A Vehicle-in-the-Loop Emulation Platform for Demonstrating Intelligent Transportation Systems

Wynita Griggs, Rodrigo Ordóñez-Hurtado, Giovanni Russo and Robert Shorten

Abstract In an emerging world of large-scale, interconnected, intelligent transportation systems, demonstrating and validating novel ideas and technologies can be a challenging one. Traditionally, one is presented with a choice to make, between performing demonstrations with a few proof-of-concept "outfitted" vehicles, or experimenting with large-scale computer simulation models. In this chapter, we revisit a recent vehicle-in-the-loop (VIL) emulation platform that was developed with the goal in mind of taking steps towards countering the above validation dilemma. Roughly speaking, it was shown that a real, outfitted test vehicle, equipped with novel intelligent transportation technologies, could be "embedded" in a large-scale traffic emulation being performed with the microscopic traffic simulation package SUMO, thus allowing the real vehicle and driver to interact with thousands of simulated cars on a common road map in real-time. In our present work, we now provide an overview of the latest updates to the VIL platform, which include some enhancements to increase the platform's versatility and improve its functionality.

Wynita Griggs
University College Dublin, School of Electrical, Electronic and Communications Engineering, Belfield, Dublin 4, Ireland e-mail: wynita.griggs@ucd.ie

Rodrigo Ordóñez-Hurtado
IBM Research – Ireland, Optimisation and Control Group, IBM Dublin Technology Campus, Building 3, Damastown Industrial Estate, Mulhuddart, Dublin 15, Ireland (R. Ordóñez-Hurtado contributed towards this work while at University College Dublin, School of Electrical, Electronic and Communications Engineering, Belfield, Dublin 4, Ireland.) e-mail: rodrigo.ordonez.hurtado@ibm.com

Giovanni Russo
IBM Research – Ireland, Optimisation and Control Group, IBM Dublin Technology Campus, Building 3, Damastown Industrial Estate, Mulhuddart, Dublin 15, Ireland e-mail: grusso@ie.ibm.com

Robert Shorten
University College Dublin, School of Electrical, Electronic and Communications Engineering, Belfield, Dublin 4, Ireland e-mail: robert.shorten@ucd.ie

This work was partially supported by Science Foundation Ireland grant 11/PI/1177.

1 Introduction

Vehicle-in-the-loop (VIL) simulation is emerging as an increasingly popular tool by which to circumvent a common dilemma arising in regards to the demonstration and validation of novel intelligent transportation technologies. This dilemma concerns the fact that intelligent transportation systems (ITS) are often intended to be deployed in large urban areas or major cities, and thus gaining access to fleets of thousands of vehicles equipped with the prototype technologies and communications abilities necessary for demonstrating ITS is usually not practical nor easily achievable. On the one hand, simulators can be used to compensate and emulate large scale, but cannot accommodate for all of the unmodelled vehicle dynamics, and other complexities, uncertainties, technical issues, and driver attitudes and responses that might arise in the real world [41]; not to mention that, given the rapid development and deployment of ITS, the associated experience of being in a connected vehicle scenario for many drivers will be brand new. On the other hand, small, real-world test fleets of one to twenty vehicles can demonstrate proof-of-concept, but cannot accurately predict the outcomes of ITS applied in the context of much larger fleet sizes and city-wide scenarios [12].

In a recent paper [12], we explored a low-cost, relatively straightforward method of merging large-scale traffic simulation and the proof-of-concept capability provided by real-world vehicles. A VIL platform for embedding, in real time, a real vehicle into SUMO (Simulation of Urban MObility) was built. SUMO is an open source, microscopic road traffic simulation package primarily being developed at the Institute of Transportation Systems at the German Aerospace Centre (DLR) [23]. Utilising our VIL platform, we then demonstrated a number of experimental ITS applications that we had developed primarily at the National University of Ireland, Maynooth in Maynooth, Ireland. These applications were examples of large-scale feedback systems built upon infrastructure-to-vehicle (I2V) capabilities. A goal of our demonstrations was to illustrate how to provide drivers of real vehicles the opportunity to somewhat experience what it would feel like to be part of a large-scale connected vehicle scenario, where the rest of the traffic in the scenario was simulated by SUMO in order to avoid the necessity of large, real vehicle test fleets. In addition, our platform aims to: (i) permit feedback of real vehicle and driver responses that, in pure simulation alone, may be unmodelled, unpredicted or unexpected (e.g. delays in reaction time and imprecise vehicle control); (ii) be inexpensive to build; and (iii) illustrate potential for performing safe field demonstrations (e.g. on empty roads or test tracks) while scenarios such as wide-spread traffic congestion or accidents are simulated.

