possible to local landmarks.

The tasks included driving to work, driving from work to pick up a friend at the airport, driving to lunch, and driving home from work. Participants were only instructed to drive as they normally would. Each drive varied in length from 10 to 25 min. As time allowed, participants did multiple drives per session.

This design highlights two crucial components promoting higher realism in driving and consequently in collected data: (1) familiarity of the driving environment, and (2) immersing participants in the tasks. The experiment produced a total of 132 h of driving time with 315 GB of collected data.

3.2 Annotation of the Database

We have created a semi-automatic data annotation tool to label the sensor data with 28 distinct driving maneuvers. This data annotation tool is unique in that we have made parts of the annotation process automatic, enabling the user just to verify automatically generated annotations, rather than annotating everything from scratch.

The purpose of the data annotation is to label the sensor data with meaningful classes. Supervised learning and modeling techniques then become available with labeled data. For example, one can train classifiers for maneuver detection [29] or for inattention detection [30]. Annotated data also provides a basis for research in characterizing driver behavior in different contexts.

The driver activity classes in this study were related to maneuvering the vehicle with varying degrees of required attention. An alphabetical listing is presented in Table 1. Note that the classes are not mutually exclusive. An instant in time can be labeled simultaneously as "TurningRight" and "Starting," for example.

We developed a special purpose data annotation tool for the driving domain (Fig. 2). This was necessary because available video annotation tools do not provide a simultaneous view of the sensor data, and tools meant for signals, such as speech, do not allow simultaneous and synchronous playback of the video. The major properties of our annotation tool are listed below:

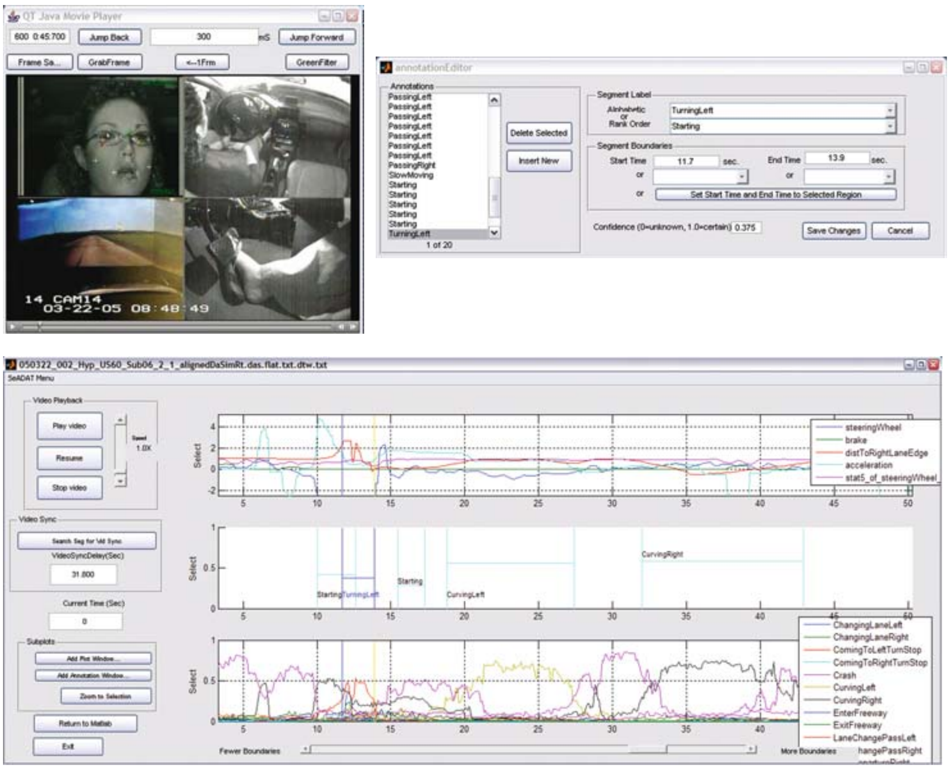
1. Ability to navigate through any portion of the driving sequence
2. Ability to label (annotate) any portion of the driving sequence with proper time alignment
3. Synchronization between video and other sensor data
4. Ability to playback the video corresponding to the selected sensor signal segment
5. Ability to visualize any number of sensor variables
6. Provide persistent storage of the annotations
7. Ability to modify existing annotations

Since manual annotation is a tedious process, we automated parts of the process by taking advantage of automatic classifiers that are trained from data to detect the driving maneuvers. With these classifiers, annotation becomes an instance of active learning [7]. Only if a classifier is not very confident in its decision, its results are presented to the human to verify. The iterative annotation process is thus as follows:

Table 1. Driving maneuvers used in the study

ChangingLaneLeft	ChangingLaneRight
ComingToLeftTurnStop	ComingToRightTurnStop
Crash	CurvingLeft
CurvingRight	EnterFreeway
ExitFreeway	LaneChangePassLeft
LaneChangePassRight	LaneDepartureLeft
LaneDepartureRight	Merge
PanicStop	PanicSwerve
Parking	PassingLeft
PassingRight	ReversingFromPark
RoadDeparture	SlowMoving
Starting	StopAndGo
Stopping	TurningLeft
TurningRight	WaitingForGapInTurn
Cruising (other)	

"Cruising" captures anything not included in the actual 28 classes



600 0:45:700 | 300 mS

Annotations: PassingLeft, PassingLeft, PassingLeft, PassingLeft, PassingLeft, PassingLeft, PassingLeft, PassingRight, SlowMoving, Starting, Starting, Starting, Starting, Starting, TurningLeft — 1 of 20

Segment Label — Alphabetic or Rank Order: TurningLeft; Starting

Segment Boundaries — Start Time 11.7 sec. End Time 13.9 sec. — Set Start Time and End Time to Selected Region

Confidence (0=unknown, 1.0=certain) 0.375

Speed 1.0X

VideoSyncDelay(Sec) 31.800

Current Time (Sec) 0

Fig. 2. The annotation tool. *Top left*: the video playback window. *Top right*: annotation label window. *Bottom*: sensor signal and classifier result display window. Note that colors have been re-used. See Fig. 5 for a complete legend

1. Manually annotate a small portion of the driving data
2. Train classifiers based on all annotated data
3. Apply classifiers to a portion of database
4. Present unsure classifications to the user to verify
5. Add new verified and annotated data to the database
6. Go to 2

As the classifier improves due to increased size of training data, the decisions presented to the user improve, too, and the verification process takes less time [25]. The classifier is described in detail in the next section.

4 Driving Data Classification

We describe now a classifier that has turned out to be very appropriate for driving sensor data. Characteristics of this data are hundreds of variables (sensors), millions of observations, and mixed type data. Some variables have continuous values and some are categorical. The latter fact causes problems with conventional statistical classifiers that typically operate entirely with continuous valued variables. Categorical variables need to be converted first into binary indicator variables. If a categorical variable has a large number of levels (possible discrete values), each of them generates a new indicator variable, thus potentially multiplying the dimension of the variable vector. We attempted this approach using Support Vector Machines [24] as classifiers, but the results were inferior compared to ensembles of decision trees.

