Detecting Road Events Using Distributed Data Fusion: Experimental Evaluation for the Icy Roads Case

Jovan Radak, Bertrand Ducourthial, Véronique Cherfaoui, Stéphane Bonnet

Detecting road events using distributed data fusion: experimental evaluation for the icy roads case

Jovan Radak, Bertrand Ducourthial, Véronique Cherfaoui and Stéphane Bonnet
Sorbonnes Universités, Université de Technologie de Compiègne,
CNRS, Heudiasyc, Centre de Recherche Royallieu, CS 60319, 60203 Compiègne Cedex, France
contact author: Bertrand.Ducourthial@utc.fr

***Abstract*—One of the main ideas in the area of intelligent transport systems is to use all possible information, coming from vehicles and infrastructure, in order to make the system "smarter" and avoid potentially dangerous situations – collisions, accidents, bottlenecks... However data is sometimes unreliable due to source and communication network quality, leading vehicles or even the whole system to wrong decisions. We present a generic method for detecting dangerous events on the road. To support unreliable data sources, it uses distributed data fusion. Moreover, to deal with network failures, it relies on a self-stabilizing generic distributed algorithm. Our method mixes measurements obtained from vehicle onboard sensors as well as wireless sensors placed close to the road and connected to road side units. Each vehicle computes how confident it is about a potential dangerous event using both local and remote data. To evaluate our approach, we implemented it using a specific hardware and software platform. Moreover, we instantiated a simple, yet efficient application to detect icy roads, based on temperature measurements. Thanks to both in-lab and actual on-the-road experiments, we demonstrate the possibility to deduce proper results from unreliable data and, consequently, the correctness and usefulness of our approach.**

Keywords: event detection, distributed data fusion, belief function, VANET, V2V and V2I communication, road experiment.

I. Introduction

A. Motivations

Data fusion of large amounts of data in the context of intelligent transportation systems is often a solution to the data source inaccuracy and unreliability problem. In general, data may come from different sources and is heterogeneous by nature. It is relatively easy to come up with a conclusion on how to mix two pieces of information together when their data sources are homogeneous, reliable and give similar readings. But, in order to process large amounts of heterogeneous data, some form of data fusion has to be applied. Many different data fusion techniques are proposed [1], [2], depending on the specific problem to be solved and the data sources [3]. However, in the case of conflicting or missing data sources, it may be problematic to reach a decision. The properties of the theory of belief functions make it specially suitable in the presence of imprecise, uncertain and incomplete data [4].

From another point of view, intelligent transportation systems rely on the smartness of each part of the system. Enabling vehicles and additional communication infrastructure with "smart" algorithms may help solving some of the major safety problems [5] in vehicular networks. In general, the intelligence of vehicles heavily depends on a large amount of gathered data, that is processed and then used to give drivers or vehicles useful information. Fusion of such information can be used in different applications, from various positioning problems [6] to detecting false nodes in networks [7]. It appears to be specially suitable for future smart cities equipped with a large number of sensors that may act as data sources not only for their citizens but for vehicles as well. However, it relies on unstable networks with very short useful communication times, especially between vehicles and road-side units. An approach supporting both unreliable data sources and network failures is then required.

B. Our approach

In this paper, we present a generic method to handle multiple data sources in unreliable vehicular networks, that can be used to design applications in the context of intelligent transportation systems for advance detection of dangerous events on the road. This work completes our previous paper [8]. It presents results from road experiments using sensors, road-side units (RSU) and vehicles equipped with on-board units (OBU) [9].

Our approach relies on distributed data fusion: it takes conflicting pieces of information into account and applies calculations on them until it reaches a decision. For this purpose, the theory of belief functions framework is used. The data sources are standalone sensors connected (wirelessly) to some road-side units. In-vehicle data sources, connected to on-board units, are also taken into account. OBUs and RSUs communicate when they are in range of each others. However, a moving vehicle may converge to an erroneous confidence because it considered data sent by sources which are now too far away to be pertinent. To prevent such a situation, data fusion is performed thanks to a self-stabilizing distributed algorithm, meaning it is able to converge after transient failures, including neighborhood updates [10]. The algorithm is generic and is data- representation agnostic. Hence, many applications can be derived from our method.

To the best of our knowledge, no study is similar enough to our approach to be directly compared. Hence, the bulk of our effort is in implementing the whole system, including harware and software elements, to prove the validity and practicality of our approach through real-world, on-the-road experiments. To this aim, we instantiated our method to design a temperature-sensor-based icy-road detection application. A large part of our work is thus practical and dedicated to experiments. On-the-road

experiments were carried out according to several predefined scenarios. They were supplemented by in-lab experiments, thanks to a network emulation program that allows road tests to be replayed and extended. We show that the application correctly assesses the risk of dangerous icy road conditions even in the presence of conflicting information sources and when blending stationary and moving sources. Hence, this technique surpasses a simple alert diffusion and is more robust to deficient sensors and to unstable neighborhoods.

The remainder of this article details the method and the practical development used for its validation. We start our presentation describing the related work (Section II). Next, we define the basic concepts and introduce our generic approach for detecting dangerous events. Then we explain how it can be instantiated, here as an application assessing the risk of encountering icy roads. (Section III). In Section IV we explain how this application is implemented using a detection system relying on sensors, road-side units, on-board units and dedicated software. Section V presents our experimental study. We did an extensive evaluation using both our testbed and a network emulator using real data obtained from the testbed. In Section VI, we conclude on our findings and a few directions for our future work.

II. Related work

The need for smart transportation systems, namely smart cars, has been noticed in early works [11]. The authors emphasized control of these devices as a main problem. The presence of numerous data sources are making decision problems even more complex. Different solutions were proposed to deal with large amounts of data sources in various applications [2]. The area of information fusion has been especially well studied for various applications of wireless sensor networks [12], [13]. We are proposing a different approach using one or more wireless sensors within the range of exactly one road-side unit, avoiding multihop communication between sensors and allowing only RSUs to exchange information with sensors. In this way, we are simplifying communication requirements and using wireless sensors efficiently in terms of energy.

In the last few years, we have witnessed a great effort to standardize communication protocols aimed at vehicular networks. The IEEE 802.11p standard is especially suited to applications in vehicular networks [14], in both vehicle-to-vehicle (V2V) and vehicle-to-infrastructure communication (V2I) cases. Also, the ETSI has standardized vehicular wireless communications in the network and transport layers of the communication stack. It has proposed facilities that support vehicle safety through cooperative awareness and warning messages sent asynchronously to vehicles. These facilities include two specialized types of messages: CAM – cooperative awareness message and DEMN – decentralized environmental notification message [15].

An important aspect in the functioning of intelligent transport systems is the way drivers are alerted of possibly dangerous driving conditions on the road [16] and the impact of drivers' behavior on the whole system [17]. In [18], the authors demonstrated the importance of early alerts and appropriate signaling of the possibility of dangerous events on the road to drivers. An interesting solution that puts together in-car mobile applications, hazardous events and alerts signalling to drivers is presented in [19]. The authors have developed a system that detects speed bumps and potholes based on synchronized sensor readings and extraction from a video feed taken simultaneously with these readings. Although an interesting solution, it is limited to usage of smartphones and specific capabilities of certain types of mobile phones. This study is also limited in terms of data dissemination, *i.e.* it is not clear how other users can make use of this data and if some calculations can be done cooperatively. In [20], the authors present a testbed enabling communication between vehicles and the infrastructure. This test infrastructure gives interesting results and shows the significance of the approach. However, due to the different scope of the article, the authors do not cover the possibility of equipping the infrastructure with sensors that may communicate with vehicles.

