future internet

MDPI

Article

Environmental-Based Speed Recommendation for Future Smart Cars †

Ioannis Galanis [1,*], Iraklis Anagnostopoulos [1], Priyaa Gurunathan [2] and Dona Burkard [2]

[1] Department of Electrical and Computer Engineering, Southern Illinois University, Carbondale, IL 62901, USA; iraklis.anagno@siu.edu

[2] Research & Innovation Center, Ford Motor Company, Dearborn, MI 48124, USA; pgurunat@ford.com (P.G.); dburkar1@ford.com (D.B.)

* Correspondence: ioannis.galanis@siu.edu

† This paper is an extended version of our paper published in IEEE International Symposium on Circuits and Systems (ISCAS 2018).

Received: 31 January 2019; Accepted: 20 March 2019; Published: 24 March 2019

Abstract: Modern vehicles are enhanced with increased computation, communication and sensing capabilities, providing a variety of new features that pave the way for the deployment of more sophisticated services. Specifically, smart cars employ hundreds of sensors and electronic systems in order to obtain situational and environmental information. This rapid growth of on-vehicle multi-sensor inputs along with off-vehicle data streams introduce the *smart* car era. Thus, systematic techniques for combining information provided by on- and off-vehicle car connectivity are of remarkable importance for the availability and robustness of the overall system. This paper presents a new method to employ service oriented agents that cohesively align on- and off-vehicle information in order to estimate the current status of the car. In particular, this work combines, integrates, and evaluates multiple information sources targeting future smart cars. Specifically, the proposed methodology leverages weather-based, on-route, and on-vehicle information. As a use case, the presented work informs the driver about the recommended speed that the car should adapt to, based on the current status of the car. It also validates the proposed speed with real-time vehicular measurements.

Keywords: smart cars; environmental information; speed recommendation; points of interest

1. Introduction

Modern high-end vehicles are equipped with up to 100 embedded Electronic Control Units (ECUs), each executing standard services with pre-defined inputs [1,2]. Advanced driver-assistant systems (ADAS) and multimedia applications are some examples of services that are realized by running complex algorithms on embedded ECUs [3]. However, those pre-specified functions cannot be enhanced throughout the traditional vehicle architecture, since it is very expensive to develop new services on the current static vehicle architecture [4]. For that reason, the current design has become an obstacle towards the evolution of the functionality in the car, and thus modern vehicles are shifting from the traditional distributed hardware modules with fixed non-scalable functionality to a more robust distributed multi-agent based architecture.

Moreover, modern vehicles integrate and exploit a large number of sensors, in order to acquire an accurate representation of the environment and automate several driving tasks, that were traditionally handled by humans [1]. For instance, radar, lidar, GPS, are some of the types of sensors which are deployed in modern vehicles and create input streams that are processed by the ECUs when running complex algorithms for sophisticated services, such as cruise control or forward-collision warning [5,6].

Vehicles are also equipped with visual sensors which generate a huge amount of visual data with rich and resourceful content [7]. This data is streamed to a variety of heterogeneous computing elements in order to create a precise representation of the surrounding environment of the car [8]. This increased amount of information requires innovative and efficient off-loading techniques for information extraction [9]. All these advantages in the functionality of modern vehicles has established the term *smart cars*.

Apart from the increased computing and sensing capabilities that allow smart cars to process input data from multiple sources, they present growing connectivity capabilities as well [10]. Modern vehicles are connected over the Internet and cooperate with other vehicles or road-side units. A variety of technologies and different communication protocols are employed to serve different application contexts, such as vehicle safety, comfort, and entertainment. Specifically, connected vehicles are linked to the cloud, to other vehicles or even to road-side infrastructure utilizing wireless technologies (WiFi, 4G, LTE, etc.) [11]. The impact of vehicle connectivity is considered crucial in terms of vehicle safety and, for that reason, the National Highway Traffic Safety Administration (NHTSA) of the USA expects that every new car will be equipped with connectivity capabilities after year 2020 [12]. Thus, smart cars are becoming the most sophisticated entity in the Internet of Things (IoT) era, hence they prevail as a replacement to modern cars introducing the era of *Internet of Vehicles* [13–15].

Increased vehicle connectivity brings major challenges to the automotive research engineering groups that try to utilize all the available information sources efficiently. Existing approaches focus on processing data from single information sources, while their outcome is strongly affected by the information quality. A stand-alone information source or sensor cannot overcome certain physical limitations, i.e., the limited sensor range or the field of view. For instance, camera systems have difficulty in detecting bright objects when the camera sensor is blinded by the sun, whereas a radar system is able to detect other vehicles or obstacles in the same situation. As a result, data are susceptible to imprecision and inconsistency [14,16] and making single information source approaches inefficient and error prone, as they can lead to imprecise results. Therefore, it becomes prominent to abandon traditional single source approaches and move on multi source strategies.

In the context of smart cars era, the available computation and connection capabilities can support the development of multi-source framework that combines heterogeneous information sources. The main objective of a multi-source framework that targets automotive domain is to increase driver's awareness about any possible danger in the observed environment. In other words, improve the coverage of the observation area and provide a better knowledge and perception of the environment of the vehicle, in order to make the driving experience safer, more comfortable and more efficient. In order to facilitate development of more sophisticated applications for road safety, infotainment, smart and green transportation, it is necessary to combine multiple heterogeneous sources of information [17]. This new automotive era is realized with increasing communication capabilities of future cars (wireless connectivity) and transforms the current static perception of vehicles to an intelligent entity that dynamically interacts with its rapidly changing environment [18,19]. Thus, future intelligent transportation systems will combine different sources of information and create an enhanced environmental awareness of the vehicle, offering an advanced resourceful travel experience for the passengers/motorists.

In this paper, we present a systematic methodology to cohesively align information provided by off- and on-vehicle sources. Specifically, the proposed approach utilizes enhanced connectivity of modern vehicles to gather data about weather, on-route and vehicular information and it manages to create a more accurate perception of the surrounding environment of future smart cars. As a use case, the presented methodology combines information from multiple heterogeneous sources and alerts the driver about the recommended speed that the car should follow, so that it can avoid dangerous driving. Compared to our original work presented in [14], the differentiators of the presented methodology are the following:

- We further explain the details of our original implementation in [14], providing more details about the data analysis (Section 3.2). We also evaluate and compare the presented classification model against existing techniques and we present the results in Section 4.4.
- We introduce a new feature that utilizes route-based information and extracts specific intermediate way-points that dictate change in vehicle's speed.
- We utilize vehicle data as a new information source in order to calculate the mismatch between the theoretical engine speed and the actual engine speed of the vehicle. Such mismatch may cause the trigger of advanced driver-assistance systems (e.g., ABS, ESC).
- We validate weather- and route-based information with real-time vehicle data and inform the driver about a recommended (dynamic) speed that the car should adapt to, based on the current road status.
- We developed the overall service on top of Hydra [20], a distributed multi-agent computing framework that targets smart cars and it provides self-management functions such as in-car workload balancing and failure recovery. Section 4.1 further explains the implementation presented in [14] and enhances this work with more technical details regarding the development of the proposed methodology on real computing boards by utilizing the computation infrastructure in [20].

It is worth mentioning that, since such sophisticated services require flexible computing architectures, the proposed methodology was developed as a service layer on top of distributed computing framework for smart cars (Hydra [20]). However, Hydra was designed in order to provide self-management functionality and to support services as a box for modern vehicles improving scalability. The main differences between the proposed work and Hydra [20] are summarized as follows:

1. Hydra is a generic framework that supports the deployment of a variety of applications, and several automotive benchmarks were tested in [20]. However, in this paper, we present a new methodology that we deploy on Hydra for the first time.
2. The proposed methodology includes several features that were developed on top of Hydra, which enhances Hydra's functionality. Specifically, the presented methodology:
   (a) gathers environmental information (weather and on-route), in order to create a better understanding about the physical characteristics of the road, and
   (b) it combines this information with on-vehicle data in order to suggest a speed limit so as the vehicle achieves a targeted road surface index and reduces the danger of an accident (e.g., loss of traction).