4.1 Decision Trees

Decision trees, such as CART [6], are an example of non-linear, fast, and flexible base learners that can easily handle massive data sets even with mixed variable types.

A decision tree partitions the input space into a set of disjoint regions, and assigns a response value to each corresponding region (see Fig. 3). It uses a greedy, top-down recursive partitioning strategy. At every step a decision tree uses exhaustive search by trying all combinations of variables and split points to achieve the maximum reduction in node impurity. In a classification problem, a node is "pure" if all the training data in the node has the same class label. Thus the tree growing algorithm tries to find a variable and a split point of the variable that best separates the data in the node into different classes. Training data is then divided among the resulting nodes according to the chosen decision test. The process is repeated for each resulting node until a certain maximum node depth is reached, or until the nodes become pure. The tree constructing process itself can be considered as a type of embedded variable selection, and the impurity reduction due to a split on a specific variable could indicate the relative importance of that variable to the tree model.

For a single decision tree, a measure of variable importance is proposed in [6]:

$$VI(x_i, T) = \sum_{t \in T} \Delta I(x_i, t) \tag{1}$$

where $\Delta I(x_i, t) = I(t) - p_L I(t_L) - p_R I(t_R)$ is the decrease in impurity due to an actual (or potential) split on variable x_i at a node t of the optimally pruned tree T. The sum in (1) is taken over all internal tree nodes where x_i is a primary splitter. Node impurity $I(t)$ for classification $I(t) = Gini(t)$ where $Gini(t)$ is the Gini index of node t:

$$Gini(t) = \sum_{i \neq j} p_i^t p_j^t \tag{2}$$

and p_i^t is the proportion of observations in t whose response label equals i $(y = i)$ and i and j run through all response class numbers. The Gini index is in the same family of functions as *cross-entropy*, $-\sum_i p_i^t log(p_i^t)$, and measures node impurity. It is zero when t has observations only from one class, and reaches its maximum when the classes are perfectly mixed.

However, a single tree is inherently instable. The ability of a learner (a classifier in this case) to generalize to new unseen data is closely related to the stability of the learner. The stability of the solution could be

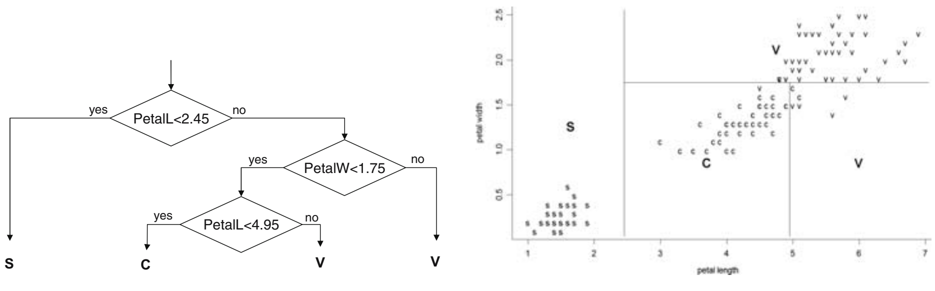


Fig. 3. An example of a decision tree for a three-class classification problem. The right side depicts the data with two variables. The final decision regions are also displayed

loosely defined as a continuous dependence on the training data. A stable solution changes very little when the training data set changes a little. With decision trees, the node structure can change drastically even when one data point is added or removed from the training data set. A comprehensive treatment of the connection between stability and generalization ability can be found in [3].

Instability can be remedied by employing ensemble methods. Ensemble methods train multiple simple learners and then combine the outputs of the learners for the final decision. One well known ensemble method is bagging (bootstrap aggregation) [4]. Bagging decision trees is explained in detail in Sect. 4.2.

Bagging can dramatically reduce the variance of instable learners by providing a regularization effect. Each individual learner is trained using a different random sample set of the training data. Bagged ensembles do not overfit the training data. The keys to good ensemble performance are base learners that have a low bias, and a low correlation between their errors. Decision trees have a low bias, that is, they can approximate any nonlinear decision boundary between classes to a desirable accuracy given enough training data. Low correlation between base learners can be achieved by sampling the data, as described in the following section.

4.2 Random Forests

Random Forest (RF) is a representative of tree ensembles [5]. It grows a forest of decision trees on bagged samples (Fig. 4). The "randomness" originates from creating the training data for each individual tree by sampling both the data and the variables. This ensures that the errors made by individual trees are uncorrelated, which is a requirement for bagging to work properly.

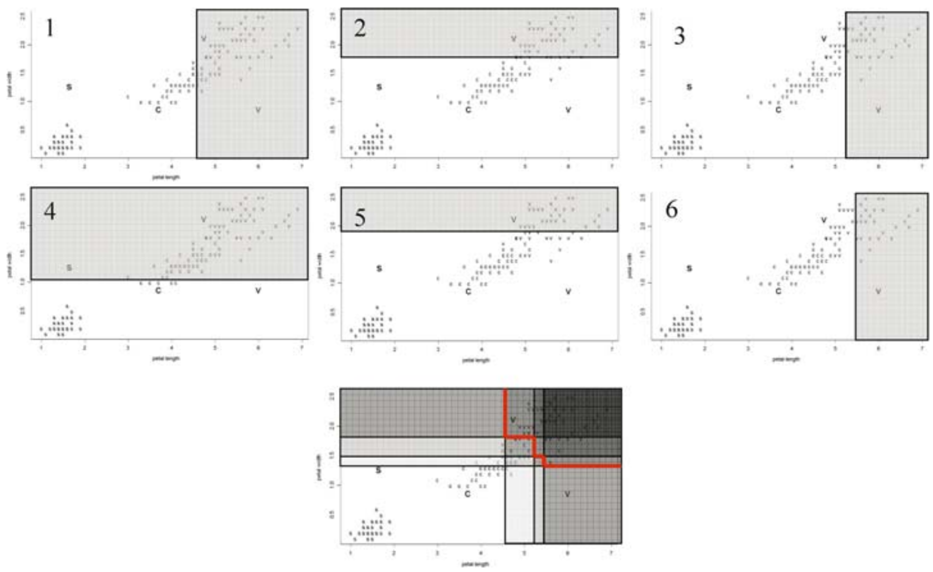


Fig. 4. A trivial example of Random Forest in action. The task is to separate class "v" from "c" and "s" (*upper right corner*) based on two variables only. Six decision trees are constructed sampling both examples and variables. Each tree becomes now a single node sampling one out of the two possible variables. Outputs (decision regions) are averaged and thresholded. The final nonlinear decision border is outlined as a *thick line*

Each tree in the Random Forest is grown according to the following parameters:

1. A number m is specified much smaller than the total number of total input variables M (typically m is proportional to $\sqrt{M}$).
2. Each tree of maximum depth (until pure nodes are reached) is grown using a bootstrap sample of the training set.
3. At each node, m out of the M variables are selected at random.
4. The split used is the best possible split on these m variables only.