We are presenting a solution that links these two interesting points along with the problems presented in previous paragraphs. We investigate how to use multiple stationary and mobile sources of the same data type to deal with uncertainties and inaccuracies of these data sources. We show how to make use of inaccurate data sources, with the help of robust distributed data-fusion algorithms. Finally, in order to evaluate our approach, a solution that gives an assessment of the risk of finding ice on the road is proposed. We discuss results and how this approach is suitable to generate early warnings to drivers even in the presence of erroneous sensors and disturbing vehicles.

III. Distributed data fusion for dangerous event detection

Data fusion can be described as a mechanism that combines data retrieved from different sources to reduce its uncertainty and/or to generate decisions. In general, data fusion is either centralized (collecting data and applying an algorithm in some kind of central unit) or distributed (each unit is capable of applying the algorithm on data gathered locally from its neighborhood). In the remainder of this section we present the basic concepts of the theory of belief functions (Section III-A), a generic distributed data fusion algorithm based on this framework (Section III-B) and its instantiation to detect icy roads (Section III-C).

A. Theory of belief functions

The theory of belief functions, or Dempster-Shafer theory, is one of the frameworks used in data fusion [3]. This theory belongs to the probabilistic approaches for dealing with unreliable sources [21]. The Bayesian approach can be used to solve a similar set of problems. There is a lot of controversy and different arguments about which approach is the best suited [22]. We use the theory of belief functions to model uncertainties and lack of information [23]. It generalizes both the set-membership approach and the probability theory. In the Dempster-Shafer theory, the set $\Omega = \omega_1, ..., \omega_n$ of mutually exclusive propositions is called the frame of discernment. The main difference relative to the probability theory is the fact that the mass of evidence

is attributed not only to single hypotheses ω_i, but to any subset of Ω, including the empty set.

Following the general framework for the belief representation [4], we define the state of belief of the node v on the global frame of discernment Ω. A state of belief is assigned using *basic belief assignment*, most commonly represented as a *mass function*, denoted m^Ω. A mass function m^Ω is a mapping from 2^Ω (the set of subsets of Ω) to the interval $[0, 1] \in \mathbb{R}$. The sum of all masses is equal to 1. A mass value is directly proportional to confidence: $m^\Omega(A)$ increases as the node gets more confident in $A \subset \Omega$. The masses can be combined with different types of operators such as the *conjunctive operator*, which emphasizes agreement when it combines two mass functions that are reliable and independent, or the *Dempster's operator*, which ignores conflict, spreading it to the other sets. These two operators are associative and commutative. In this work, we are using the *cautious operator*, denoted $\textcircled{\wedge}$ [24]. This operator is associative, commutative and idempotent. Idempotency is an important property because it allows data coming from sources that are not independent to be combined.

In our explanations we use the term *direct confidence* to denote mass values of data obtained solely from the node itself, and the term *distributed confidence* to denote the combination of the mass values obtained from the node itself and those received from its neighbors. In this case, distributed confidence combines all data sources in the network. Moreover, we use the term *discounting* to denote the modification of a confidence so that it is less informative. In our algorithm it is used to prioritize confidences from closer nodes in comparison to confidences from nodes farther away. To calculate values that we can use in decision making, we are applying a pignistic transformation of the mass functions (given as direct and distributed confidences). This transformation computes the contribution of each subset W of Ω to a given hypothesis A knowing that W will contribute if $A \in W$. The pignistic transformation [25] of the mass function $m^\Omega(A)$ for all subsets W of Ω containing A set is defined as:

$$BetP^\Omega(A) = \sum_{W \subseteq \Omega, A \in W} \frac{1}{|W|} \frac{m^\Omega(W)}{(1 - m^\Omega(\emptyset))} \tag{1}$$

B. Distributed data fusion

We will now describe our generic approach for distributed data fusion. It relies on Algorithm 1, previously studied in [10]. This algorithm is divided into two event-driven rules.

The first rule (Lines 1–3) handles data received from communication with other nodes. It is activated each time node v receives a message from one of its neighbors and its input memory $\texttt{IN}_v$ is updated with data sent by the neighbor (Line 3). A message contains the distributed confidence computed by the sender. This is done by the second rule.

The second rule does the distributed data fusion computation and sending (Lines 4–11). It is done periodically by each node. Nodes are not synchronized and each one of them starts calculations when its timer expires. This rule assumes that each node is able to compute its direct confidence from a local measurement obtained from an external data source (such as a sensor) using the function compute_direct_confidence (Line 5). The *basic belief assignment* is done using sigmoid-like functions that map the measurement to a mass function, spreading confidence on subsets of the frame of discernment Ω. This function depends on the targeted application; we give an example in the next subsection.

Algorithm 1: **Distributed confidence calculation for the node** v

```
1   Upon the arrival of a new message:
2       receive( dist_conf ) from node u
3       IN_v[u] ← dist_conf
4   Upon the expiration of the timer of the node v:
5       OUT_v ← compute_direct_confidence()
6       for each u in IN_v do
7           OUT_v ← OUT_v ⩓ r(IN_v[u])
8       end for
9       send( OUT_v ) in the neighborhood
10      Remove old messages in IN_v
11      Restart the timer
```

The distributed confidence is computed iteratively combining the direct confidence of the node (stored in $\texttt{OUT}_v$) with all data received from neighbors (Lines 5–8). It relies on the cautious operator $\textcircled{\wedge}$ and the discounting $\mathbf{r}$ (Line 7). Old data is removed from the input memory $\texttt{IN}_v$ so that if a node leaves the neighborhood, its data will not be taken into account anymore after a fixed number of timers (considering several timers is useful to prevent message loss). Finally, the result – the distributed confidence of the node stored in $\texttt{OUT}_v$ – is sent to the neighborhood (Line 9) and the timer is restarted to program the next computation (Line 11).

Each node computes its distributed confidences using those received from its neighbors. Thus, the direct confidence of a node will be taken into account by its neighbors, then by the neighbors of its neighbors and so on. In this way, each node is contributing to the computation of what we call the distributed confidence.

Discounting is applied to each distributed confidence received to give priority to closer sources over farther ones. Thus, a distant information source will still be taken into account in the computation of the distributed confidence of a given node but with decreasing importance as its output will be discounted at each hop (given as the $\mathbf{r}$ function at Line 7). As a consequence, the distributed confidence computed by each node may be different, as even though the same information sources (the direct confidences of each node of the network) are taken into account by each node, they are discounted differently depending on the position of the node in the network. The result reflects the local situation in the vicinity of the node without ignoring information from other, more distant nodes.

Data incest appears when a single piece of information, coming from a single source, is used in the fusion process more than once. Such a situation can appear with our distributed data fusion algorithm; however it is solved thanks to the idempotency of the cautious operator $\textcircled{\wedge}$. The properties of the cautious operator $\textcircled{\wedge}$ and the discounting $\mathbf{r}$ both ensure convergence of the algorithm even in the presence of transient failure [10].