2. Related Work

Previous research works have introduced methodologies that try to combine multiple sources of information, taking advantage of the increased sensor and computing capabilities on modern vehicles, so that they can achieve a more accurate representation of the surrounding environment. The ultimate goal is to make the best decision according to the current circumstances, in order to improve safety and comfort of vehicular experiences for the passengers.

Many research works have focused on the integration of multiple information sources in order to improve passenger safety. Authors in [17] envision the future smart car as an intelligent vehicle that can interact with various information sources (on-board sensors, road side infrastructure or Internet). Authors in [21] propose an optimization scheme that tries to optimize vehicle's trajectory and thus optimize the efficiency of the driver. They manage to minimize vehicle's fuel consumption by calculating the optimal velocity trajectory. Authors in [22] utilize information from multiple sources through increased vehicle communication capabilities (vehicle-to-vehicle and vehicle-to-infrastructure) in order to provide a more efficient driving style. Researchers in [23] aim at increasing vehicle's

perception by cooperatively utilizing information from other vehicles or infrastructure, through wireless connectivity. Based on that enhanced perception, the car can provide augmented situational awareness for a safer and more efficient vehicular experience for the passenger. Authors in [24] try to increase passenger safety by detecting events that indicate dangerous driving, combining on-vehicle sensor information with cloud information, while the authors in [25] present a methodology that aims to enhance vehicle's perception of the surrounding environment in order to detect dangerous parts of the road. Their approach combines on-vehicle measurements with wireless sensor inputs from Road Side Units (RSUs) and other vehicles to calculate the confidence of a possible road hazard. Additionally, the authors in [14] utilize weather information to provide a recommended vehicle speed. In [26], a multi-source framework is presented that performs real-time analysis in order to detect dangerous parts on the road surface. The research team in [27] proposes a novel methodology that targets the increasing of environmental perception and knowledge of the vehicle by combining on-vehicle, on-route and visual information. Authors in [28] propose a new system that incorporates cloud-based services, on-vehicle sensor data and information from roadside infrastructure, targeting the enhancement of vehicle's safety and identifying an efficient vehicle trajectory. Authors in [29] propose a deep learning-based architecture that merges data from multiple inputs, combining on-route information, visual information and vehicular measurements that capture driving environment from multiple points of view. Other research teams that target intelligent transportation systems manage to detect dangerous events on the road by mixing on-vehicle measurements and metrics that are gathered from Road Side Units (RSUs), using wireless sensors [30].

Moreover, many research works focus on utilizing multiple sources in order to make the passenger's experience more comfortable. For example, automated driving is one of the main characteristics of future smart cars. Authors in [31] combine different sources of information, such as localization information (via GPS) and road side units, in order to automate driving tasks and achieve enhanced passenger comfort. In [32], it is stated that future cars will incorporate enhanced connectivity capabilities with both the infrastructure (through Cloud) and other vehicles, so that future cars can offer advanced entertainment and cognitive services during travel time. Authors in [33] take advantage of the advanced wireless capabilities of modern cars and they analyze real-time traffic data in order to estimate local traffic density, so as to provide a more efficient and comfortable driving travel. Authors in [34] are able to increase intelligent vehicle's perception of the surrounding environment. They detect and classify moving objects on the street with high accuracy, by performing combinational processing of data from heterogeneous visual sensors. Other researchers [35] utilize wireless networking, e.g., vehicle-to-vehicle and vehicle-to-infrastructure communication, in order to alleviate traffic congestion and enhance comfort during a vehicle trip. Authors in [36] study a new coordinated system that would facilitate vehicle traffic in critical conditions, i.e., crossing a specific intersection, in order to achieve the highest traffic efficiency.

Regarding traffic management at crossroads with Vehilce-to-Vehicle (V2V) communication, the authors in [37] assume that vehicles are connected and adapt their speed according to other vehicles (in a cooperative way), while the authors in [38] present a multi-agent model to manage the vehicles in platoon. As described in [39,40], variable speed limit control is an alternative method to mitigate congestion and improve traffic operations (e.g., freeway bottlenecks). However, these approaches focus on improving the quality of the driving experience by reducing the travel time. The proposed work utilizes the connectivity of modern vehicles to gather data about weather, on-route and vehicular information in order to create a better understanding about the physical characteristics of the environment and how the vehicle should adapt. Additionally, since such sophisticated services require flexible computing architectures, the proposed methodology was developed as a service layer on top of distributed computing framework (Hydra [20]) for smart cars.

3. Proposed Approach

The goal of this paper is to combine, integrate and evaluate multiple information sources for modern vehicles. The proposed approach utilizes the increased connectivity of modern cars and combines it with on-vehicle data. Specifically, weather- and route-based information sources are merged with real-time vehicle data in order to obtain an estimation of the status of the road. As a use case, the proposed approach focuses on providing a recommended speed to the driver, based on the weather status estimation and extracted information throughout the car's route.

3.1. Overview

Overall, the proposed methodology considers a vehicle as a system with N number of input sources and $S = \{s_1, s_2, ..., s_N\}$ is the set of the available sources. Each source s_i monitors and generates $y_i = [v_1, v_2, \cdots v_N]$ values that form the initial set of data for the system: $Y = [y_1, y_2, ..., y_N]^T$. The next step is to rank and identify which inputs actually have an effect on the targeted metric. Data is filtered by a ranking function that selects the necessary portion of data for the specific application $Y \xrightarrow{\mathcal{F}} Y'$ and thus produces a new data set $Y' = [y_1^{'}, y_2^{'}, ..., y_N^{'}]^T$. The purpose of this pruning procedure is to eliminate any redundant inputs in order to drop the dimensionality of the input matrix and achieve lower complexity of the design. After selecting the appropriate data, the next step is to extract the desired features. A function $G(\cdot)$ gets as input the output $y_i^{'}$ of the previous step and produces a set of estimations:

$$D = G(Y^{'}) = \{g_1(y_1^{'}), g_2(y_2^{'}), ..., g_N(y_N^{'})\}. \quad (1)$$

Based on the targeted metric, the function $G(\cdot)$ can (i) be a well defined formula; or (ii) an approximation depending on the complexity of the solution. In particular, the former case includes a mathematical representation of the system which incorporates a physical understanding of the functionality and the monitoring scheme. For example, on-vehicle sensors and monitoring ECUs can provide real-time information regarding the engine speed, the vehicle speed, the transmission gear position, etc. All of these values can be used to calculate various vehicle dynamics metrics. In the latter case, the system representation might not be available or may be too complex. However, it is possible to monitor the system and gather corresponding data values. In this case, the desired metric is calculated based on estimations derived from the monitored data. However, the critical part in this step is the quality of the data set. Generally, the confidence of the estimation increases as the data set includes more elements and the number of outliers is reduced. It should be noted that the initial model of the Equation (1) describes the general formulation of the problem, which is applicable to any number of information sources. In particular, the problem of combining N different information sources is tackled from the software perspective, as the presented methodology is developed on top of Hydra, a multi-agent distributed framework that targets smart cars [20].

Figure 1 depicts the overview of the proposed approach. In this paper, we utilized proprietary framework provided by Ford, regarding off-vehicle connectivity—cloud and Road Side Units (RSUs)—to retrieve weather data for a specific test-route and estimate the status of the road. Based on this estimation, a calculation of Road Surface Index (RSI) is performed and a recommended speed is provided. The estimation of road status based on weather information requires an approximate solution as it is too complicated to solve by a single mathematical formula. Thus, the goal is to create a data set with multiple observations throughout the year and correlate the different weather attributes with on-road surface status. Additionally, the proposed framework employs route-based information sources in order to capture specific intermediate way-points that dictate changes in the vehicle's speed. These points are referred to as *Points of Interest* as they can provide useful and important information that can affect the nominal speed of the vehicle. Points of Interest are also based on vehicle's physical characteristics (weight, size, etc.) and road structure (bridges, turns, etc.) as well. Moreover, by utilizing the on-vehicle data, we validate the recommended speed by calculating the

mismatch between the theoretical value of the engine speed and the actual value. Such mismatch may cause the trigger of advanced driver-assistance systems (e.g., ABS, ESC).