Note that for each tree to be constructed, *bootstrap sampling* is applied. A different sample set of training data is drawn with replacement. The size of the sample set is the same as the size of the original dataset. This means that some individual samples will be duplicated, but typically 30% of the data is left out of this sample (out-of-bag). This data has a role in providing an unbiased estimate of the performance of the tree.

Also note that the sampled variable set does not remain constant while a tree is grown. Each new node in a tree is constructed based on a *different* random sample of m variables. The best split among these m variables is chosen for the current node, in contrast to typical decision tree construction, which selects the best split among all possible variables. This ensures that the errors made by each tree of the forest are not correlated. Once the forest is grown, a new sensor reading vector will be classified by every tree of the forest. Majority voting among the trees produces then the final classification decision.

We will be using RF throughout our experimentation because of it simplicity and excellent performance.[1] In general, RF is resistant to irrelevant variables, it can handle massive numbers of variables and observations, and it can handle mixed type data and missing data. Our data definitely is of mixed type, i.e., some variables are continuous, some variables are discrete, although we do not have missing data since the source is the simulator.

4.3 Random Forests for Driving Maneuver Detection

A characteristic of the driving domain and the chosen 29 driving maneuver classes is that the classes are not mutually exclusive. For example, an instance in time could be classified simultaneously as "SlowMoving" and "TurningRight." The problem cannot thus be solved by a typical multi-class classifier that assigns a single class label to a given sensor reading vector and excludes the rest. This dictates that the problem should be treated rather as a *detection* problem than a *classification* problem.

Furthermore, each maneuver is inherently a sequential operation. For example, "ComingToLeftTurnStop" consists of possibly using the turn signal, changing the lane, slowing down, braking, and coming to a full stop. Ideally, a model of a maneuver would thus describe this sequence of operations with variations that naturally occur in the data (as evidenced by collected naturalistic data). Earlier, we have experimented with Hidden Markov Models (HMM) for maneuver classification [31]. A HMM is able to construct a model of a sequence as a chain of hidden states, each of which has a probabilistic distribution (typically Gaussian) to match that particular portion of the sequence [22]. The sequence of sensor vectors corresponding to a maneuver would thus be detected as a whole.

The alternative to sequential modeling is instantaneous classification. In this approach, the whole duration of a maneuver is given just a single class label, and the classifier is trained to produce this same label for every time instant of the maneuver. Order, in which the sensor vectors are observed, is thus not made use of, and the classifier carries the burden of being able to capture all variations happening inside a maneuver under a single label. Despite these two facts, in our initial experiments the results obtained using Random Forests for instantaneous classification were superior to Hidden Markov Models.

Because the maneuver labels may be overlapping, we trained a separate Random Forest for each maneuver treating it as a binary classification problem – the data of a particular class against all the other data. This results in 29 trained "detection" forests.

[1] We use Leo Breiman's Fortran version 5.1, dated June 15, 2004. An interface to Matlab was written to facilitate easy experimentation. The code is available at http://www.stat.berkeley.edu/users/breiman/.

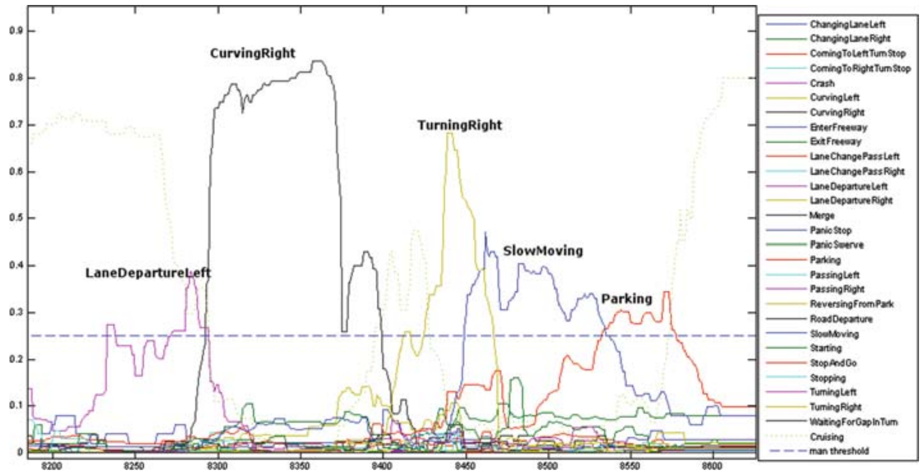


Fig. 5. A segment of driving with corresponding driving maneuver probabilities produced by one Random Forest trained for each maneuver class to be detected. Horizontal axis is the time in tenths of a second. Vertical axis is the probability of a particular class. These "probabilities" can be obtained by normalizing the random forest output voting results to sum to one

New sensor data is then fed to all 29 forests for classification. Each forest produces something of a "probability" of the class it was trained for. An example plot of those probability "signals" is depicted in Fig. 5. The horizontal axis represents the time in tenths of a second. About 45 s of driving is shown. None of the actual sensor signals are depicted here, instead, the "detector" signals from each of the forests are graphed. These show a sequence of driving maneuvers from "Cruising" through "LaneDepartureLeft," "CurvingRight," "TurningRight," and "SlowMoving" to "Parking."

The final task is to convert the detector signals into discrete and possibly overlapping labels, and to assign a confidence value to each label. In order to do this, we apply both median filtering and low-pass filtering to the signals. The signal at each time instant is replaced by the maximum of the two filtered signals. This has the effect of patching small discontinuities and smoothing the signal while still retaining fast transitions. Any signal exceeding a global threshold value for a minimum duration is then taken as a segment. Confidence of the segment is determined as the average of the detection signal (the probability) over the segment duration.

An example can be seen at the bottom window depicted in Fig. 2. The top panel displays some of the original sensor signals, the bottom panel graphs the raw maneuver detection signals, and the middle panel shows the resulting labels.

We compared the results of the Random Forest maneuver detector to the annotations done by a human expert. On the average, the annotations agreed 85% of the time. This means that only 15% needed to be adjusted by the expert. Using this semi-automatic annotation tool, we can drastically reduce the time that is required for data processing.

5 Sensor Selection Using Random Forests

In this section we study which sensors are necessary for driving state classification. Sensor data is collected in our driving simulator; it is annotated with driving state classes, after which the problem reduces to that of feature selection [11]: "Which sensors contribute most to the correct classification of the driving state into various maneuvers?" Since we are working with a simulator, we have simulated sensors that would be expensive to arrange in a real vehicle. Furthermore, sensor behavioral models can be created in software. Goodness or noisiness of a sensor can be modified at will. The simulator-based approach makes it possible to study the problem without implementing the actual hardware in a real car.