This approach accepts any kind of frame of discernment and can be applied to a large set of applications in the

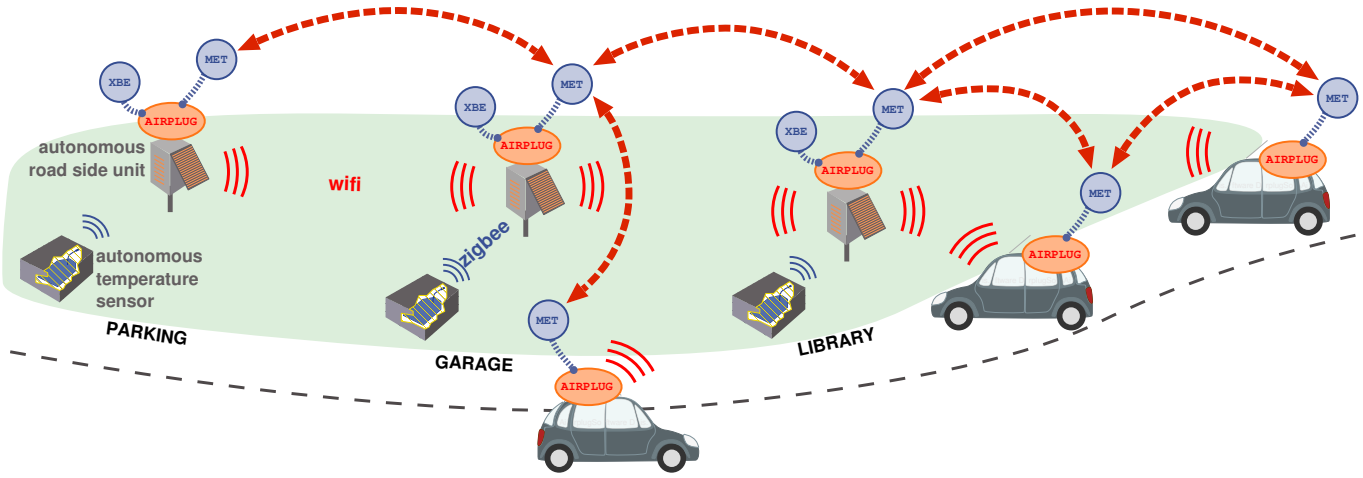


Fig. 1: Testbed with Xbee sensors, road-side units and software support from Airplug platform

Intelligent Transportation System field. To evaluate its interest and robustness, we instantiate it through a simple yet efficient application for icy road detection.

C. Application to the robust detection of icy roads

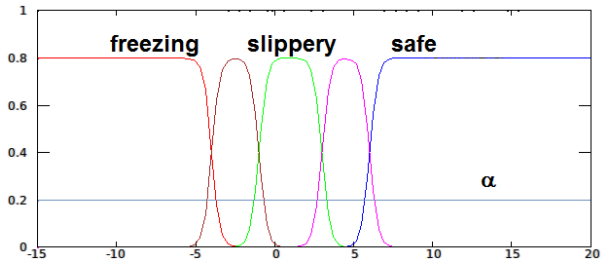
In this section, we explain how to instantiate our generic method to detect a given road hazard: icy roads. This consists in designing a model adapted to this problem and assigning direct confidences from measurements (function compute_direct_confidence in Algorithm 1).

While it is clear that the road is dangerous for some temperatures and not for others, there exists a range of temperatures where there is a doubt. We define three road states, corresponding roughly to three temperature ranges. These states lead to the definition of the following frame of discernment: $\Omega = \{freezing, slippery, safe\}$.

We assign basic belief assignments from temperature sensor measurements using sigmoid-like functions defined for the subsets of the Ω set (Fig. 2). The idea is to give a large mass to the state corresponding to the measured temperature and smaller ones to the other subsets of Ω. As the measurement gets closer to a threshold between two states, the mass gets more and more spread over several subsets of Ω. Moreover, we do not completely trust sensors and a mass is always given to the subset Ω itself, representing their unreliability (given as α, $0 < \alpha < 1$). We chose not to consider the subset $\{freezing, safe\}$ corresponding to disjoint events. When assigning masses for the direct confidence, there is no value for the subset $\emptyset$ because it represents the "conflict" proposition (some values may appear for $\emptyset$ after combination of several sources in the distributed confidence).

Thresholds and reference values that define road states are chosen in such a way that they correspond to real values. T_{cur} is the current temperature – value read from the sensor, T_{ref}, T_{thr1}, T_{thr2} and λ are the reference values chosen in such a way that sigmoids have appropriate shapes (Fig. 2) as well as to fit the values needed to describe icy roads. These parameters can be changed easily and we can create a model that can be tested with higher temperatures than those that produce an icy road (which is useful to perform tests during any season).

For our purposes, the road is *safe* for temperatures above $6°C$, between *safe* and *slippery* in the interval $\{3°C, 6°C\}$, most probably *slippery* in the temperature interval $\{-1°C, 3°C\}$, between *freezing* and *slippery* in the interval $\{-1°C, -4°C\}$, and most probably *freezing* for temperatures below $-4°C$. α is fixed to 0.2; it corresponds to the belief of an incorrect temperature (unreliable sensor).



$$
\begin{aligned}
m(\{freezing\}) &= (1-\alpha) - \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}+T_{\text{thr2}})}} \\
m(\{freezing, slippery\}) &= \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}+T_{\text{thr2}})}} - \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}+T_{\text{thr1}})}} \\
m(\{slippery\}) &= \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}+T_{\text{thr1}})}} - \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}-T_{\text{thr1}})}} \\
m(\{slippery, safe\}) &= \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}-T_{\text{thr1}})}} - \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}-T_{\text{thr2}})}} \\
m(\{safe\}) &= \frac{1-\alpha}{1+e^{-\lambda(T_{\text{cur}}-T_{\text{ref}}-T_{\text{thr2}})}} \\
m(\{\Omega\}) &= \alpha
\end{aligned}
$$

Fig. 2: Sigmoid-like functions definitions for subsets of Ω.

Note that instantiating our method on a given problem (here icy road detection) requires expertise in the related field to define the frame of discernment, the sigmoids-like functions and the thresholds.

IV. Implementation of the detection system

In this section, we present the hardware and software implementation of our road event detection system. We present the system architecture, its hardware components, the framework used to implement the applications and the software components of the system. Its instantiation for icy road detection only impacts the type of sensors used and the software parameters.

A. System architecture

Our detection system (Fig. 1) relies on stationary and mobile measurements combined by using our distributed data fusion application, described in the previous section. Stationary measurements are performed by wireless sensors located close to the road, sending their measurements to solar-powered road-side units (Fig. 3). Road-side units communicate between themselves and with on-board units in their transmission range. Vehicles in the testbed are fitted with on-board units allowing them to communicate with road-side units and other vehicles.

RSUs and OBUs use WiFi to communicate between themselves. Messages that are exchanged in this communication are those of the distributed data fusion application, namely the distributed confidence that is computed periodically by each node. Communication with sensors relies on dedicated packets exchanged through Zigbee modules – that both wireless sensors and RSUs have. OBUs are not fitted with Zigbee modules and receive measurements directly from vehicle sensors.