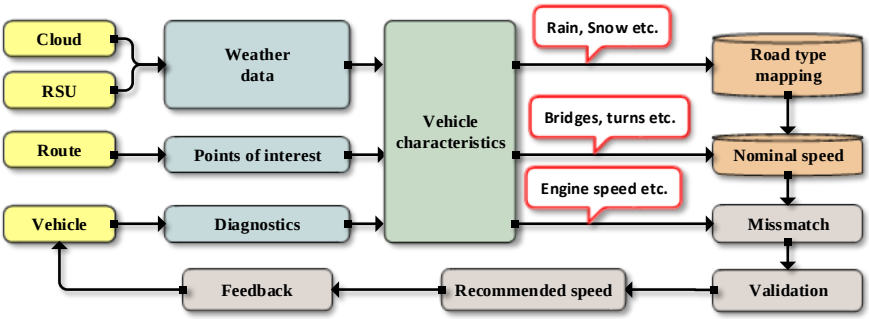


Figure 1. Overview of the proposed approach.

3.2. Road Status Estimation Based on Weather Information

Weather conditions can directly affect the behavior of the vehicle on the road. For example, driving on a rainy day can affect the stopping distance of a car, or it can influence the way a driver performs a maneuver. It is more likely that a driver can lose control of the car, if there's water or snow or ice on a road surface. In order to quantify the effect of weather conditions on the road, we utilize the concept of Road Surface Index (RSI), as presented in [41]. RSI is defined as a quantitative relation between the weather condition and the friction level on the road and describes the different surface condition categories. RSI depends on the road status which is affected by weather characteristics. RSI values vary from 0.1 (minimum value—ice covered) to 1.0 (maximum value—bare and dry surface). In the proposed work, we modify the approach in [41] and we identify four *road classes* regarding the status of the road: (i) Dry; (ii) Wet; (iii) Snow; and (iv) Ice. Accordingly, the RSI values for each class are: (i) $[0.75 - 1.0]$, (ii) $[0.25 - 0.83]$, (iii) $[0.15 - 0.25]$, (iv) $[0.1 - 0.2]$.

In order to estimate the status of the road surface, we utilized information from cloud and RSU services. However, the calculation of RSI as a closed formula based on weather information is inherently complex as it is a multi-dimensional and stochastic problem. In fact, the actual value of RSI depends on a variety of factors and the accurate calculation of RSI is not in the scope of this work. As a result, we utilize an RSI-speed correlation presented in [42] that estimates the value of RSI with respect to speed while avoiding direct calculation of RSI.

3.2.1. Data Analysis for Road Condition Estimation

Weather services provide a plethora of metrics that cover a variety of weather attributes. However, not all of these metrics affect road class in the same way. For that reason, in this work, we built a prediction model that is trained on gathered weather metrics and estimates the road status. Weather data and road condition information were gathered by cloud and Road Side Units (RSUs) and weather services for a period of one year covering various areas in the United States of America. Based on the desired location, we receive the current weather conditions at that particular location. All the values of the available weather attributes (temperature, humidity, precipitation and others) are recorded and further analyzed, as it is described in the following section. All of the gathered values are stored and form the training set D that is utilized in the experiments.

In order to identify which weather parameters affect the road condition, we develop a classification tree that takes as input a set of attributes and decides in which class this input belongs to. Reducing the number of input attributes can drop the complexity of the classifier and also eliminate redundant information.

In order to measure how pure are the subsets, we use *Impurity* as a metric that quantifies how well the classes of our dataset are separated. **Example:** In order to define Impurity, we utilize the notation of "positive" and "negative" class. As an illustrative example, we consider a dataset that

contains two classes: one positive and one negative. The terms are conventional and the term "positive" corresponds to the important rare data, while "negative" corresponds to the non-important one. Let N_{pos} and N_{neg} denote the number of training samples that belong to the positive and negative classes at node x, respectively. Moreover, N_{total} represents the total samples that belong to node x. We wish to find the attribute that splits the dataset D into the purest subsets. Let

$$p_{pos} = \frac{N_{pos}(node\_x)}{N_{total}(node\_x)}, \quad p_{neg} = \frac{N_{neg}(node\_x)}{N_{total}(node\_x)} \tag{2}$$

be the fraction of the positive and negative training samples that belong to the positive and negative class, respectively, in a given node x. Thus, Impurity is a number between [0, 1] and it is not affected by the number of samples. Impurity takes its maximum value when the number of positive and negative samples are equal: $N_{pos} = N_{neg} = \frac{N_{total}}{2} \Longrightarrow p_{pos} = p_{neg} = 0.5$. In this example, we use the metric *Entropy* to calculate Impurity at a particular node x:

$$Entropy(node\_x) = -(p_{pos} \cdot log_2(p_{pos}) + p_{neg} \cdot log_2(p_{neg})). \tag{3}$$

In a general case where training data can belong to any of j number of classes, we consider an attribute i that splits the set D into l subsets $D_1, D_2, \cdots D_l$. We calculate the Impurity of the dataset D utilizing the metric *Entropy* as follows:

$$Entropy(D, i) = -\sum_{k=1}^{l} p(k) \cdot log_2 p(k), \tag{4}$$

where $p_k = \frac{|D_k|}{|D|}$, $k = 1, 2, \cdots l$, is the ratio of training examples classified as class j in each subset D_k. We utilized different attributes to split the dataset and we observed different Impurity measures. Therefore, not all the attributes affect the estimation of the road class in the same way. In order to decide which are the most important features, we utilized the concept of *Information Gain*. Information Gain (IG) is a metric that marks how important a given attribute of the feature vectors is. Assuming that D is the gathered training dataset with N attributes, the IG of splitting the dataset with the attribute i is calculated as:

$$IG(D, i) = Entropy(D) - Entropy(D, i) = Entropy(D) - \sum_{k=1}^{l} \frac{|D_k|}{|D|} \cdot Entropy(D_k), \tag{5}$$

where $Entropy(D)$ is the entropy of the original dataset and $Entropy(D, i)$ is the average entropy of the produced subsets $D_1, D_2, \cdots D_l$. From all the available attributes, we select the one that maximizes the difference of Equation (5). It should be noted that maximizing the IG is equivalent to minimizing the average entropy, since the term $Entropy(D)$ has the same value for all attributes.

The next step is to rank the attributes and decide which of them among the given set of training feature vectors are most useful for predicting the road status among the four classes (Dry, Wet, Snow and Ice). Table 1 presents the values of IG for all the weather attributes gathered throughout the year. Specifically, Table 1 presents that the most important information is provided by temperature, whereas the least important comes from gust wind. Based on the IG values, we choose to keep the first six weather attributes.

Then, we build the desired classification tree, which takes as input all the weather attributes and estimates the road class (Dry, Wet, Snow and Ice). We reduce the number of attributes that the classification tree uses based on the calculated IG. In this way, we drop the complexity of the classification and training, eliminate redundant information, and avoid over-fitting. The creation of the classification tree is presented in Algorithm 1. Given the input data set D and the pruned set of attributes A, the classification tree checks if the set D is homogeneous (lines 5–7). This is

true if the majority of the instances of the current set belongs to a single class. If the initial data set D is not homogeneous, then, for each $a \in A$, the original set D is split into subsets $D_{1,2,3,\ldots l}$. The algorithm selects the split that gives the most pure subsets, using the concept of Impurity, which is the corresponding decision boundary that terminates the recursive split of the original data set. The classification tree splits the k^{th} subset D_k into l sub-sets (line 10) and selects the attribute that produces the minimum Impurity (lines 11–16). This procedure is repeated recursively until the *majority* is reached (lines 21–23), which is the minimum purity threshold that is accepted for a node. Unless this threshold is satisfied, the algorithm continues to split current set into sub sets, until the *max_depth* is reached (lines 17–18).

Table 1. Information gain of input attributes.