Variable selection methods can be divided in three major categories [11, 12, 16]. These are:

1. Filter methods, that evaluate some measure of relevance for all the variables and rank them based on the measure (but the measure may not necessarily be relevant to the task, and any interactions that variables may have will be ignored)
2. Wrapper methods, that using some learner, actually learn the solution to the problem evaluating all possible variable combinations (this is usually computationally too prohibitive for large variable sets)
3. Embedded methods that use a learner with all variables, but infer the set of important variables from the structure of the trained learner

Random Forests (Sect. 4.2) can act as an embedded variable selection system. As a by-product of the construction, a measure of variable importance can be derived from each tree, basically from how often different variables were used in the splits of the tree and from the quality of those splits [5]. For an ensemble of N trees the importance measure (1) is simply averaged over the ensemble.

$$M(x_i) = \frac{1}{N} \sum_{n=1}^{N} VI(x_i, T_n) \tag{3}$$

The regularization effect of averaging makes this measure much more reliable than a measure extracted from just a single tree.

One must note that in contrast to simple filter methods of feature selection, this measure considers multiple simultaneous variable interactions – not just two at a time. In addition, the tree is constructed for the exact task of interest. We apply now this importance measure to driving data classification.

5.1 Sensor Selection Results

As the variable importance measure, we use the tree node impurity reduction (1) summed over the forest (3). This measure does not require any extra computation in addition to the basic forest construction process. Since a forest was trained for each class separately, we can now list the variables in the order of importance for each class. These results are combined and visualized in Fig. 6.

This figure has the driving activity classes listed at the bottom, and variables on the left column. Head- and eye-tracking variables were excluded from the figure. Each column of the figure thus displays the importances of all listed variables, for the class named at the bottom of the column. White, through yellow, orange, red, and black, denote decreasing importance.

In an attempt to group together those driving activity classes that require a similar set of variables to be accurately detected, and to group together those variables that are necessary or helpful for a similar set of driving activity classes, we clustered first the variables in six clusters and then the driving activity classes in four clusters. Any clustering method can be used here, we used spectral clustering [19]. Rows and columns are then re-ordered according to the cluster identities, which are indicated in the names by alternating blocks of red and black font in Fig. 6.

Looking at the variable column on the left, the variables within each of the six clusters exhibit a similar behavior in that they are deemed important by approximately the same driving activity classes. The topmost cluster of variables (except "Gear") appears to be useless in distinguishing most of the classes. The next five variable clusters appear to be important but for different class clusters. Ordering of the clusters as well as variable ordering within the clusters is arbitrary. It can also be seen that there are variables (sensors) that are important for a large number of classes.

In the same fashion, clustering of the driving activity classes groups those classes together that need similar sets of variables in order to be successfully detected. The rightmost and the leftmost clusters are rather distinct, whereas the two middle clusters do not seem to be that clear. There are only a few classes that can be reliably detected using a handful of sensors. Most notable ones are "ReversingFromPark" that

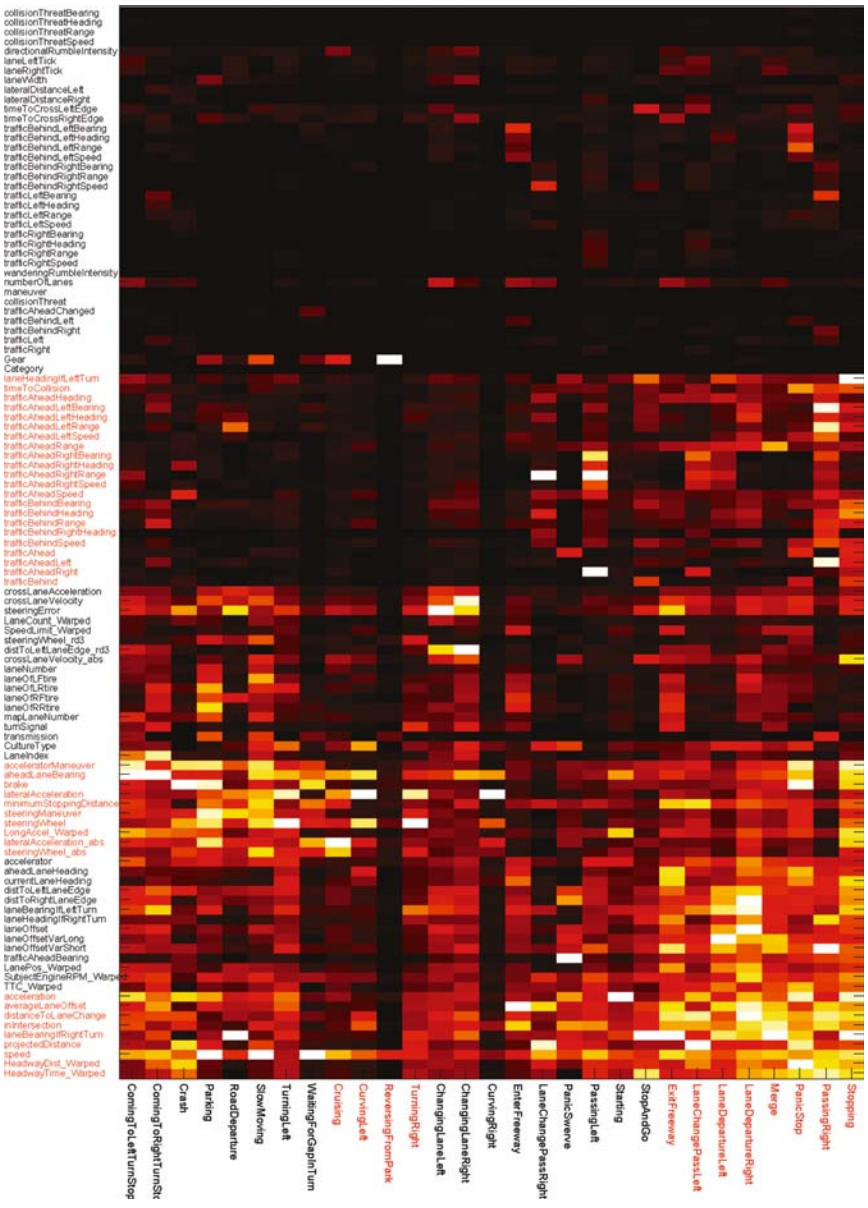


Fig. 6. Variable importances for each class. See text for explanation

only needs “Gear,” and “CurvingRight” that needs “lateralAcceleration” with the aid of “steeringWheel.” This clustering shows that some classes need quite a wide array of sensors in order to be reliably detected. The rightmost cluster is an example of those classes.