B. Hardware components

The hardware part of the testbed consists of cheap off-the-shelf wireless sensors and a dedicated embedded hardware platform, called the Airbox and developed in our laboratory, that is used in both RSUs and OBUs. The testbed, as deployed, features 3 RSUs and 3 Xbee sensors communicating wirelessly with the RSUs via the 802.15.4 protocol. Data is gathered using the serial ports of the RSUs. We have developed a support program that is used to connect computing units (RSUs or regular PCs) with wireless sensors and gather data through the Xbee modem connected to the serial port of the RSU [9].

The wireless sensors[1] are equipped with a temperature sensor. We have deployed a simple topology – using exactly one sensor for each RSU. The support program allows more complicated scenarios with multiple sensors per RSU but this strictly depends on the application for which this testbed is used.

The Airbox units are based on the IGEP platform[2] – a TI OMAP controller with 512 MB of RAM. They are running the Linux operating system and a specialized software suite, called Airplug and described in the next section. The driver part of the support program is responsible for establishing a connection between the Airbox serial port, the Xbee module and the Xbee sensor. The application exchanges binary packets between the RSU and the wireless sensors and decodes them upon reception. Decoded data is further transferred to other applications that may use them.

[1] http://www.digi.com/products/wireless-modems-peripherals/wireless-range-extenders-peripherals/xbee-sensors

[2] https://www.isee.biz/products/igep-processor-boards

C. Airplug framework

The Airplug software distribution is a program suite aiming to fully support development, testing and deployment of dynamic networks. It relies on a simple framework based on a few development rules focusing on the implementation of portable software for highly dynamic networks. These rules are: *(i)* usage of the standard input/output system to ensure independence from the programming language used for implementation, *(ii)* usage of standard ASCII text messages to ensure portability (support of different operating systems), *(iii)* a simple message addressing scheme which includes addressing of applications with the value pair (app_name, area), where *area* is a keyword determining the range of broadcast (LCH to broadcast to the localhost only, AIR to broadcast to the neighborhood) and *(iv)* reliance on broadcast and on managing message visibility with subscriptions to certain applications.

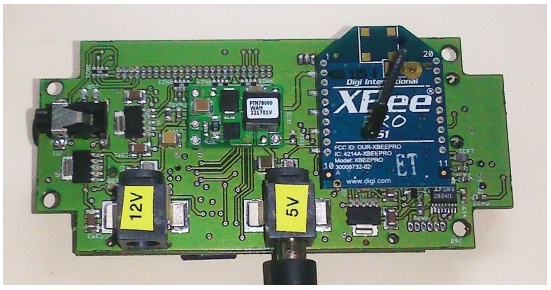
The Airplug software distribution supports several *modes* – independent implementations that complement each other. These modes include: the **terminal mode** – a standard UNIX-compatible command line implementation; the **emulation mode** – a network emulator that reproduces wireless communications using inter-process communication (IPC) while keeping upper layers identical to those used in experiments [26] and the **live mode**, an efficient implementation suitable for execution on constrained embedded systems used during real experiments. Currently, the Airplug software distribution is mainly written in C and Tcl/Tk; it can easily be extended by writing applications (in any programming language) that follow those guidelines. We used the terminal mode to develop and test the applications, the emulation mode to test our algorithms in a realistic environment and finally the live mode to perform experiments on the real testbed with vehicles.

D. Software components

The application that makes the connection between wireless sensors and RSUs is called XBE (Fig. 1). At the lower level, the XBE application enables communication and packet exchange using Digi International's proprietary format through a serial port and an Xbee module. At the higher level, the XBE application sends temperature data gathered from an Xbee sensor. The XBE application can be used in different scenarios. While its basic feature is to gather data from wireless Xbee sensors and transfer them to other applications, it can also be used to gather data and store it in a predefined log file, to read data from log files that were created in previous experiments and to read user-generated data files. Basic manipulation of gathered data is possible. For example, we can add an offset to data to simulate different environmental conditions (useful for recreating a specific type of environment – *e.g.* winter temperatures during summer). Log files can be read at the rate that was used during recording or at a higher rate as specified by the parameters of the application. All these parameters can be used both in emulation and live modes [9].

We have developed the MET application to test and develop distributed data fusion algorithms. This application is a generic implementation of the belief function framework explained in Section III. MET is able to generate values needed to study

(a) Xbee sensor (with xbee modem)



(b) Airbox – IGEP based development boards, used as RSU

(c) RSU enclosed with energy harvesting unit and Xbee sensor

Fig. 3: Hardware elements of the detection system

the robustness properties of the algorithm and to generate test measurements according to a given function and periodicity. This last feature has been used on the Airbox units installed in vehicles lacking temperature sensors and for emulation purposes. Another feature of this application is to accept measurements coming from other applications. This functionality is used during experiments: XBE sends the gathered data and MET applies the data fusion algorithm to the received data (Fig. 1).

MET can apply any user-defined frame of discernment (Ω set) and the sigmoid functions defined for it. Independently of the data source, MET uses the sigmoid functions defined by user-supplied equations (Fig. 2), and calculates the direct confidences according to the received values and the defined frame of discernment. To ease the cautious computation, MET is applying calculations on the weights instead of mass functions, which are just a different representation of the basic belief assignment obtained through the commonalities [10]. Data sets on which MET works are then vectors of discretized weights with one component for each subset of Ω, not including Ω. Each vector component has a value in $(0, 1]$.

MET receives a measurement from XBE and then computes the local direct confidence as a mass function using the frame of discernment and the sigmoids (Fig. 2). It then converts these values to weight vectors and computes the distributed confidence merging weight vectors with those received from neighbors. It then proceeds with the calculation of the distributed confidence as explained in Section III and sends it to the neighbors.

V. Experiments and results

This section presents the experimental study of the icy road detection application designed to evaluate our method. We give a detailed explanation of both the methodology and experimental setup before presenting and commenting our results.

A. Methodology and setup

Methodology. Our aim is to show the practical interest of our method dedicated to the detection of dangerous events on the road, as well as its robustness against unreliable or misplaced sources of data. For this purpose, we have deployed our testbed (described in Section IV) to test the icy road detection application on a realistic situation. We focus on a limited number of vehicles and RSUs (as depicted in Figure 1) primarily to show how a single piece of information influences data fusion calculations. This way, we can observe how alert generation improves with new data sources. Moreover we compare the results between a *regular* and a *misplaced sensor* scenario, the latter being a variant with a temperature sensor placed indoor, in the garage of our laboratory, thus giving erroneous measurements of the road temperature.

Our experimental study relies on real-life experiments using both the hardware and the software platform described in Section IV. Additionally, these experiments have been replayed and supplemented using the Airplug network emulator. This tool replaces inter-vehicle communications by inter-process communications, keeping the programs as deployed on the Airboxes during road tests [26]. It is convenient to replay and extend real experiments while still offering accurate results. For some scenarios, we give the results both from on-the-road and in-lab emulation experiments for comparison purposes.

When using the Airplug network emulator, the GPS positions of the RSUs and the vehicles are the same as in the real experiment; they have been recorded during real tests and are replayed during emulation. There is no real sensor in the emulation; the XBE applications are then used in emulation mode: they output data coming from a log file recorded during the real tests.