Attributes	Information Gain
Temperature (C)	0.690
Dew Point	0.415
Wind chill	0.414
Humidity (mm)	0.308
Elevation (m)	0.289
Precipitation (mm)	0.283
Snow	0.147
Precipitation history (mm)	0.120
Rain	0.092
Snowfall(mm)	0.043
Snow depth (mm)	0.030
Gust Wind	0.013

Algorithm 1 Classification Tree Algorithm.

1: **Input:** *data* D, *set of attributes* A, *majority, max_depth*
2: **Output:** *tree* T *with labeled levels*
3: $depth \leftarrow 0$
4: **GrowTree(D,A,max_level,majority,max_depth)**
5: **if** $Homogeneous(D, majority)$ **then**
6: **return** Label(D)
7: **end if**
8: **for** *each i* **do**
9: $I_{min} \leftarrow 1$
10: Split D into subsets $D_1, D_2, ..., D_l$
11: **for** *each* $a \in A$ **do**
12: **if** $Impurity(\{D_1, ..., D_l\}) < I_{min}$ **then**
13: $I_{min} \leftarrow Impurity(\{D_1, ..., D_l\})$
14: $a_{best} \leftarrow a$
15: **end if**
16: **end for**
17: **if** $depth > max\_depth$ **then**
18: **break**
19: **else if** $D_i \neq \emptyset$ **then**
20: $T_i \leftarrow GrowTree(D_i, A, majority, max\_depth)$
21: **else**
22: $depth \leftarrow depth + 1$
23: T_i *is a leaf with Label*(D)
24: **end if**
25: **end for**
26: **end**
27: **return** tree with root S and children T_i

3.2.2. Stages of Execution

At this point, we would like to clarify the procedure for creating the classification tree and utilizing it in a vehicle. Specifically, three steps are required:

Collection of data: The first step focuses on the collection of data. Weather data and road condition information were gathered by RSUs and weather services for a period of one year covering various areas in the United States of America.

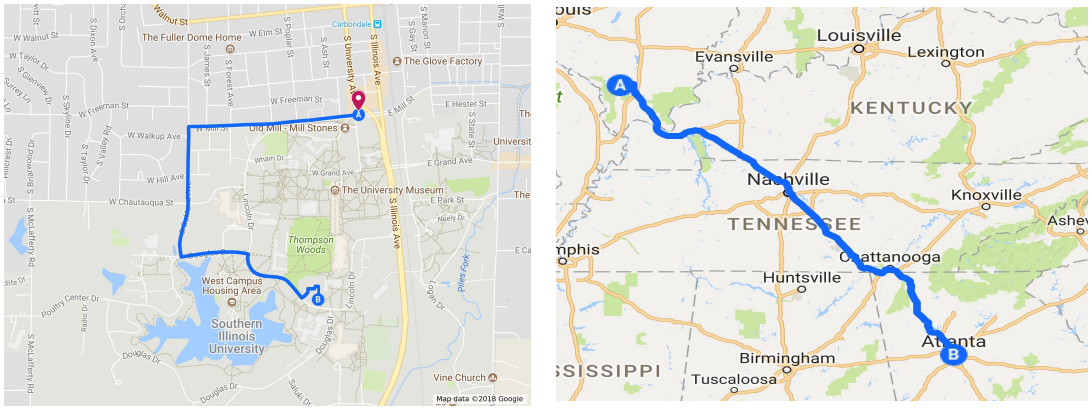
Training on cloud: The second step focuses on the training and the construction of the classifier (classification tree). This step is performed on the cloud (as it is computationally intensive) and it is executed only once. Algorithm 1 presents the algorithmic steps that are needed in order to create the classification tree based on a given dataset.

Inference on vehicle: After the training of the classifier is completed, the classifier is sent back to the vehicle. Then, the vehicle utilizes it in real time to estimate the weather condition without need for cloud communication, which would add significant delay. This action is called inference. Inputs to the classifier are the attributes from the RSUs and other weather based services.

3.3. On-Route Information

As shown previously, the proposed approach can estimate the road class solely based on information about the weather conditions. However, weather conditions alone cannot provide any additional information about physical characteristics of the road. For that reason, on-route information is introduced as a new source of information that offers knowledge about the structure of the road.

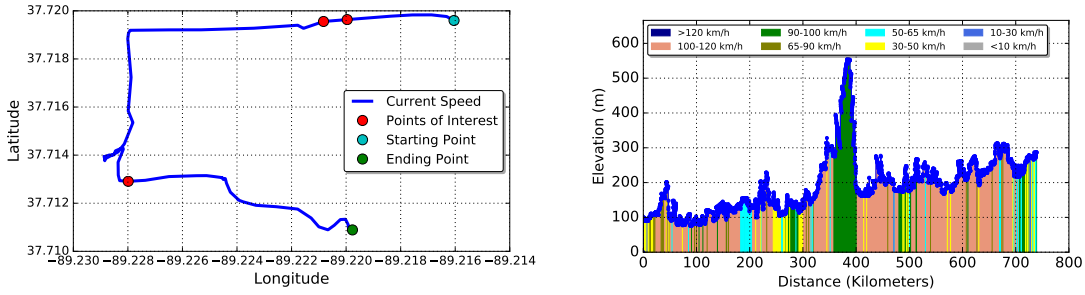
A vehicle connects to a cloud service and, for each intermediate point of the route, it transmits the current geographical coordinates. Then, it receives a variety of attributes concerning road characteristics: speed limit, soft or hard turns, highway exits, bridges, etc. The information about the nominal speed is classified into eight regions: (i) R_0: >120 km/h; (ii) R_1: 100–120 km/h; (iii) R_2: 90–100 km/h; (iv) R_3: 65–90 km/h; (v) R_4: 50–65 km/h; (vi) R_5: 30–50 km/h; (vii) R_6: 10– 30 km/h; and (viii) R_7: $<$ 10 km/h. Additionally, from on-route information, we extract intermediate way points where the car needs to reduce its speed. At these points, the driver can potentially experience loss of vehicle control. We define as *Points of Interest (PoI)* the intermediate points of a route that force a vehicle to change its speed, considering the road characteristics and the vehicle dynamics. In that sense, PoIs include every single point that nominal speed changes: turns, ramps, highway exits, including the vehicle type characteristics as well. For instance, a turn can be light or sharp, so that the nominal speed to pass over that turn will be slightly different for different types of vehicles (e.g., trucks or a passenger car). In order to be comprehensive, on-route information is extracted for both urban and highway drive scenarios. As an example, consider that a vehicle starts its route from a point A and finishes at point B. The proposed framework captures the PoIs and notifies the driver to adjust the speed of the car according to the nominal speed. Figure 2a shows a scenario of utilizing on-route information during an urban trip. The route that is shown in Figure 2a depicts a route in Southern Illinois University Campus in Carbondale, IL. Figure 3a illustrates the *PoI* that were captured for the presented urban drive scenario (red bullets). In fact, for the presented route, three PoIs were captured: (i) a traffic light; (ii) a stop sign; and (iii) a pedestrian crosswalk. It is evident that all the aforementioned points are common characteristics of urban driving and require a significant change in vehicle speed.



(**a**) Urban driving route (**b**) Highway driving scenario

Figure 2. Route information in an urban and highway driving scenario.

Another useful piece of information that can be retrieved from on-route information is the elevation of the current position of a vehicle. Based on the longitude and the latitude of each intermediate point of a route, we can retrieve elevation information for that particular point. Elevation can be useful to extract more PoIs, since it can be a factor which changes the nominal speed of the vehicle, due to special weather conditions or road characteristics (road grade). For example, it is more likely to get snow or ice in higher elevations due to local weather characteristics. It can also contribute to extracting the incline, since a sudden change in the road's grade can cause a major change in speed. Figure 2b depicts a highway driving scenario. The route starts from Carbondale, IL, USA and ends at Atlanta, GA, USA. For that given route, the nominal speed limits and the Points of Interest are determined. Since this route is much longer than the urban scenario presented in Figure 2a, it is worth depicting the Points of Interest in correlation with the elevation. In fact, for that particular route, the terrain changes drastically, as it is illustrated in Figure 3b. Since most of the route consists of highway and freeway segments, the recommended speed for the majority of the trip is the highway speed limit. However, it is worth mentioning that in Figure 3b we can observe many color changes, which reflect the recommended speed. For example, there is a sudden drop in elevation at the 250th kilometer of the trip. At this point, speed category decreases from R_1 to R_5. Additionally, when the road elevation is experienced suddenly (approximately at the 370th kilometer of the trip), the speed category decreases from R_1 to R_2.