In this present work, we now aim to provide an overview of the latest updates to our VIL platform; see Section 4. Our updates include enhancements that aim to increase the platform's versatility and improve its functionality. We accompany the descriptions of these updates with some discussion on the potential ITS applications that our VIL platform could now demonstrate and validate; in particular, a

more in-depth example of a novel speed advisory system is provided in Section 5. First, however, we revisit the basic architecture of our VIL platform in Section 3, and explore some state of the art in regards to hardware-in-the-loop (HIL) and VIL simulation in general, in Section 2. Our future objectives are discussed in Section 6.

2 State of the Art

HIL simulation provides a means by which to add the complexity of a real plant under control to a virtual testing platform. Circumstances that can warrant the use of HIL simulation include: tight development schedules; practicality; development program cost savings; test safety; and the need to consider real humans and/or systems in the loop. HIL simulation has had a place in the automotive industry for quite some time; for instance, in the testing of automotive anti-lock braking systems [19], diagnostic software [22], and electric vehicle drive trains and controllers [31]. Recently, however, in addition to the established focus on testing new systems intended for integration into a single vehicle, in models of said vehicles, the concept of HIL simulation has been expanded upon such that it also encompasses the testing of entire vehicles, equipped with Advanced Driver Assistance Systems (ADAS) or ITS technologies (along with real drivers or autonomous driving capabilities), in potentially very large-scale or many multiple different traffic scenario simulations. The term being used for this kind of testing is vehicle-in-the-loop (VIL) simulation; see, for example, [16].

2.1 VIL Simulation Platforms

A number of VIL simulation platforms for testing ADAS currently exist. Examples include those described in [6, 4, 30, 20, 21], as well as in [12], and its predecessor [13]. The work in [6] particularly focused on autonomous intervening assistance systems for collision avoidance or mitigation. A VIL test setup was described in order to build upon existing validation methods such as driving simulators, traffic flow simulations, and test vehicles that collide with substitutes such as foam cubes; see, also, [5]. Similarly, with respect to accident mitigation and avoidance, it was observed in [4] that VIL testing had a place, not only in validating the technical functionality of intervening systems, but also in studying the behaviour of real drivers as they interacted with the new technology. VIL was proposed as a method to combine virtual, visual simulation with the kinesthetic, vestibular and auditory feedback of a real car driving on a real test track, and thus VIL was promoted as capable of offering a variety of new options for evaluating ADAS, and of providing a real driving experience combined with the safety and test replication abilities of driving simulators. See, also, [37].

The work of [11] presents a midway approach to ADAS prototyping and validation, laying somewhere between HIL and VIL, that the authors refer to as vehicle-hardware-in-the-loop (VeHIL). Their particular system is called SERBER. As opposed to having a driver travelling on a real test track, as was the case in [4], real vehicles in the setup of [11] are physically locked on a chassis dynamometer and thus the tests are conducted indoors. In this setup, environmental parameters such as humidity, ambient light, temperature, and so on, can be easily controlled. The chassis dynamometer is paired with multi-sensor road environment simulation software. TASS International offers a VeHIL laboratory; see [40].

In [1], the concept of subsystem development occurring in parallel across different parts of the globe (for example, in globally distributed company departments) was tackled via the introduction of a new validation concept called X-in-the-distance-loop. Typically, bringing such subsystems together physically in order to validate interactions between them, as well as the system's behaviour in total, requires effort in terms of time and costs in regards to transportation. Furthermore, additional problems concerning confidentiality may need to be taken into account. It was proposed in [1] to utilise internet connectivity in order to perform validation experiments from distributed geographical locations. As such, capabilities for real-time data transfer capabilities had to be realised, and the required information from each side defined by the developers and testers together.