Not every result can be shown here. We focus on the case of the first vehicle encountering an icy road. We show how our algorithm updates the distributed confidence about the icy road event while the vehicle is approaching. Results are displayed using the pignistic probabilities calculated using Equation 1. These probabilities can directly be used to warn the driver.

Scenarios. We divided our study into three groups of scenarios. The first group, called the *single-vehicle scenarios*, considers the three RSUs with their sensors and a single vehicle that is driving by the RSUs. The second group, called the *two-vehicle scenarios*, completes the first one with a second vehicle driving behind the first one the same lane. Though this vehicle does not have more information about the icy road area, it may influence the result of the first one. Finally, the third group, called the *three-vehicle scenarios* completes the second one with a third vehicle moving in the opposite direction (Fig. 1). This vehicle has already encountered the icy road area and this may affect the result of the first vehicle.

Figure 4 is based on a screenshot of the Airplug emulator

using OpenStreetMap tiles. It depicts the three-vehicle scenario and the testbed as deployed in the vicinity of the laboratory. The blue arrow shows the direction of the vehicles on the road, which are represented by blue circles. The RSUs are represented by black squares. The vehicles encounter RSU-L (for Library) first, then RSU-G (for Garage) and finally RSU-P (for car Park). Each element in the emulation features a small histogram showing the pignistic probabilities for the direct and the distributed confidences (see Equation 1). In our experiments, the dangerous icy road spot is located near RSU-P and shown as a red hexagon on Fig. 4.

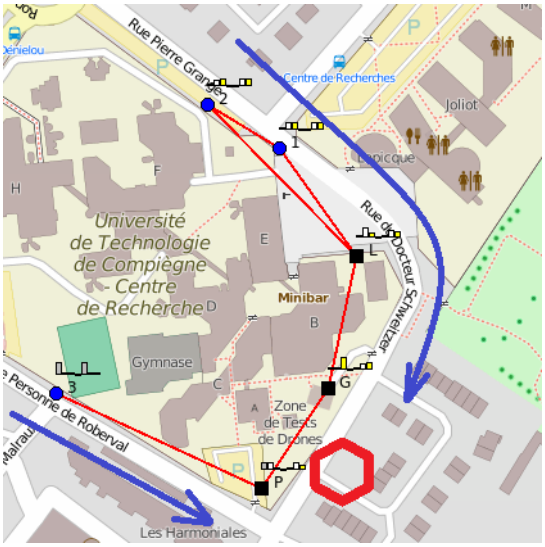


Fig. 4: Map showing the positions of the road-side units (black squares) and of three vehicles (blue circles); their direction of movement is given by the blue arrows and the icy part of the road is marked with a red hexagon near RSU-P.

In each group of scenarios (featuring one, two and three vehicles), we considered two sub-cases differing in the placement of the wireless sensors. The first one is the *regular* case in which all sensors are placed close to the road and output correct temperature measurements. In the second case, one sensor, connected to RSU-G, is placed indoor in the garage and measures higher temperatures that do not reflect the road temperature. This is an example of a misplaced sensor, thus we refer to this sub-case as *misplaced*. Our aim is to study the behavior of our system in such a situation and to observe how it adapts to these kind of extreme cases. In total, we have six scenarios, namely 1R and 1M for the single-vehicle regular/misplaced ones, 2R and 2M for the two-vehicle regular/misplaced ones, and 3R and 3M for three-vehicle regular/misplaced ones. Additionally, for clarity we add the suffix -road (as in 1R-road) when the scenario has been run on the road and -emu (as in 3M-emu) when it has been run in the network emulator.

Parameters. Our main aim in these scenarios is to demonstrate the possibility of detecting an icy road segment (the dangerous spot) before the vehicle actually reaches it. We want to show how data fusion is handling multiple sources of data, both stationary and mobile, and how, using this data, it properly assesses the state of the road. We also want to show that the alert linked to the possibility of icy roads is appropriately propagated to the vehicles even when a sensor outputs erroneous measurements. Thus, for the regular scenarios we have chosen road temperatures of $3°C$, $-1°C$ and $-3°C$ for RSUs L, G and P respectively. For the misplaced scenario, the temperature measured by the sensor of RSU-G is $21°C$ – typical indoor temperature. The other sensors outputs are identical to the regular scenario.

In order to have realistic scenarios for both emulation and real experiments, vehicles are generating decreasing temperatures. The function for the temperature that the vehicle *"measures"* is constructed in such a way that the vehicle reports temperatures similar to those of the RSUs when it is passing near them. In the one-vehicle scenarios, the vehicle starts with a temperature of $7°C$ that decreases linearly each second by $0.133°C$. For the two-vehicle scenarios, the leading vehicle starts with a temperature of $5.5°C$ and the second vehicle with a temperature of $7°C$. Both measured temperatures are linearly decreasing each second by $0.133°C$. For the three-vehicle scenarios, the first two vehicles have the same measured temperature as in the two-vehicle scenarios while the third vehicle, driving in the opposite direction, starts with a temperature of $-6.67°C$ and linearly increases its temperature by $0.111°C$.

Timing. In the beginning of both one-vehicle scenarios, the vehicle is out of range of any RSU. The periods of communication between the vehicle and the different RSUs are: *(i)* the period during which the vehicle only communicates with RSU-L from $t = 12\,\text{s}$ to $t = 28\,\text{s}$; *(ii)* the period during which the vehicle communicates with RSUs L and G from $t = 28\,\text{s}$ to $t = 41\,\text{s}$; *(iii)* the period during which the vehicle communicates with all three RSUs from $t = 41\,\text{s}$ to $t = 51\,\text{s}$; *(iv)* communication with RSUs G and P from $t = 51\,\text{s}$ to $t = 70\,\text{s}$ and; *(v)* communication with RSU P only from $t = 70\,\text{s}$ to $t = 73\,\text{s}$. We consider that the vehicle enters the dangerous area (icy road) at $t = 55\,\text{s}$.

Both two-vehicle scenarios start with vehicles communicating with each other, but both of them are out of range of any RSU. Vehicle 1 is starting closer to the RSUs, while Vehicle 2 is further away at a location similar to the starting point of the vehicle in the one-vehicle scenarios. Both vehicles are driving around the RSUs in the same direction and on the same paths as the vehicle in the one-vehicle scenarios, Vehicle 1 driving 10 seconds ahead of Vehicle 2. This means that timings given in the previous paragraph apply to Vehicle 2 in this scenario, while Vehicle 1 has exactly the same timings shifted by 10 seconds. For example, comparing this observation with the previous scenario, we can see that the leading vehicle (Vehicle 1) starts communicating with RSU-L at $t = 12 - 10 = 2\,\text{s}$, then starts communicating with RSU-G at $t = 28 - 10 = 18\,\text{s}$ and with RSU-P at $t = 41 - 10 = 31\,\text{s}$.