(**a**) Points of Interest in an urban driving scenario (**b**) Points of Interest in a highway driving scenario

Figure 3. Points of Interest extracted from on-route information in an urban and a highway driving scenario.

3.4. On-Vehicle Information

The last step of the proposed methodology is the integration of vehicle-based information sources. Vehicle signals can be accessed through the Controller Area Network (CAN bus) standard, which is considered the dominant solution for transferring information between Electronic Control Units (ECUs) in vehicles [43]. Specifically, the CAN bus is a message-based communication protocol designed to

allow ECUs and devices to communicate with each other in applications without a host computer (CAN standard ISO 11898) [44]. ECUs are micro-controllers that are customized both on software and hardware, each handling pre-specified inputs and services, such as navigation, infotainment, cruise control and others. In order to access the CAN bus signals and retrieve the appropriate information, we utilized a Ford proprietary data acquisition tool. The tool gathers on-vehicle data that provides information on vehicle stability, position and speed. The first step is to identify which signals can be used in order to detect any change regarding the road class. We categorize the available data in groups and we monitor the values of on-vehicle data providing stability and speed of vehicle. The goal is to detect an engine speed mismatch that reflects a change of the road surface class. Under dry road conditions, there is a theoretical value that connects the engine and the vehicle speed. There are several parameters that affect the theoretical nominal engine speed of the vehicle under dry road conditions (axle ratio, transmission rate, tire diameter, engine load). Thus, we utilize the difference of the nominal engine speed with the current engine speed (engine speed mismatch), in order to estimate the condition of the road surface.

4. Experimental Results

4.1. System Overview

The proposed methodology was developed and deployed on top of a distributed agent-based computing deployment for smart cars, named Hydra [20]. The developed approach was chosen as the system infrastructure because it employs a fast and lightweight mechanism for software oriented agents. Each agent runs Hydra as a software service, which manipulates the execution of running tasks and performs self-optimization functions regarding optimal on-vehicle resource utilization. Hydra employs four basic modules: (1) Agent discovery and recovery; (2) Monitoring; (3) Prediction and (4) Election.

Regarding the communication of the software agents inside the vehicle, Hydra exploits the agent discovery and recovery module, which is responsible to separately find out the existence of other agents. The software agents announce their presence and services in the vehicle's internal network. That way, agent-to-agent connection is established and then every agent is aware of the presence of the following neighbor, forming a cyclic ring. However, this ring can be interrupted if an agent goes offline and is not accessible anymore. To address this issue, Hydra sends and receives messages between agents that contain useful information about their status. There are four types of messages: `ANNOUNCE`, `UPDATE`, `BID` and `BID WIN` and each message has tow corresponding actions: `SET` and `GET`. The messages serve the purpose of establishing a new connection in a preexisting agent ring and update the neighbor agent with the current status. Agent i sends a series of update messages to agent $i+1$ and, if that fails, then agent i considers agent $i+1$ offline, abandons the connection and re-connects with the next available agent. Messages also facilitate the bidding process, in case a task needs to be offloaded to another agent. In summary, the messages that Hydra is exploiting are presented in Table 2. The communication layer was developed on top of Dlib [45] and ZeroMQ [46] libraries.

When an agent is running Hydra, one or more tasks are executed on the agent. However, the configuration of each task is actually unknown initially, and needs to be monitored at run time. Hence, after the initialization of the agents and the establishment of the communication, each agent starts its monitoring manager. Every agent that is running Hydra conducts the same monitoring process locally, sampling the status of their resources periodically. For that reason, monitoring module defines a sampling period in order to get a sufficient number of updates about the current resource usage. Based on the data that are gathered through the local monitoring procedure, run-time optimizations can be performed, based on a desired objective.

In case there is a resource constraint, Hydra employs a prediction scheme that projects the resource utilization in the next sampling period. This mechanism is based on the transition matrix concept, where the status of each resource utilization is described by a set of predefined $m > 1$ states

$\{s_1, s_2, ..., s_m\}$, so as to build a transition probability matrix $M = M(s_i, s_j)$, where $M = M(s_i, s_j) = Pr(transition\, s_i \rightarrow s_j)$.

If the predicted utilization is above the desired threshold, then Hydra triggers the election mechanism. The agent that detects the threshold violation communicates with the rest of agents in order to initiate the election process. It requests from all the available agents to send their bids and selects the winner according to which one has the highest resource availability. Then, it announces the winner of the bid and migrates one or more applications to the corresponding agent. The same procedure is followed in case there is an agent failure, so that the workload that was running on that agent is shared among its neighbors. Thus, from the user's perspective, the system's functionality is preserved and the running workload remains steady across the multi-agent system.

Table 2. Inter-agent communication messages in Hydra.

		Message Type			
		ANNOUNCE	**UPDATE**	**BID**	**WIN BID**
Action type	**SET**	New agent appears	Require update	Ask for bid	Announce winner
	GET	Receive information	Send update	Get bids	Assign workload to the winner

4.2. Application Evaluation

In order to evaluate our approach, we tested the proposed methodology on real Ford vehicles. The experimental set up included Hydra, a distributed multi-agent framework [20], which was running the task of road class estimation based on weather information and simultaneously it was collecting data from the vehicle by using a Ford proprietary data acquisition tool. In order to evaluate and validate the proposed framework, we launched two different scenarios. Scenario 1 was performed in an urban driving environment, so as to evaluate the proposed methodology in a realistic environment. In Scenario 1, a test vehicle was used around the Carbondale, IL, USA area and Scenario 2 was performed in Ford test track field in Dearborn, MI, USA where an experienced and professional driver was responsible for all the actions. The purpose of this scenario is to push the car to its limits in order to evaluate the on-vehicle information that the car could provide. For the previous experiments, we utilized three agents that were realized by three *armv7l* boards running the following tasks: (i) Board 1 was running the process of road status estimation based on weather information; (ii) Board 2 was aggregating the estimations from weather- and on-route-based estimations; and (iii) Board 3 was collecting vehicle data at run-time.

In order to evaluate the proposed road class estimation scheme, the accuracy of the developed classification tree was measured by performing cross validation on the data that was captured during a period of one year covering various areas in the United States of America. Regarding the parameters of Algorithm 1, we set $max\_depth = 5$, $majority = 95\%$ and we utilized the six attributes with the highest information gain (Table 1). We chose these values as we noticed that a greater value for max_depth was resulting in data over-fitting, increasing also the classification time. Moreover, lower values of $majority$ resulted in more outliers. Figure 4 presents the classification tree that was built in order to classify the road status into four distinct classes. Based on the method described in Section 3.2, the original dataset D is split into subsets by the attribute that maximizes the IG. This operation is repeated recursively, until a pure node is acquired (majority = 95%) or the maximum depth of the tree has been reached. It is noted that the presented algorithm is executed as a cloud service, as there are significant advantages in terms of performance, compared to an on-vehicle execution. This happens due to (i) higher storage availability and (ii) computation capabilities of the cloud.

We evaluate the performance of the presented classification tree and we illustrate the results using the concept of the *confusion matrix*. A confusion matrix is a table that is used to describe the

performance of a classification model on a set of test data for which the true values are known [47]. Each row of the matrix represents the instances of the actual class while each column represents the instances of the predicted class. As depicted in Table 3, which presents the accuracy of the developed classification tree (confusion matrix), 90.2% of the *Dry* data was classified correctly while 1.8% of them was misclassified as *Ice*. Additionally, the classifier achieved the smaller accuracy while predicting the *Snow* and *Ice* road classes. The misclassified inputs are justified by the imperfections of the classifier we are using and of course by the problem of distinguishing different road class from each other. For instance, it is explained in [48] that different road classes can co-exist on the same road section and, for this reason, it is inherently very hard to completely separate them from each other.