5.2 Sensor Selection Discussion

We present first results of a large scale automotive sensor selection study aimed towards intelligent driver assistance systems. In order to include both traditional and advanced sensors, the experiment was done on a driving simulator. This study shows clusters of both sensors and driving state classes: what classes need similar sensor sets, and what sensors provide information for which sets of classes. It also provides a basis to study detecting an isolated driving state or a set of states of interest and the sensors required for it.

6 Driver Inattention Detection Through Intelligent Analysis of Readily Available Sensors

Driver inattention is estimated to be a significant factor for 78% of all crashes [8]. A system that could accurately detect driver inattention could aid in reducing this number. In contrast to using specialized sensors or video cameras to monitor the driver we detect driver inattention by using only readily available sensors. A classifier was trained using Collision Avoidance Systems (CAS) sensors which was able to accurately identify 80% of driver inattention and could be added to a vehicle without incurring the cost of additional sensors.

Detection of driver inattention could be utilized in intelligent systems to control electronic devices [21] or redirect the driver’s attention to critical driving tasks [23].

Modern automobiles contain many infotainment devices designed for driver interaction. Navigation modules, entertainment devices, real-time information systems (such as stock prices or sports scores), and communication equipment are increasingly available for use by drivers. In addition to interacting with on-board systems, drivers are also choosing to carry in mobile devices such as cell phones to increase productivity while driving. Because technology is increasingly available for allowing people to stay connected, informed, and entertained while in a vehicle many drivers feel compelled to use these devices and services in order to multitask while driving.

This increased use of electronic devices along with typical personal tasks such as eating, shaving, putting on makeup, reaching for objects on the floor or in the back seat can cause the driver to become inattentive to the driving task. The resulting driver inattention can increase risk of injury to the driver, passengers, surrounding traffic and nearby objects.

The prevailing method for detecting driver inattention involves using a camera to track the driver’s head or eyes [9, 26]. Research has also been conducted on modeling driver behaviors through such methods as building control models [14, 15] measuring behavioral entropy [2] or discovering factors affecting driver intention [10, 20].

Our approach to detecting inattention is to use only sensors currently available on modern vehicles (possibly including Collision Avoidance Systems (CAS) sensors) without using head and eye tracking system. This avoids the additional cost and complication of video systems or dedicated driver monitoring systems. We derive several parameters from commonly available sensors and train an inattention classifier. This results in a sophisticated yet inexpensive system for detecting driver inattention.

6.1 Driver Inattention

What is Driver Inattention?

Secondary activities of drivers during inattention are many, but mundane. The 2001 NETS survey in Table 2 found many activities that drivers perform in addition to driving. A study, by the American Automobile Association placed miniature cameras in 70 cars for a week and evaluated three random driving hours from

Table 2. 2001 NETS survey

96%	Talking to passengers
89%	Adjusting vehicle climate/radio controls
74%	Eating a meal/snack
51%	Using a cell phone
41%	Tending to children
34%	Reading a map/publication
19%	Grooming
11%	Prepared for work

Activities drivers engage in while driving

each. Overall, drivers were inattentive 16.1% of the time they drove. About 97% of the drivers reached or leaned over for something and about 91% adjusted the radio. Thirty percent of the subjects used their cell phones while driving.

Causes of Driver Inattention

There are at least three factors affecting attention:

1. Workload. Balancing the optimal cognitive and physical workload between too much and boring is an everyday driving task. This dynamic varies from instant to instant and depends on many factors. If we chose the wrong fulcrum, we can be overwhelmed or unprepared.
2. Distraction. Distractions might be physical (e.g., passengers, calls, signage) or cognitive (e.g., worry, anxiety, aggression). These can interact and create multiple levels of inattention to the main task of driving.
3. Perceived Experience. Given the overwhelming conceit that almost all drivers rate their driving ability as superior than others, it follows that they believe they have sufficient driving control to take part of their attention away from the driving task and give it to multi-tasking. This "skilled operator" over-confidence tends to underestimate the risk involved and reaction time required. This is especially true in the inexperienced younger driver and the physically challenged older driver.

Effects of Driver Inattention

Drivers involved in crashes often say that circumstances occurred suddenly and could not be avoided. However, due to laws of physics and visual perception, very few things occur suddenly on the road. Perhaps more realistically an inattentive driver will suddenly notice that something is going wrong. This inattention or lack of concentration can have catastrophic effects. For example, a car moving at a slow speed with a driver inserting a CD will have the same effect as an attentive driver going much faster. Simply obeying the speed limits may not be enough.

Measuring Driver Inattention

Many approaches to measuring driver inattention have been suggested or researched. Hankey et al. suggested three parameters: average glance length, number of glances, and frequency of use [13]. The glance parameters require visual monitoring of the drivers face and eyes. Another approach is using the time and/or accuracy of a surrogate secondary task such as Peripheral Detection Task (PDT) [32]. These measures are yet not practical real time measures to use during everyday driving.

Boer [1] used a driver performance measure, steering error entropy, to measure workload, which unlike eye gaze and surrogate secondary-tasks, is unobtrusive, practical for everyday monitoring, and can be calculated in near real time.

We calculate it by first training a linear predictor from "normal" driving [1]. The predictor uses four previous steering angle time samples (200 ms apart) to predict the next time sample. The residual (prediction error) is computed for the data. A ten-bin discretizer is constructed from the residual, selecting the discretization levels such that all bins become equiprobable. The predictor and discretizer are then fixed, and applied to a new steering angle signal, producing a discretized steering error signal. We then compute the running entropy of the steering error signal over a window of 15 samples using the standard entropy definition $E = -\sum_{i=1}^{10} p_i \log_{10} p_i$, where p_i are the proportions of each discretization level observed in the window.

Our work indicates that steering error entropy is able to detect driver inattention while engaged in secondary tasks. Our current study expands and extends this approach and looks at other driver performance variables, as well as the steering error entropy, that may indicate driver inattention during a common driving task, such as looking in the "blind spot."

Experimental Setup

We designed the following procedure to elicit defined moments of normal driving inattention.

The simulator authoring tool, HyperDrive, was used to create the driving scenario for the experiment. The drive simulated a square with curved corners, six kilometers on a side, 3-lanes each way (separated by a grass median) beltway with on- and off-ramps, overpasses, and heavy traffic in each direction. All drives used daytime dry pavement driving conditions with good visibility.

For a realistic driving environment, high-density random "ambient" traffic was programmed. All "ambient" vehicles simulated alert, "good" driver behavior, staying at or near the posted speed limit, and reacted reasonably to any particular maneuver from the driver.

This arrangement allowed a variety of traffic conditions within a confined, but continuous driving space. Opportunities for passing and being passed, traffic congestion, and different levels of driving difficulty were thereby encountered during the drive.