In the three-vehicle scenarios the first two vehicles (with the numbers 1 and 2 on the Fig 4) have the same timings as in the two-vehicle scenarios. The third vehicle is moving in the opposite direction, it first starts to communicate with RSU-P at $t = 12\,\text{s}$, then with RSU-G at $t = 15\,\text{s}$ and RSU-P $t = 28\,\text{s}$, with all three RSUs and in the last two steps with

RSU-G and RSU-L and finally only with RSU-L, before it loses connection at $t = 76$ s with any of them. Vehicle 3 is connected with Vehicle 1 from $t = 17$ s to $t = 32$ s and with Vehicle 2 from $t = 22$ s to $t = 37$ s. The timings given in the previous paragraphs are calculated from the emulation scenarios. In the real experiments these values were varying depending on the traffic in the part of the city where the experiments were conducted and the communication range between the vehicles.

B. Road experiments

We analyze here the on-the-road results obtained using either a single or two vehicles. This is a proof of concept for our method. Comparison with emulation results is discussed in the next section.

1R-road: single-vehicle on-the-road regular scenario. The vehicle-computed pignistic probability is drawn in Figure 5. We can see that the vehicle is getting an alert at $t = 15$ s, when it starts communicating with RSU-L and the value for the {*slippery*} state reaches the highest value of the three. We can observe that the value for the {*slippery*} state drops a few times. This is due to message losses during communication with the RSUs. By changing the timers values in the MET application, we may avoid such phenomena. However, the application may be less reactive. Hence we chose to keep this setting because it does not disturb the alert generation.

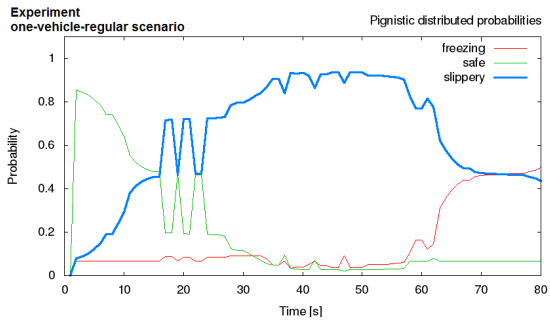


Fig. 5: [1R-road] Pignistic probabilities for the distributed confidence computed in the vehicle.

1M-road: single-vehicle on-the-road misplaced scenario. In this case, we show the influence of erroneous readings from one sensor and their impact on the generation of the icy road alert. We can see that the vehicle is alerted at $t = 25$ s, 10 seconds later than in the 1R-road case (Fig. 6). At this point, the vehicle is still only in the range of the first RSU. Hence, even if a sensor is erroneous, the vehicle still has a large-enough time to react (about 30 s before reaching the dangerous area).

2R-road: two-vehicle on-the-road regular scenario. Results are given in Figure 7. We observe that there are some fluctuations until $t = 6$ s, after that the vehicle is continuously alerted.

In this two-vehicle scenario, the first vehicle gets connected to RSU-L at $t = 2$ s. This graph should then be compared with Figure 5 at $t = 12$ s (see the timing description in Section V-A). Taking into account this 10 s shift, the first vehicle detects the danger approximately at the same date and the same position as

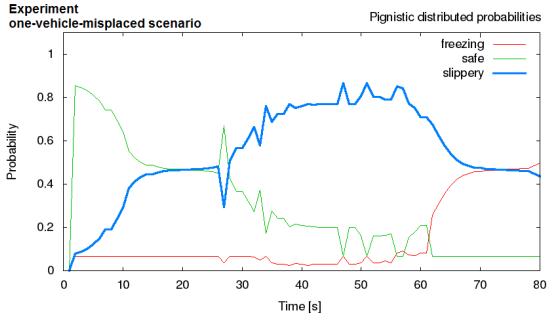


Fig. 6: [1M-road] Pignistic probabilities in the vehicle.

in the 1R-road case[3]. This means that the second vehicle does not disturb the first vehicle while the information it announces may contradict the RSU (being located behind the first vehicle, and thus farther away from the danger area).

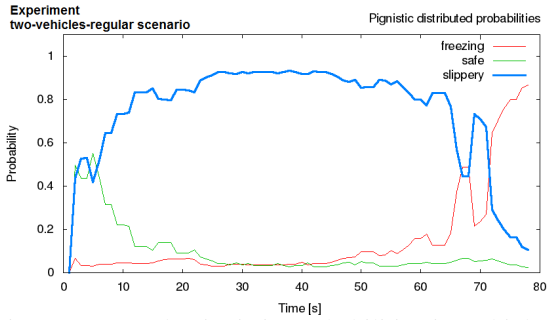


Fig. 7: [2R-road] Pignistic probabilities in Vehicle 1.

2M-road: two-vehicle on-the-road misplaced scenario. As in the 1M-road case, one of the sensors (connected to RSU-G) is providing false readings, interfering with icy road detection. Results are presented in Figure 8. The first vehicle is alerted at approximately $t = 20$ s.

Compared to the 2R-road case, the alert is generated approximately 14 seconds later (to be compared with the 10 s difference between the 1R-road and 1M-road cases). For comparison purposes with the 1M-road case, the timing of Vehicle 1 has to be shifted by 10 s (see the timing description in Section V-A), leading to an alert detection at $t = 30$ s, to be compared with the 25 s in the 1M-road case. Hence, the presence of the second vehicle adds an approximately 5 s delay in the case of the misplaced-sensor scenario.

This can be explained by the fact that the second vehicle stays under the influence of the RSU-G and its erroneous information for a long time. The first vehicle will then be influenced by the false measurement during a longer time, either directly by RSU-G or indirectly by the second vehicle, also under the influence of RSU-G.

It is worth noting that, even in this unfavorable case, our system still generates an alert in Vehicle 1 with a comfortable delay (about 25 s before reaching the danger area).

[3]The fluctuations before $t = 6$ s seem to correspond to the stabilization of the algorithm at the start of the scenario.

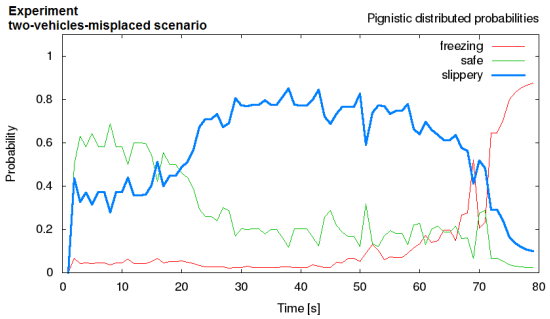


Fig. 8: [2M-road] Pignistic probabilities in Vehicle 1.

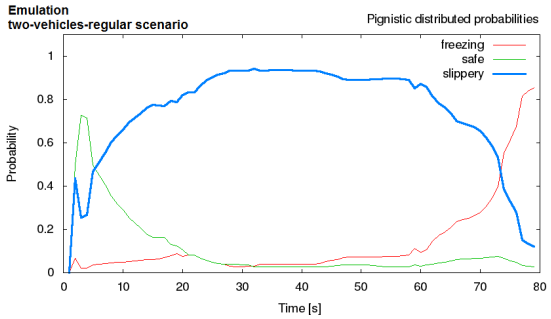


Fig. 9: [2R-emu] Pignistic probabilities in Vehicle 1

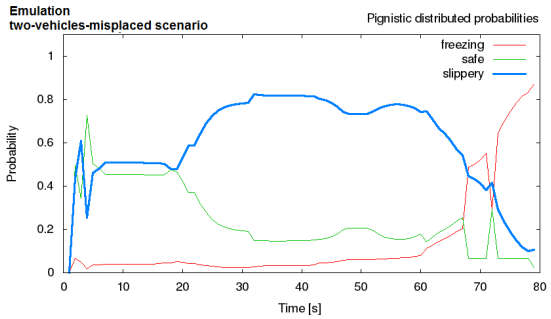


Fig. 10: [2M-emu] Pignistic probabilities in Vehicle 1.