The objective in [32] was to implement a VIL platform, where intersection control policies for autonomous vehicles, formerly only tested in simulation, were tested with a real autonomous vehicle that interacted with multiple virtual vehicles, at a real intersection, in real time. In such experiments, having all real autonomous vehicles (as opposed to a single, real autonomous vehicle interacting with a number of virtual vehicles) would have proven expensive; especially in the event that any of the control policies were to fail and an accident ensued. At the same time, the experiment yielded results that differed to those obtained when using a fully simulated environment.

GrooveNet, a vehicle-to-vehicle (V2V) network simulator capable of achieving communication between simulated vehicles, real vehicles, and between real and simulated vehicles, was described in [25]. With this approach, it became feasible to deploy a small fleet of vehicles in the field (e.g. in the order of a dozen) to test protocols that in truth involved hundreds or thousands of vehicles, the rest of which were simulated. GrooveNet was designed to investigate V2V issues; in particular, with respect to wireless communication issues in mobility networks.

We conclude this section by drawing attention back to our own work described in [12]. This VIL platform was developed with low cost and construction simplicity in mind, in addition to the idea of offering human drivers in real vehicles, on real roads or test tracks, opportunities to test and validate new ITS technologies in large-scale, interconnected, simulated traffic scenarios. In what follows, we briefly review our

platform's existing architecture. We will then discuss the latest updates to our work and provide example applications.

3 Platform Architecture

The first iteration of our VIL platform, described in [12], was constructed as a preliminary prototype. Its setup was simple in design and consisted of an open source microscopic traffic simulation package, emulating potentially thousands of virtual vehicles, sitting on a workstation computer in a control room; together with some ITS applications, also deployed on the workstation computer, written in Python. A real vehicle was "embedded" into our traffic simulations, the real vehicle being represented in the emulations by an avatar; and data was transferred between the real vehicle, and the workstation computer in the control room, over a cellular network via a smartphone carried with the real driver. Some simple TCP-based client/server programming sat on both the smartphone side and workstation computer side. Our open source traffic simulation package of choice was SUMO [23]. This simulator is designed to handle large road networks, and comes with a "remote control" interface, TraCI (short for Traffic Control Interface) [45], that allows one to adapt the simulation and to control singular vehicles on the fly. An illustration of the platform setup that was described in [12] is presented in Fig. 1.

Some experimental ITS applications, developed primarily at the National University of Ireland, Maynooth in Ireland, that we then proceeded to demonstrate using our VIL platform, in [12], consisted of the following: (a) an intelligent speed recommender system [29]; (b) emissions regulation, via context-aware hybrid vehicle engine mode control [36]; and (c) local rerouting around an obstruction [17]. As mentioned, Python scripts containing algorithms that were unique to each specific ITS application were deployed on the workstation computer. These Python scripts additionally contained some Transmission Control Protocol (TCP) server code for handling connections with, and data flow to/from, the smartphone; as well as some code that enabled our scripts as clients which communicated with SUMO, via TraCI, the traffic simulator itself thus performing as a server. (TraCI uses a TCP-based client/server architecture to provide access to SUMO.) These latter two aspects of our scripts (i.e. the TCP-based components) were universal with respect to all three of our ITS applications.

Let us review the setup of the platform with regards to two of these demonstrated ITS applications more closely.

Fig. 1 Illustration of the VIL platform setup described in [12]. In [12], the essential notion was that ITS applications were deployed on the workstation computer also interfacing hundreds of vehicles in SUMO, while a corresponding application was deployed on the smartphone which served as the interface to the driver and the real vehicle. The simulated vehicles, and the real vehicle, both provided inputs for the control algorithms and applications to be tested. (The real vehicle and computer sub-images contained within this illustration were downloaded from https://openclipart.org on 27 July 2016 and 22 September 2017, respectively.)

3.1 Intelligent Speed Recommender

The goal of this application was to detect approaching traffic bottlenecks, e.g. roadwork zones or traffic jams, and provide drivers with recommended travelling speeds, ensuring that vehicles travelled at safe speeds and distances from the vehicles ahead of them as they approached, entered and left the bottleneck. The notion was that a traffic bottleneck could be emulated on demand in the virtual environment, while proof-of-concept of the system was being demonstrated in the real vehicle on an

empty road. Validating the system with the VIL platform included obtaining an initial assessment of a real driver's attitudes and responses to the speed recommendations, as well as demonstrating how feedback from the real driver and vehicle could be incorporated into the evolving traffic situation.