After two orientation and practice drives, we collected data while drivers drove about 15 min in the simulated world. Drivers were instructed to follow all normal traffic laws, maintain the vehicle close to the speed limit (55 mph, 88.5 kph), and to drive in the middle lane without lane changes. At 21 "trigger" locations scattered randomly along the road, the driver received a short burst from a vibrator located in the seatback on either the left or right side of their backs. This was their alert to look in their corresponding "blind spot" and observe a randomly selected image of a vehicle projected there. The image was projected for 5 s and the driver could look for any length of time he felt comfortable. They were instructed that they would receive "bonus" points for extra money for each correctly answered question about the images. Immediately after the image disappeared, the experimenter asked the driver questions designed to elicit specific characteristics of the image – i.e., What kind of vehicle was it?, Were there humans in the image?, What color was the vehicle?, etc.

Selecting Data for Inattention Detection

Though the simulator has a variety of vehicle, environment, cockpit, and driver parameters available for our use, our goal was to experiment with only readily extractable parameters that are available on modern vehicles. We experimented with two subsets of these parameter streams: one which used only traditional driver controls (steering wheel position and accelerator pedal position), and a second subset which included the first subset but also added variables available from CAS systems (lane boundaries, and upcoming road curvature). A list of variables used and a brief description of each is displayed in Table 3.

Eye/Head Tracker

In order to avoid having to manually label when the driver was looking away from the simulated road, an eye/head tracker was used (Fig. 7).

When the driver looked over their shoulder at an image in their blind spot this action caused the eye tracker to lose eye tracking ability (Fig. 8). This loss sent the eye tracking confidence to a low level. These periods of low confidence were used as the periods of inattention. This method avoided the need for hand labeling.

Table 3. Variables used to detect inattention

Variable	Description
steeringWheel	Steering wheel angle
accelerator	Position of accelerator pedal
distToLeftLaneEdge	Perpendicular distance of left front wheel from left lane edge
crossLaneVelocity	Rate of change of distToLeftLaneEdge
crossLaneAcceleration	Rate of change of crossLaneVelocity
steeringError	Difference between steering wheel position and ideal position for vehicle to travel exactly parallel to lane edges
aheadLaneBearing	Angle of road 60 m in front of current vehicle position

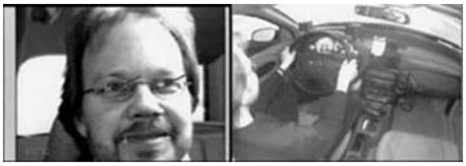


Fig. 7. Eye/head tracking during attentive driving

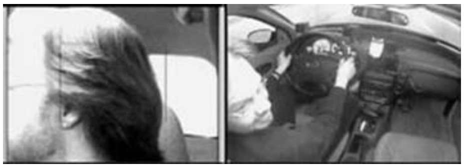


Fig. 8. Loss of eye/head tracking during inattentive driving

6.2 Inattention Data Processing

Data was collected from six different drivers as described above. This data was later synchronized and re-sampled at a constant sampling rate of 10 Hz. resulting in 40,700 sample vectors. In order to provide more relevant information to the task at hand, further parameters were derived from the original sensors. These parameters are as follows:

1. ra9: Running average of the signal over nine previous samples (smoothed version of the signal).
2. rd5: Running difference five samples apart (trend).
3. rv9: Running variance of nine previous samples according to the standard definition of sample variance.
4. ent15: Entropy of the error that a linear predictor makes in trying to predict the signal as described in [1]. This can be thought of as a measure of randomness or unpredictability of the signal.
5. stat3: Multivariate stationarity of a number of variables simultaneously three samples apart as described in [27]. Stationarity gives an overall rate of change for a group of signals. Stationarity is one if there are no changes over the time window and approaches zero for drastic transitions in all signals of the group.

The operations can be combined. For example, "_rd5_ra9" denotes first computing a running difference five samples apart and then computing the running average over nine samples.

Two different experiments were conducted.

1. The first experiment used only two parameters: steeringWheel and accelerator, and derived seven other parameters: steeringWheel_rd5_ra9, accelerator_rd5_ra9, stat3_of_steeringWheel_accel, steeringWheel_ent15_ra9, accelerator_ent15_ra9, steeringWheel_rv9, and accelerator_rv9.

2. The second experiment used all seven parameters in Table 1 and derived 13 others as follows: steeringWheel_rd5_ra9, steeringError_rd5_ra9, distToLeftLaneEdge_rd5_ra9, accelerator_rd5_ra9, aheadLaneBearing_rd5_ra9, stat3_of_steeringWheel_accel, stat3_of_steeringError_crossLaneVelocity_distToLeftLaneEdge_aheadLaneBearing, steeringWheel_ent15_ra9, accelerator_ent15_ra9, steeringWheel_rv9, accelerator_rv9, distToLeftLaneEdge_rv9, and crossLaneVelocity_rv9.

Variable Selection for Inattention

In variable selection experiments we are attempting to determine the relative importance of each variable to the task of inattention detection. First a Random Forest classifier is trained for inattention detection (see also Sect. 6.2). Variable importances can then be extracted from the trained forest using the approach that was outlined in Sect. 5.

We present the results in Tables 4 and 5. These tables provide answers to the question "Which sensors are most important in detecting driver's inattention?" When just the two basic "driver control" sensors were used, some new derived variables may provide as much new information as the original signals, namely the running variance and entropy of steering. When CAS sensors are combined, the situation changes: lane

Table 4. Important sensor signals for inattention detection derived from steering wheel and accelerator pedal

Variable	Importance
steeringWheel	100.00
accelerator	87.34
steeringWheel_rv9	68.89
steeringWheel_ent15_ra9	58.44
stat3_of_steeringWheel_accelerator	41.38
accelerator_ent15_ra9	40.43
accelerator_rv9	35.86
steeringWheel_rd5_ra9	32.59
accelerator_rd5_ra9	29.31

Table 5. Important sensor signals for inattention detection derived from steering wheel, accelerator pedal and CAS sensors

Variable	Importance
distToLeftLaneEdge	100.00
accelerator	87.99
steeringWheel_rv9	73.76
distToLeftLaneEdge_rv9	65.44
distToLeftLaneEdge_rd5_ra9	65.23
steeringWheel	64.54
Stat3_of_steeringWheel_accel	60.00
steeringWheel_ent15_ra9	57.39
steeringError	57.32
aheadLaneBearing_rd5_ra9	55.33
aheadLeneBearing	51.85
crossLaneVelocity	50.55
Stat3_of_steeringError_crossLaneVelocity_distToLeftLaneEdge_aheadLaneBearing	38.07
crossLaneVelocity_rv9	36.50
steeringError_rd5_ra9	33.52
Accelerator_ent15_ra9	29.83
Accelerator_rv9	28.69
steeringWheel_rd5_ra9	27.76
Accelerator_rd5_ra9	20.69
crossLaneAcceleration	20.64

position (distToLeftLaneEdge) becomes the most important variable together with the accelerator pedal. Steering wheel variance becomes the most important variable related to steering.