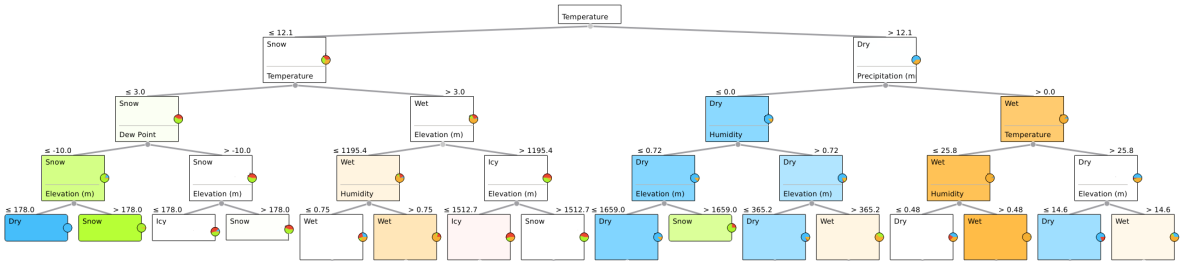


Figure 4. An instance of the classification tree with majority set to 95% and depth equal to 5.

Table 3. Confusion matrix.

		Predicted			
		Dry	**Wet**	**Snow**	**Ice**
Actual	**Dry**	90.2%	4.0%	1.4%	1.7%
	Wet	6.8%	90.2%	4.7%	6.7%
	Snow	1.1%	2.2%	82.0%	16.3%
	Ice	1.8%	3.6%	11.9%	75.3%

4.3. Evaluation Scenarios

4.3.1. Urban Driving Scenario

In Scenario 1, a test vehicle was used around the Carbondale, IL, USA area. Figure 5 illustrates the data we acquired utilizing only static on-route information in order to estimate the desired speed. The blue line depicts the speed of the vehicle during the experiment, whereas the red line reflects the recommended speed that the on-route information indicates, based on the extracted PoIs. However, on-route estimation depends on the starting and ending point of the trip and does not take into account the possible dynamic changes of the surrounding environment (weather conditions, actual road surface status).

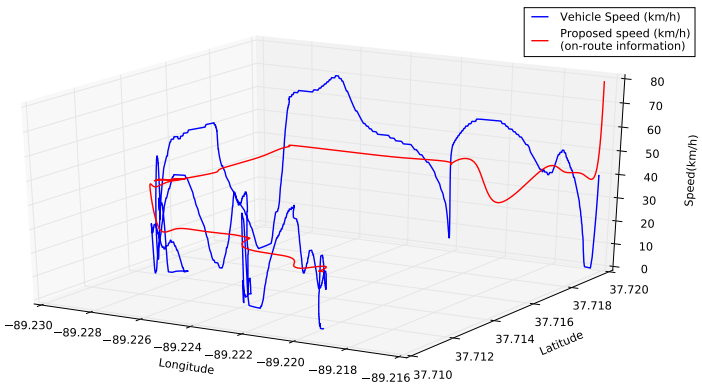
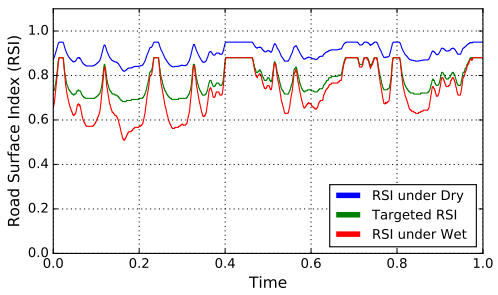


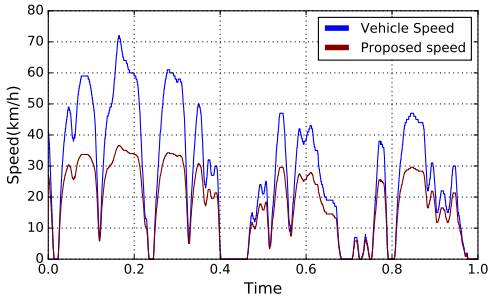
Figure 5. On-route based speed estimation.

In Scenario 2, we replicated the same urban driving scenario, but we integrated the road class estimation based on the performed weather classification analysis. The blue line in Figure 6a represents the RSI of the car, based on its speed, assuming dry road conditions, while the red line represents the RSI of the car on the same wet road surface. Based on RSI definition [41], the higher the RSI, the better traction is achieved between the road surface and the car. In that sense, when RSI has a low value (usually in snow-ice road classes), it is more likely that the vehicle could slide on the road with unpredictable consequences. Thus, the purpose of the proposed methodology is that the car should achieve the optimum RSI value, adjusting its speed accordingly. Therefore, the green line in Figure 6a represents the targeted RSI, which is the maximum RSI that the vehicle can achieve, given the current wet road conditions. In order to achieve that maximum RSI value, the car should reduce its speed. Accordingly, the blue line in Figure 6b depicts the current speed of the car and the red line represents the speed that the car should adapt to in order to maintain optimum RSI under current wet road conditions.

We also conducted another experiment on a hybrid driving scenario that included highway road segment and urban driving. Figure 7a presents the RSI estimation for the hybrid driving scenario. In particular, the blue line depicts the RSI under dry conditions. The red line represents the RSI under wet road conditions. The green line reflects the targeted RSI, which is the maximum RSI value that can be achieved under wet road surface status. Accordingly, Figure 7b illustrates the recommended speed that the car should adapt to, so that it will achieve the targeted RSI (Figure 7a). In Figure 7a, it is observed that RSI estimation under wet road status (red line) has much lower values than the RSI under wet road status for a pure urban driving scenario (Figure 6a). That difference can be explained because of the speed differences between the two scenarios. In fact, the hybrid scenario includes a significant portion of highway segments with an average speed as high as 73 km/h, whereas the average speed in the urban driving scenario barely reaches 30 km/h.

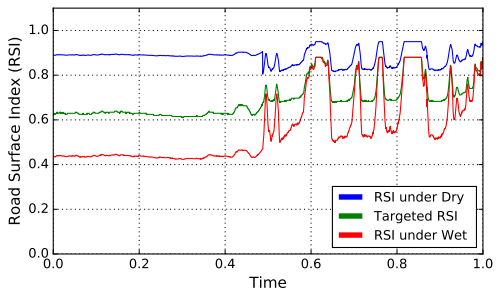


(**a**) RSI estimation and comparison for different road conditions

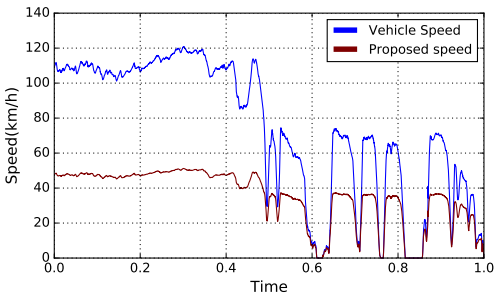


(**b**) Recommended speed range based on the proposed approach

Figure 6. Urban driving scenario.



(**a**) RSI estimation and comparison for different road conditions



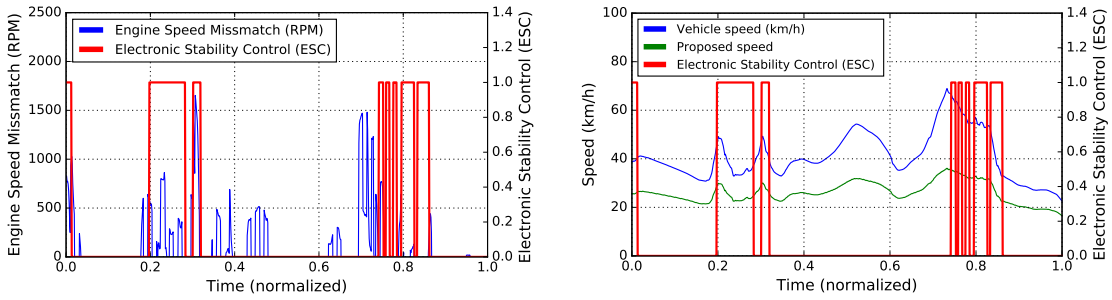
(**b**) Recommended speed range based on the proposed approach

Figure 7. Highway driving scenario.