The application algorithm consisted of two key stages. First, the traffic scenario in which the *Host Vehicle* (i.e. the vehicle of interest, receiving the recommendations) was travelling in was determined. A full list of potential traffic scenarios consisted of: Free Traffic; Approaching Congestion; Congested Traffic; Passing Bottleneck; and Leaving Congestion. An estimate of the vehicle density surrounding, and information concerning the speeds of, the Host Vehicle and the *Next Vehicle* (i.e. a point of interest along the future trajectory of the Host Vehicle) were utilised by a rule-based inference engine in order to determine the traffic scenario. Then, using this (and some additional) information, a recommended travelling speed for obtaining a safer distance to the Next Vehicle was calculated.

In our setup, in [12], the Host Vehicle was our real vehicle. For this vehicle, we utilised a 2008 Toyota Prius 1.5 5DR Hybrid Synergy Drive, pictured on the left in Fig. 2; however, any vehicle with an accessible interface or gateway to its onboard computer, such as an OBD-II diagnostic connector, would have been suitable to use. In the vehicle, we mounted a Samsung Galaxy S III mini (model no. GT-I8190N) running the Android Jelly Bean operating system (version 4.1.2). The purpose of the smartphone was to relay, over a cellular network, periodic updates from the vehicle's onboard computer in relation to its current speed, to the workstation computer in the control room running SUMO; and to receive the recommendation messages from the workstation computer and display them on a user interface for the driver. We utilised the mobile data services of a commercial mobile phone operator for the relay of data, where these services were provided using a 3G UMTS 900/2100 network.

The hardware device that we used to form the connection between the smartphone and the Prius' onboard computer (via its OBD-II diagnostic connector) was a Kiwi Bluetooth OBD-II Adaptor by PLX Devices. This device was plugged into the vehicle's OBD-II diagnostic connector and communicated the data relating to the vehicle's current speed to the smartphone via Bluetooth. A variety of existing smartphone applications were compatible for use with the Kiwi Bluetooth at the time of our experiment. We decided upon Torque Pro[1] given that an Android Interface Definition Language (AIDL) application programming interface (API) was included with it for the development of third party plug-ins. We utilised this feature to design a new plug-in for Torque Pro for our speed advisory application. The functionalities of this plug-in included: buttons for initiating and terminating communication with the workstation computer, and the associated Java socket programming for maintaining this communication and handling all of the data transfer;

[1] *Torque Pro* by Ian Hawkins. Available from Google Play: `https://play.google.com/store/apps/details?id=org.prowl.torque`. Last accessed on 28 September, 2017.

Fig. 2 Field-test vehicles.

the capability to bind to the Torque Pro service running in the background, and thus extract information in relation to the vehicle's current speed from the Prius' onboard computer; and a unique user interface that showed to the driver the vehicle's current speed, as well as the traffic scenario that the algorithm running on the workstation computer had determined that the Prius was currently travelling in, and the speed recommendation that the algorithm consequently issued to the driver. This user interface is shown in Fig. 3.

We also created a remote procedure call (RPC) framework based on TCP sockets, which allowed us to directly call MATLAB functions from our Python script that was deployed on the workstation computer. We did this because, for this particular demonstration, our ITS algorithm was initially developed using the MATLAB Fuzzy Logic Toolbox (Version 2.2.14, R2011b). Thus, by utilising the RPC framework, we were not forced to port the algorithm, developed in MATLAB, over to Python.