Inattention Detectors

Detection tasks always have a tradeoff between desired recall and precision. Recall denotes the percentage of total events of interest detected. Precision denotes the percentage of detected events that are true events of interest and not false detections. A trivial classifier that classifies every instant as a true event would have 100% recall (since none were missed), but its precision would be poor. On the other hand, if the classifier is so tuned that only events having high certainty are classified as true events, the recall would be low, missing most of the events, but its precision would be high, since among those that were classified as true events, only a few would be false detections. Usually any classifier has some means of tuning the threshold of detection. Where that threshold will be set depends on the demands of the application. It is also noteworthy to mention that in tasks involving detection of rare events, overall classification accuracy is not a meaningful measure. In our case only 7.3% of the database was inattention so a trivial classifier classifying everything as attention would thus have an accuracy of 92.7%. Therefore we will report our results using the recall and precision statistics for each class.

First, we constructed a Random Forests (RF) classifier of 75 trees using either the driver controls or the driver controls combined with the CAS sensors. Figure 9 depicts the resulting recall/precision graphs.

One simple figure of merit that allows comparison of two detectors is equal error rate, or equal accuracy, which denotes the intersection of recall and precision curves. By that figure, basing the inattention detector only on driver control sensors results in an equal accuracy of 67% for inattention and 97% for attention, whereas adding CAS sensors raises the accuracies up to 80 and 98%, respectively. For comparison, we used the same data to train a quadratic classifier [29]. Compared to the RF classifier, the quadratic classifier performs poorly in this task. We present the results in Table 6 for two different operating points of the quadratic classifier. The first one (middle rows) is tuned not to make false alarms, but its recall rate remains low. The second one is compensated for the less frequent occurrences of inattention, but it makes false alarms about 28% of the time. Random Forest clearly outperforms the quadratic classifier with almost no false alarms and good recall.

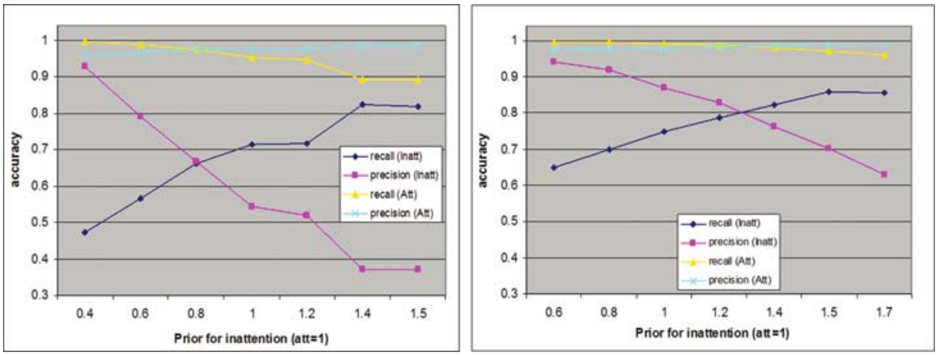


Fig. 9. Precision/recall figures for detection of inattention using only driver controls (*left*) and driver controls combined with CAS sensors (*right*). Random Forest is used as the classifier. Horizontal axis, prior for inattention, denotes a classifier parameter which can be tuned to produce different precision/recall operating points. This can be thought of as a weight or cost given to missing inattention events. Typically, if it is desirable not to miss events (high recall – in this case a high parameter value), the precision may be low – many false detections will be made. Conversely, if it is desirable not to make false detections (high precision), the recall will be low – not all events will be detected. Thus, as a figure of merit we use equal accuracy, the operating point where precision equals recall

Table 6. Comparison of Random Forest (RF) to Quadratic classifier using equal accuracy as the figure of merit

Detector	Sensors	Inattention (%)	Attention (%)
RF	Driver control sensors only	67	97
RF	Additional CAS sensors	80	98
Quadratic	Driver control sensors only	29	91
Quadratic	Additional CAS sensors	33	93
Quadratic (prior compensated)	Driver control sensors only	58	59
Quadratic (prior compensated)	Additional CAS sensors	72	72

Operating point where recall of the desired events equals the precision of the detector

Future Driver Inattention Work

The experiments described in this section are only our first steps in investigating how driver attention can be detected. We have several ideas of how to improve the accuracy of detectors based on modifications to our described approach. The first technique will be to treat inattentive periods as longer time segments that actually have a weighting mechanism that prohibits rapid toggling between states. Also, an edge detector could be trained to detect the transitions between attention/inattention states instead of states for individual time samples. Even with improvements we will end up with a less than perfect inattention detector and we will have to study user experiences in order to define levels of accuracy required before inattention detectors could be used as an acceptable driver assistant tool.

Future work should also include modeling how drivers "recover" from inattentive periods. Even with perfect eye tracking it is unlikely that when a driver's eyes return to the road that the driver is instantly attentive and aware of his environment. This is a general attention modeling problem and is not specific to our technique.

Once driver inattention is detected there still needs to be experimentation on how to best assist the driver. Good driving habits will include periods that we have defined as inattentive, such as a "blind spot" glance before changing lanes. The system must understand the appropriate frequency and duration of these "blind spot" glances and not annoy the driver by offering counter-productive or unreasonable advice.

7 Conclusion

In this chapter, we have demonstrated three instances of using computational intelligence techniques such as the random forest method in developing intelligent driver assistance systems. Random Forest appears to be a very well suited tool for massive heterogeneous data sets that are generated from the driving domain. A driver activity classifier based on Random Forests is also fast enough to run in real time in a vehicle.

First, the random forest method is used to create a semi-automatic data annotation tool for driving database creation to support data-driven approaches in the driving domain such as driving state classification. The tool significantly reduces the manual annotation effort and thus enables the user to verify automatically generated annotations, rather than annotating from scratch. In experiments going through the whole database annotation cycle, we have observed sixfold reductions in the annotation time by a human annotator [25].

Second, the random forest method is employed to identify the sensor variables that are important for determining the driving maneuvers. Different combinations of sensor variables are identified for different driving maneuvers. For example, steering wheel angle is important for determining the turning maneuver, and brake status, acceleration, and headway distance are important for determining the panic brake maneuver.

Third, our random forest technique enabled us to detect driver inattention through the use of sensors that are available in modern vehicles. We compared both traditional sensors (detecting only steering wheel angle and accelerator pedal position) and CAS sensors against the performance of a state-of-the-art eye/head tracker. As expected, the addition of CAS sensors greatly improve the ability for a system to detect inattention. Though not as accurate as eye tracking, a significant percentage of inattentive time samples could be detected by monitoring readily available sensors (including CAS sensors) and it is believed that a driver assistant system could be built to use this information to improve driver attention. The primary advantage of our system is that it requires only a small amount of code to be added to existing vehicles and avoids the cost and complexity of adding driver monitors such as eye/head trackers.