C. In-lab experiment using network emulation

For this part of our study, we rely on the Airplug network emulator which yields results that are very close to those obtained from on-the-road experiments. We begin by presenting the previously-studied two-vehicle scenarios for comparison purposes between on-the-road and in-lab experiments. We then extend our study with the three vehicles scenarios.

Validating the emulation using 2R-emu and 2M-emu. Figures 9 and 10 display the results obtained using the Airplug network emulator while replaying the regular and misplaced scenarios with two vehicles. These figures can be compared to Figures 7 and 8 respectively, obtained with on-the-road experiments.

In contrast to results obtained during the on-the-road experiments, where the value for the {*slippery*} state dropped a few times, the emulation gives smooth curves. This is due to real communication conditions during the on-the-road tests. Nevertheless, the pignistic probabilities computed during on-the-road experiments show the same tendency as that observed in emulation. The icy part of the road is detected at approximately the same time as it is in the emulations. The differences are due to the events that occur in the real environment: vehicle speed variations, communication channel limitations (message losses and variation of transmission range), etc.

While some packet losses may generate small variations, the perturbation is generally short (a single timer). An alert generation mechanism can easily support them by adding a minimal delay after which the alert is effectively confirmed. Moreover, it is possible to avert packet losses from a neighbor that has already sent its data by reusing the previous one. But this would give a less reactive system, especially when taking the departure of a neighbor into account. That is why we chose to keep this setting.

This similarity between the on-the-road and the in-lab experiments confirm our previous results with one vehicle in [8]. In the following, we use network emulation to study a three-vehicle scenario.

3R-emu and 3M-emu: three vehicles by emulation. In this scenario, a third vehicle that already learned about the danger is added, driving in the opposite direction lane. Results are presented in Figures 11 and 12.

We can see that, in the regular scenario, the alert is generated as soon as Vehicle 1 communicates with the first RSU (RSU-L). This can be explained by the fact that, at this time, Vehicle 3 is in the icy road area and is already communicating with the RSUs, enforcing the confidence in the danger. Hence, Vehicle 1 receives a confidence in the danger that is larger than in the two-vehicle scenario (Fig. 9).

At the time when Vehicle 1 starts to communicate with Vehicle 3, we can see in Figure 11 a small drop of the value for the {*slippery*} state and an increase of the value for the {*freezing*} state. This is due to measurements that are coming from Vehicle 3 as it is driving on parts of the road that are colder than $-3°C$. However, this change does not negatively impact a possible alert generation and we can actually conclude that it helps because of the additional decrease of the value for the {*safe*} state.

Regarding the misplaced scenario (Fig. 12), we can see that the freezing state (red curve) is detected later than in the two vehicles scenario (2M-emu case in Fig. 10). Indeed at this time (around $t = 70\,\mathrm{s}$) the third vehicle increases the confidence into slippery road. Compared to the regular scenario, we can see the influence of the misplaced sensor in the values of the probabilities for the slippery state (higher in Fig. 11 than in Fig. 12). However, the trend is globally the same, thus ensuring a good decision: the alert is generated in time.

VI. Conclusion

In this paper, we have presented a generic method for detecting events on the road. It relies on a robust distributed data fusion algorithm that can be instantiated with any frame of discernment. The architecture mixes stationary and mobile sources of data. The algorithm combines several sensor measurements and propagates a mass vector with confidences on every subset of the frame of discernment. The main advantage

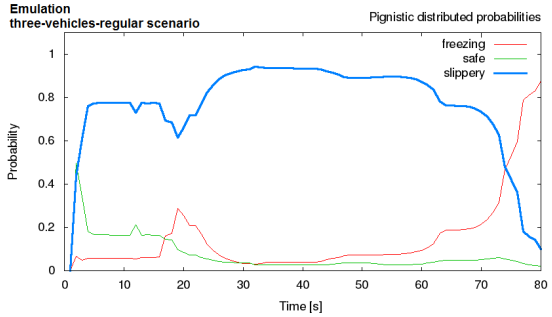


Fig. 11: [3R-emu] Pignistic probabilities in Vehicle 1.

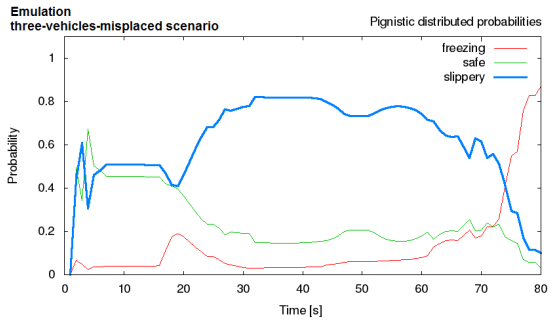


Fig. 12: [3M-emu] Pignistic probabilities in Vehicle 1.

of this technique is its robustness to erroneous measurements and to the dynamics of the vehicular network.

This generic method can be instantiated for a large set of applications, by choosing adequate sensors, determining the appropriate frame of discernment and the correct mapping to build the direct confidence from the local measurements (using sigmoid-like functions). To show its usefulness in real applications and to evaluate its robustness, we instantiated an icy road detection application based on three states: freezing, slippery, safe.

We have designed a complete proof-of-concept featuring wireless sensors and solar-powered RSUs using WiFi to communicate between themselves and with vehicles. Extensive experiments have been done in various scenarios featuring a a varying number of vehicles, with and without erroneous measurements. Results show that the distributed data fusion application enables approaching vehicle to be advised earlier than with a simple alert broadcast generated when an average temperature falls under a threshold. Moreover, data fusion can generate an alert for approaching vehicles even when one of the sensors gives completely different measurements. We have to specially emphasize the good behavior of this application when it was used in scenarios with two and three vehicles and the way it adapted to inaccurate sensor readings.

Future work will include adaptation of our application to the detection of other hazards. Specifically, we are working on traffic bottleneck and heavy rain alerts as part of our involvement in the CoMoSeF project, as well as on compliance to standardized DENM messages.

VII. Acknowledgments

This work was carried out and funded within the framework of the Labex MS2T. It was supported by the French Government, through the program "Investments for the future" managed by the National Agency for Research (Reference ANR-11-IDEX-0004-02). It has been partially supported by the Celtic Plus project CoMoSeF (Cooperative Mobility Services of the Future).

References

[1] R. Luo and M. Kay, "Multisensor integration and fusion in intelligent systems," *IEEE Trans. Syst., Man and Cybern., Syst.* , vol. 19, no. 5, pp. 901–931, Sep 1989.

[2] R. Luo, C.-C. Yih, and K.-L. Su, "Multisensor fusion and integration: approaches, applications, and future research directions," *IEEE Sensors J.*, vol. 2, no. 2, pp. 107–119, Apr 2002.

[3] B. Khaleghi, A. M. Khamis, F. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013.