4.3.2. Scenario 2

The second experiment was performed in order to explore the dynamic changes of the surrounding environment of a vehicle and validate the proposed speed from weather- and route-based information. To do so, we intended to push the car to its limits, so that we can achieve braking signals activation (such as ABS or ESC). However, driving in an urban area, the intended activation of ABS or ESC can be dangerous and unwanted. For that reason, Scenario 2 was performed in the test track field in the Ford Development Center in Dearborn, MI, USA, where an experienced and professional driver was responsible for all the actions. During the experiment, we replicated wet road conditions and the driver was performing maneuvers to avoid obstacles. That way, we verify how effective is the proposed speed recommendation in real conditions. It is worth mentioning that the driver intentionally drove aggressively in order to force the braking systems to intervene and the experiment was conducted intentionally on a wet road surface.

Figure 8a presents the estimated engine speed mismatch, compared to the theoretical nominal value. The calculation is based on an approximation formula that was specifically tailored for the conditions of the experiment (specific vehicle characteristics, no inclination of the ground). The red line in Figure 8a indicates Electronic Stability Control (ESC) activation, which is depicted as a binary (0-1) value (inactive-active). In Figure 8a, it is observed that, before the activation of ESC, engine mismatch had a significant sudden increase. However, there are cases where the value of mismatch was high, but ESC was not activated. This phenomenon occurred because of the approximation procedure that was adopted in order to estimate the engine speed mismatch. Figure 8b depicts the speed that the test vehicle had during the experiment (blue line). The red line reflects the activation of Electronic Stability Control (ESC) as it was described above. The green line in Figure 8b illustrates the proposed speed that the presented framework was recommending, based on weather and on-route information. The proposed framework estimates the current road class as "Wet" and evaluates the extracted PoIs in order to provide the recommended speed (green line). The presented methodology, however, is able to identify dangerous driving conditions and recommends a suitable speed range in a pro-active manner. As a result, a smart vehicle that is equipped with the presented framework could have possibly avoided the activation of the braking signals, adjusting its speed accordingly. It is worth mentioning that the proposed methodology alerts the driver with a message that indicates the recommended speed. However, it is the driver's final decision to either follow the recommendation or act independently. The proposed framework will be used as a basis for speed recommendation systems in autonomous vehicles.



(**a**) Engine speed mismatch estimation (**b**) Vehicle speed

Figure 8. Speed validation utilizing vehicular information in a test track scenario.

4.4. Evaluation of the Classification Method

In order to evaluate the proposed classification tree, we perform a comparison against three popular alternative machine learning algorithms: k-Nearest Neighbors (kNN), Logistic Regression and Support Vector Machine (SVM). Table 4 presents the four learning algorithms that were chosen based on their properties—grouping, geometrical and logical. The comparison data indicate that the

presented classification tree outperforms all the other algorithms having a slightly worse prediction score than SVM when predicting the Snow road class. On average, the classification tree achieves 10%, 17%, 13% and 28% higher accuracy when estimating the dry, ice, snow and wet road class, respectively.

Table 4. Comparison of machine learning algorithms in terms of accuracy.

	Dry	Ice	Snow	Wet
Classification Tree	90.2 %	75.3 %	82.0 %	90.2 %
kNN	79.6 %	65.7 %	69.8 %	80.0 %
Logistic Regression	76.9 %	57.9 %	56.3 %	72.4 %
SVM	83.5 %	50.0 %	82.3 %	34.6 %

5. Conclusions

In this paper, a systematic methodology for combining information provided by off- and on-vehicle car connectivity is presented. The proposed approach informs the driver about a recommended speed that the car should adapt to, in order to avoid dangerous driving. Specifically, weather- and route- based information sources are used, while the validation of the recommended speed is performed based on real-time vehicle data. The presented method estimates the road class status based on weather information and extracts Points of Interest throughout the vehicle's route. It also correlates the identified Points of Interest with the estimated road class, so that it alerts the driver with the proposed speed that the car should follow. As a future work, an extension of the current work is targeting Vehicle-to-Vehicle communication (V2V) in order to further improve the accuracy of the proposed speed.

Author Contributions: Methodology, Software, Experimental analysis and Paper Writing, I.G.; Methodology, Investigation, Supervision, Paper Writing, I.A.; Resources, Supervision, P.G.; Supervision, Project Administration D.B.

Funding: This research has been supported in part by grant NSF IIP 1361847 from the NSF I/UCRC for Embedded Systems at Southern Illinois University Carbondale (SIU). Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Pelliccione, P.; Knauss, E.; Heldal, R.; Ågren, S.M.; Mallozzi, P.; Alminger, A.; Borgentun, D. Automotive architecture framework: The experience of volvo cars. *J. Syst. Archit.* **2017**, *77*, 83–100,
2. Traub, M.; Maier, A.; Barbehön, K.L. Future automotive architecture and the impact of it trends. *IEEE Softw.* **2017**, *34*, 27–32.
3. Adiththan, A.; Ramesh, S.; Samii, S. Cloud-assisted control of ground vehicles using adaptive computation offloading techniques. In Proceedings of the 2018 Design, Automation Test in Europe Conference Exhibition (DATE), Dresden, Germany, 19–23 March 2018; pp. 589–592.
4. Chakraborty, S.; Lukasiewycz, M.; Buckl, C.; Fahmy, S.; Chang, N.; Park, S.; Kim, Y.; Leteinturier, P.; Adlkofer, H. Embedded Systems and Software Challenges in Electric Vehicles. In Proceedings of the Conference on Design, Automation and Test in Europe (DATE), Dresden, Germany, 12–16 March 2012; pp. 424–429.
5. Brannstrom, M.; Coelingh, E.; Sjoberg, J. Model-based threat assessment for avoiding arbitrary vehicle collisions. *IEEE Trans. Intell. Transp. Syst.* **2010**, *11*, 658–669.
6. Jiang, Y.; Qiu, H.; McCartney, M.; Sukhatme, G.; Gruteser, M.; Bai, F.; Grimm, D.; Govindan, R. Carloc: Precise positioning of automobiles. In Proceedings of the 13th ACM Conference on Embedded Networked Sensor Systems (SenSys '15), Seoul, Korea, 1–4 November 2015; ACM: New York, NY, USA, 2015, pp. 253–265. Available online: http://doi.acm.org/10.1145/2809695.2809725 (accessed on 22 March 2019).
7. Data is the New Oil in the Future of Automated Driving. Available online: https://newsroom.intel.com/editorials/krzanich-the-future-of-automated-driving/ (accessed on 22 March 2019).