A driving simulator is an excellent tool to investigate driver inattention since this allows us to design experiments and collect data on driver behaviors that may impose a safety risk if these experiments were performed in a real vehicle. There is still much to be learned about the causes and effects of driver inattention, but even small steps toward detecting inattention can be helpful in increasing driver performance.

References

1. E.R. Boer. Behavioral entropy as an index of workload. In *Proceedings of the IEA/HFES 2000 Congress*, pp. 125–128, 2000.
2. E.R. Boer. Behavioral entropy as a measure of driving performance. In *Driver Assessment*, pp. 225–229, 2001.
3. O. Bousquet and A. Elisseeff. Algorithmic stability and generalization performance. In *Proceedings of NIPS*, pp. 196–202, 2000.
4. L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.
5. L. Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.
6. L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*, CRC Press, Boca Raton, FL, 1984.
7. D. Cohn, L. Atlas, and R. Ladner. Improving generalization with active learning. *Machine Learning*, 15(2):201–221, 1994.
8. T.A. Dingus, S.G. Klauer, V.L. Neale, A. Petersen, S.E. Lee, J. Sudweeks, M.A. Perez, J. Hankey, D. Ramsey, S. Gupta, C. Bucher, Z.R. Doerzaph, J. Jermeland, and R.R. Knipling. The 100 car naturalistic driving study: Results of the 100-car field experiment performed by Virginia tech transportation institute. Report DOT HS 810 593, National Highway Traffic Safety Administration, Washington DC, April 2006.
9. L. Fletcher, N. Apostoloff, L. Petersson, and A. Zelinsky. Driver assistance systems based on vision in and out of vehicles. In *Proceedings of IEEE Intelligent Vehicles Symposium*, pp. 322–327. IEEE, Piscataway, NJ, June 2003.
10. J. Forbes, T. Huang, K. Kanazawa, and S. Russell. The BATmobile: Towards a Bayesian automated taxi. In *Proceedings of Fourteenth International Joint Conference on Artificial Intelligence*, Montreal, Canada, 1995.
11. I. Guyon and A. Elisseeff. An introduction to feature selection. *Journal of Machine Learning Research*, 3:1157–1182, 2003.
12. I. Guyon, S. Gunn, M. Nikravesh, and L. Zadeh. *Feature Extraction, Foundations and Applications*. Springer, Berlin Heidelberg New York, 2006.
13. J. Hankey, T.A. Dingus, R.J. Hanowski, W.W. Wierwille, C.A. Monk, and M.J. Moyer. The development of a design evaluation tool and model of attention demand. Report 5/18/00, National Highway Traffic Safety Administration, Washington DC, May 18, 2000.
14. R.A. Hess and A. Modjtahedzadeh. A preview control model of driver steering behavior. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pp. 504–509, November 1989.
15. R.A. Hess and A. Modjtahedzadeh. A control theoretic model of driver steering behavior. *IEEE Control Systems Magazine*, 10(5):3–8, 1990.
16. H. Liu and H. Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer, Boston, MA, 1998.
17. J.C. McCall and M.M. Trivedi. Driver behavior and situation aware brake assistance for intelligent vehicles. *Proceedings of the IEEE*, 95(2):374–387, 2007.
18. V.L. Neale, S.G. Klauer, R.R. Knipling, T.A. Dingus, G.T. Holbrook, and A. Petersen. The 100 car naturalistic driving study: Phase 1-experimental design. Interim Report DOT HS 809 536, Department of Transportation, Washington DC, November 2002. Contract No: DTNH22-00-C-07007 by Virginia Tech Transportation Institute.
19. A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14: Proceedings of the NIPS 2001*, 2001.
20. N. Oza. Probabilistic models of driver behavior. In *Proceedings of Spatial Cognition Conference*, Berkeley, CA, 1999.
21. F.J. Pompei, T. Sharon, S.J. Buckley, and J. Kemp. An automobile-integrated system for assessing and reacting to driver cognitive load. In *Proceedings of Convergence 2002*, pp. 411–416. IEEE SAE, New York, 2002.
22. L.R. Rabiner. A tutorial on Hidden Markov Models and selected applications in speech recognition. *Proceedings of the IEEE*, 77(2):257–286, 1989.

23. D. Remboski, J. Gardner, D. Wheatley, J. Hurwitz, T. MacTavish, and R.M. Gardner. Driver performance improvement through the driver advocate: A research initiative toward automotive safety. In *Proceedings of the 2000 International Congress on Transportation Electronics, SAE P-360*, pp. 509–518, 2000.
24. B. Schölkopf and A. Smola. *Learning with Kernels.* MIT Press, Cambridge, MA, 2002.
25. C. Schreiner, K. Torkkola, M. Gardner, and K. Zhang. Using machine learning techniques to reduce data annotation time. In *Proceedings of the 50th Annual Meeting of the Human Factors and Ergonomics Society*, San Francisco, CA, October 16–20, 2006.
26. P. Smith, M. Shah, and N. da Vitoria Lobo. Adetermining driver visual attention with one camera. *IEEE Transactions on Intelligent Transportation Systems*, 4(4):205, 2003.
27. K. Torkkola. Automatic alignment of speech with phonetic transcriptions in real time. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP88)*, pp. 611–614, New York City, USA, April 11–14, 1988.
28. K. Torkkola, M. Gardner, C. Schreiner, K. Zhang, B. Leivian, and J. Summers. Sensor selection for driving state recognition. In *Proceedings of the World Congress on Computational Intelligence (WCCI), IJCNN*, pp. 9484–9489, Vancouver, Canada, June 16–21, 2006.
29. K. Torkkola, N. Massey, B. Leivian, C. Wood, J. Summers, and S. Kundalkar. Classification of critical driving events. In *Proceedings of the International Conference on Machine Learning and Applications (ICMLA)*, pp. 81–85, Los Angeles, CA, USA, June 23–24, 2003.
30. K. Torkkola, N. Massey, and C. Wood. Driver inattention detection through intelligent analysis of readily available sensors. In *Proceedings of the 7th Annual IEEE Conference on Intelligent Transportation Systems (ITSC 2004)*, pp. 326–331, Washington DC, USA, October 3–6, 2004.
31. K. Torkkola, S. Venkatesan, and H. Liu. Sensor sequence modeling for driving. In *Proceedings of the 18th International FLAIRS Conference*, AAAI Press, Clearwater Beach, FL, USA, May 15–17, 2005.
32. W. Van Winsum, M. Martens, and L. Herland. The effects of speech versus tactile driver support messages on workload, driver behaviour and user acceptance. TNO-report TM-00-C003, TNO, Soesterberg, The Netherlands, 1999.
33. C. Wood, B. Leivian, N. Massey, J. Bieker, and J. Summers. Driver advocate tool. In *Driver Assessment*, 2001.