[4] D. Dubois and H. Prade, "Evidence, knowledge, and belief functions," *Int. J. of Approx. Reason.*, vol. 6, no. 3, pp. 295 – 319, 1992.

[5] N.-E. El Faouzi, H. Leung, and A. P. Kurian, "Data fusion in intelligent transportation systems: Progress and challenges - a survey," *Information Fusion*, vol. 12, no. 1, pp. 4–10, 2011.

[6] M. Fogue, F. J. Martinez, P. Garrido, M. Fiore, C. F. Chiasserini, C. Casetti, J. C. Cano, C. M. T. Calafate, and P. Manzoni, "On the use of a cooperative neighbor position verification scheme to secure warning message dissemination in VANETs," in *Proc. of IEEE LCN 2013*, Sydney, Australia, Oct. 2013, pp. 276–279.

[7] N. El Zoghby, V. Cherfaoui, B. Ducourthial, and T. Denoeux, "Distributed data fusion for detecting sybil attacks in VANETs," in *Proc. of Belief Functions*, Compiègne, France, 2012, pp. 351–358.

[8] J. Radak, B. D. Ducourthial, and V. Cherfaoui, "Early detection of dangerous events on the road," in *Proc. of VNC 2014*, Paderborn, Germany, Dec. 2014.

[9] J. Radak, B. D. Ducourthial, V. Cherfaoui, and S. Bonnet, "Design and implementation of the vehicular network testbed using wireless sensors," in *Proc. of WiSARN 2014*, Benidorm, Spain, May 2014.

[10] B. Ducourthial, V. Cherfaoui, and T. Denoeux, "Self-stabilizing distributed data fusion," in *Stabilization, Safety, and Security of Distributed Systems*, LNCS vol. 7596, Toronto, Canada, 2012, pp. 148–162.

[11] P. Varaiya, "Smart cars on smart roads: problems of control," *IEEE Trans. on Automatic Control*, vol. 38, no. 2, pp. 195–207, Feb 1993.

[12] D. Tapia, F. de la Prieta, J. Bajo, J. Corchado, and O. Garcia, "Wireless sensor networks for data acquisition and information fusion: A case study," in *Proc. of Information Fusion (FUSION)*, Edinburgh, Scotland, July 2010, pp. 1–8.

[13] A. Daz-Ramrez, L. A. Tafoya, J. A. Atempa, and P. Meja-Alvarez, "Wireless sensor networks and fusion information methods for forest fire detection," *Procedia Technology*, vol. 3, pp. 69 – 79, 2012.

[14] F. A. Teixeira, V. F. e Silva, J. L. Leoni, D. F. Macedo, and J. M. Nogueira, "Vehicular networks using the {IEEE} 802.11p standard: An experimental analysis," *Vehicular Communications*, vol. 1, no. 2, pp. 91 – 96, 2014.

[15] J. Santa, F. Pereniguez, A. Moragon, and A. Skarmeta, "Vehicle-to-infrastructure messaging proposal based on CAM/DENM specifications," in *Proc. of IFIP Wireless Days*, Valencia, Spain, Nov 2013, pp. 1–7.

[16] C. Fang, C. Chiou, C. Chen, and S. Chen, "Dangerous driving condition analysis in driver assistance systems," in *Proc. of IEEE ITSC*, St. Louis, USA, Oct 2009, pp. 1–6.

[17] F. Dressler and C. Sommer, "On the impact of human driver behavior on intelligent transportation systems," in *Proc. of IEEE VTC 2010-Spring*, Taipei, Taiwan, May 2010, pp. 1–5.

[18] L. Malta, C. Miyajima, and K. Takeda, "A study of driver behavior under potential threats in vehicle traffic," *IEEE Trans. Intell. Transp. Syst.*, vol. 10, no. 2, pp. 201–210, June 2009.

[19] F. Orhan and P. Eren, "Road hazard detection and sharing with multimodal sensor analysis on smartphones," in *Proc. of NGMAST*, Prague, Czech Republique, Sept 2013, pp. 56–61.

[20] B. Schweiger, C. Raubitschek, B. Bäker, and J. Schlichter, "ElisaTM - car to infrastructure communication in the field," *Comput. Netw.*, vol. 55, no. 14, pp. 3169–3178, Oct. 2011.

[21] J. F. Ferreira, J. Dias, "Probabilistic Approaches to Robotic Perception," *Springer Tracts in Advanced Robotics*, Springer 2014.

[22] J. Pearl, "Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference," *Morgan Kaufmann Publishers Inc.*, San Francisco, CA, USA, 1988.

[23] P. Smets, "Data fusion in the transferable belief model," in *Proc. of Information Fusion (2000)*, vol. 1, Paris, France, July 2000, pp. PS21–PS33 vol.1.

[24] T. Denoeux, "Conjunctive and disjunctive combination of belief functions induced by nondistinct bodies of evidence," *Artif. Intell.*, vol. 172, no. 2-3, pp. 234–264, 2008.

[25] P. Smets, "Decision making in the tbm: the necessity of the pignistic transformation," *Int. J. of Approx. Reason.*, vol. 38, no. 2, pp. 133 – 147, 2005.

[26] A. Buisset, B. Ducourthial, F. El Ali, and S. Khalfallah, "Vehicular networks emulation," in *Proc. of ICCCN 2010*, Zürich, Switzerland, Aug. 2010.

Jovan RADAK received his graduate engineer degree in 2005. from the University of Novi Sad, Serbia. He worked as a Research Assistant at the Department of Electronics, at the University of Novi Sad (Dec. 2005 - Aug. 2008) and as a PhD Researcher in Inria Lille – Nord Europe research center (Sep. 2008 - Dec. 2011). He obtained his PhD in Computer Science from the University of Lille I, France in Dec. 2011. In Apr. 2013, he joined Heudiasyc Laboratory at the Université de Technologie de Compiègne in France as a Postdoc Researcher. His research interests are wireless sensor networks, vehicular networks, distributed data fusion...

Bertrand DUCOURTHIAL received the Ph.D. degree in Computer Science from Paris Sud University in 1999. He joined the Université de Technologie de Compiègne and the Heudiasyc lab. He became full Professor in 2010. His research deals with dynamic ad-hoc networks (e.g., VANET, UAVs): modeling, networking, distributed algorithms, security, robustness, embedded software (Airplug Software Distribution)... His work is funded by industrial, regional, national and European projects.

Véronique CHERFAOUI received the M.S. degree in computer science from Lille University, France, in 1988 and the Ph.D. degree in control of systems from the Université de Technologie de Compiègne (UTC), France in 1992. She defended an *Habilitation á Diriger les Recherches* (HDR) in 2009. She is now an Associate Professor in the Computer Engineering Department at the UTC. Her research interests in the Heudiasyc-CNRS laboratory are multi-sensor data fusion, distributed data fusion, data association and real-time perception systems for intelligent vehicles.

Stéphanne BONNET received the M.Sc. degree in computer science and the Ph.D. degree in control engineering from the Université de Technologie de Compiègne, France. He is now a research engineer at this same institution and has contributed to several research projects, covering a wide array of related fields in embedded systems engineering, robotics and communications. His current main interests are autonomous vehicles, related vehicle-to-vehicle and vehicle-to infrastructure communication systems and their impact on future smart cities.