8. Paden, B.; Čáp, M.; Yong, S.Z.; Yershov, D.; Frazzoli, E. A survey of motion planning and control techniques for self-driving urban vehicles. *IEEE Trans. Intell. Veh.* **2016**, *1*, 33–55.
9. Perala, S.S.N.; Galanis, I.; Anagnostopoulos, I. Fog Computing and Efficient Resource Management in the era of Internet-of-Video Things (IoVT). In Proceedings of the IEEE International Symposium on Circuits and Systems, Florence, Italy, 27–30 May 2018.
10. Arena, F.; Pau, G. An overview of vehicular communications. *Future Internet* **2019**, *11*, 27.
11. Abbasi, I.; Khan, A.S. A review of vehicle to vehicle communication protocols for vanets in the urban environment. *Future Internet* **2018**, *10*, 14.
12. NHTSA Calls For V2V Technology In Models Built After 2020. Available online: http://automotivedigest.com/2014/08/nhtsa-eyes-half-million-crashes-prevented-v2v-technology/ (accessed on 22 March 2019).
13. Storck, C.R.; Duarte-Figueiredo, F. A 5G V2X Ecosystem Providing Internet of Vehicles. *Sensors* **2019**, *19*, 550. doi:10.3390/s19030550.
14. Galanis, I.; Gurunathan, P.; Burkard, D.; Anagnostopoulos, I. Weather-based road condition estimation in the era of Internet-of-Vehicles (IoV). In Proceedings of the IEEE International Symposium on Circuits and Systems, Florence, Italy, 27–30 May 2018.
15. Gerla, M.; Lee, E.-K.; Pau, G.; Lee, U. Internet of vehicles: From intelligent grid to autonomous cars and vehicular clouds. In Proceedings of the 2014 IEEE World Forum on Internet of Things (WF-IoT), Seoul, Korea, 6–8 March 2014; pp. 241–246.
16. Khaleghi, B.; Khamis, A.; Karray, F.O.; Razavi, S.N. Multisensor data fusion: A review of the state-of-the-art. *Inf. Fusion* **2013**, *14*, 28–44.
17. Lu, N.; Cheng, N.; Zhang, N.; Shen, X.; Mark, J.W. Connected vehicles: Solutions and challenges. *IEEE Internet Things J.* **2014**, *1*, 289–299.
18. Faezipour, M.; Nourani, M.; Saeed, A.; Addepalli, S. Progress and challenges in intelligent vehicle area networks. *Commun. ACM* **2012**, *55*, 90–100.
19. Qu, F.; Wang, F.-Y.; Yang, L. Intelligent transportation spaces: Vehicles, traffic, communications, and beyond. *IEEE Commun. Mag.* **2010**, *48*, 136–142.
20. Galanis, I.; Olsen, D.; Anagnostopoulos, I. A multi-agent based system for run-time distributed resource management. In Proceedings of the IEEE International Symposium on Circuits and Systems, Baltimore, MD, USA, 28–31 May 2017.
21. Ozatay, E.; Onori, S.; Wollaeger, J.; Ozguner, U.; Rizzoni, G.; Filev, D.; Michelini, J.; di Cairano, S. Cloud-based velocity profile optimization for everyday driving: A dynamic-programming-based solution. *IEEE Trans. Intell. Transp. Syst.* **2014**, *15*, 2491–2505.
22. Zhang, F.; Xi, J.; Langari, R. Real-time energy management strategy based on velocity forecasts using v2v and v2i communications. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 416–430.
23. Kim, S.-W.; Liu, W.; Ang, M.H.; Frazzoli, E.; Rus, D. The impact of cooperative perception on decision making and planning of autonomous vehicles. *IEEE Intell. Transp. Syst. Mag.* **2015**, *7*, 39–50.
24. Jiang, Y.; Qiu, H.; McCartney, M.; Halfond, W.G.; Bai, F.; Grimm, D.; Govindan, A. Carlog: A platform for flexible and efficient automotive sensing. In Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, Memphis, TN, USA, 3–6 November 2014; pp. 221–235.
25. Radak, J.; Ducourthial, B.; Cherfaoui, V.; Bonnet, S. Detecting road events using distributed data fusion: Experimental evaluation for the icy roads case. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 184–194.
26. Orhan, F.; Eren, P.E. Road hazard detection and sharing with multimodal sensor analysis on smartphones. In Proceedings of the 2013 IEEE Seventh International Conference on Next Generation Mobile Apps, Services and Technologies (NGMAST), Prague, Czech Republic, 25–27 September 2013; pp. 56–61.
27. Garcia, F.; Martin, D.; de la Escalera, A.; Armingol, J.M. Sensor fusion methodology for vehicle detection. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 123–133.
28. Kumar, S.; Gollakota, S.; Katabi, D. A cloud-assisted design for autonomous driving. In Proceedings of the First Edition of the MCC Workshop On Mobile Cloud Computing, Helsinki, Finland, 17 August 2012; pp. 41–46.
29. Jain, A.; Koppula, H.S.; Raghavan, B.; Soh, S.; Saxena, A. Car that knows before you do: Anticipating maneuvers via learning temporal driving models. In Proceedings of the IEEE International Conference on Computer Vision, Santiago, Chile, 7–13 December 2015; pp. 3182–3190.

30. Juga, I.; Nurmi, P.; Hippi, M. Statistical modelling of wintertime road surface friction. *Meteorol. Appl.* **2013**, *20*, 318–329.
31. Kokogias, S.; Svensson, L.; Pereira, G.C.; Oliveira, R.; Zhang, X.; Song, X.; Mårtensson, J. Development of platform-independent system for cooperative automated driving evaluated in gcdc 2016. *IEEE Trans. Intell. Transp. Syst.* **2017**, *19*, 1277–1289.
32. Mangharam, R. The car and the cloud: Automotive architectures for 2020. *Bridg. Front. Eng. Natl. Acad. Eng.* **2012**, *42*, 25–33.
33. Khan, S.M.; Dey, K.C.; Chowdhury, M. Real-time traffic state estimation with connected vehicles. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1687–1699.
34. Chavez-Garcia, R.O.; Aycard, O. Multiple sensor fusion and classification for moving object detection and tracking. *IEEE Trans. Intell. Transp. Syst.* **2016**, *17*, 525–534.
35. Popescu, O.; Sha-Mohammad, S.; Abdel-Wahab, H.; Popescu, D.C.; El-Tawab, S. Automatic incident detection in intelligent transportation systems using aggregation of traffic parameters collected through v2i communications. *IEEE Intell. Transp. Syst. Mag.* **2017**, *9*, 64–75.
36. Silva, C.; Silva, L.; Santos, L.; Sarubbi, J.; Pitsillides, A. Broadening understanding on managing the communication infrastructure in vehicular networks: Customizing the coverage using the delta network. *Future Internet* **2019**, *11*, 1. doi:10.3390/fi11010001.
37. Du, W.; Abbas-Turki, A.; Koukam, A.; Galland, S.; Gechter, F. On the v2x speed synchronization at intersections: Rule based system for extended virtual platooning. *Procedia Comput. Sci.* **2018**, *141*, 255–262.
38. Chen, B.; Gechter, F. A cooperative control method for platoon and intelligent vehicles management. In Proceedings of the 2017 IEEE Vehicle Power and Propulsion Conference (VPPC), Belfort, France, 11–14 December 2017; pp. 1–5.
39. Y. Zhang and P. A. Ioannou. Combined variable speed limit and lane change control for highway traffic. *IEEE Trans. Intell. Transp. Syst.* **2017**, *18*, 1812–1823.
40. Zhang, Y.; Ioannou, P.A. Coordinated variable speed limit, ramp metering and lane change control of highway traffic. *IFAC-PapersOnLine* **2017**, *50*, 5307–5312.
41. Usman, T.; Fu, L.; Miranda-Moreno, L.F. Quantifying safety benefit of winter road maintenance: Accident frequency modeling. *Accid. Anal. Prev.* **2010**, *42*, 1878–1887.
42. Götzfried, F. Policies and strategies for increased safety and traffic flow on european road networks in winter. In Proceedings of the Final Conference of the COST Action, Delft, The Netherlands, 29–30 October 2008.
43. Fredriksson, L.-B. Can for critical embedded automotive networks. *IEEE Micro* **2002**, *22*, 28–35.
44. International Standard. *11898: Road Vehicles—Interchange of Digital Information—Controller Area Network (Can) for High-Speed Communicationa*; International Standards Organization: Geneva, Switzerland, 1993.
45. King, D.E. Dlib-ml: A machine learning toolkit. *J. Mach. Learn. Res.* **2009**, *10*, 1755–1758.
46. ZeroMQ. Available online: http://zguide.zeromq.org/page:all (accessed on 22 March 2019).
47. Ting, K.M. Confusion matrix. In *Encyclopedia of Machine Learning and Data Mining*; Springer: Berlin, Germany, 2017; pp. 260–260.
48. Kutila, M.; Pyykönen, P.; Kauvo, K.; Eloranta, P. In-vehicle sensor data fusion for road friction monitoring. In Proceedings of the 2011 IEEE International Conference on Intelligent Computer Communication and Processing (ICCP), Cluj-Napoca, Romania, 25–27 August 2011; pp. 349–352.