For our demonstration then, the Toyota Prius was driven around a circuit defined on the north campus of the National University of Ireland, Maynooth. The demonstration was performed at a time of day when the rest of the real traffic on the road was minimal (i.e. early in the morning). The road map generated for use in SUMO, and the real road circuit that the Prius travelled upon, were topographically the same. Twenty-two virtual vehicles with different characteristics (i.e. maximum speeds,

Fig. 3 Plug-in user interface for the speed recommender system demonstrated in [12].

sizes, acceleration and deceleration capabilities, etc.) were emulated in SUMO, in real time, during the Prius' test drive, along with a variety of local speed limits (lower than the real speed limits on the road), in order to create bottlenecks of traffic that involved both the simulated vehicles, and the Prius, which was represented in the emulation by an avatar. At the beginning of the test, both the Prius, and its avatar in the simulation, departed from the same positional starting point on the circuit. Updates to the Prius' position in the SUMO map were then made based (only) on its real-time travelling speed as obtained through its OBD-II diagnostic connector. Given that its real GPS coordinates, for example, were not used in making updates to its position in the SUMO map, map-matching with respect to representing the Prius' actual position (on the real road) accurately in the SUMO map, was poor. Thus, it was noted that the relay of real-time positional data would be critical for much future work, and that better map-matching was an important element yet to be included in the platform. Other results and observations from the demonstration, e.g. in relation to the driver's comfort at following the speed recommendations, were reported in [12].

3.2 Emissions Regulation

In this ITS application, the different engine modes available to hybrid vehicles were utilised to regulate emissions levels, in a fair manner for all participating vehicles, over geographical zones where maximum levels of tolerated pollution were applied. Specifically, the application sought to orchestrate the way in which each vehicle, from a large fleet of hybrids, would uniquely utilise its internal combustion versus

electric engine modes to regulate pollution, through simple communication signals from a central infrastructure. A number of algorithms were provided in [36] as a means of achieving this orchestration. In [12], the simplest of these algorithms was demonstrated; namely, a simple integral controller was applied in a stochastic framework.

Specifically, a threshold was set in regards to what the permissable level of global CO emissions to be tolerated during our traffic scenario emulation would be; and then feedback regulation was employed by a central infrastructure, which sent periodic signals to each of the hybrid vehicles participating in the scenario, such that the global CO emissions in the scenario would be regulated around the threshold in a manner that was fair to all participating hybrids. The signals that were sent by the central infrastructure pertained to a probability value, as follows. In our scenario, internal combustion engine use resulted in a vehicle emitting CO, whilst driving in fully electric mode did not. Engine mode instruction for each unique vehicle, at regular intervals in time, was dictated by a periodic "coin flip", the result of this "coin flip" (i.e. random number generation) being compared to the aforementioned probability value. This probability value reflected what the global CO emissions for the scenario were at the current moment. If current CO levels for the scenario were zero, then the probability of any participating hybrid vehicle being allowed to wholly utilise its internal combustion engine, for example, was one. This probability degraded towards zero as global CO emissions in the scenario rose. The results of the demonstration are provided in [12]. We describe, next, further specifics of the demonstration setup in regards to our VIL platform.

It was assumed that we had access to all necessary environmental information for our demonstration, such as the exact amounts of CO being emitted by each vehicle (both virtual and real cars; this information was derived using the formulae cited in [36]); and we set an artificial regional pollution threshold. For our real vehicle that was embedded in the emulation to provide the proof-of-concept aspect of the demonstration, we again elected to use the 2008 Toyota Prius 1.5 5DR Hybrid Synergy Drive, as pictured on the left in Fig. 2. Again, we utilised the Samsung Galaxy S III mini (model no. GT-I8190N) to relay, over the cellular network, periodic updates from the vehicle's onboard computer, via the Kiwi Bluetooth OBD-II Adaptor, in relation to its current speed (this speed information was utilised on the workstation computer to compute the Prius' CO emissions via the formulae cited in [36]); and to receive messages from the workstation computer in relation to engine mode orchestration for the next time step (i.e. to receive the current global probability value arising from the current global CO level).

The plug-in for Torque Pro that we developed for the application again consisted of buttons for initiating and terminating communication with the workstation computer, and the associated Java socket programming for maintaining this communication and handling all of the data transfer; and again had the capability to bind to the Torque Pro service running in the background, to extract information in relation

to the vehicle's current speed from the Prius' onboard computer. The user interface again showed the driver of the Prius its current speed; and this time, also displayed a message in relation to what engine mode the driver of the Prius should currently be employing, after the plug-in performed one of its "coin flips" and compared the random number obtained to the corresponding most recent probability value received from the workstation computer. The user interface additionally consisted of two buttons that allowed the driver to engage or disengage automatic engine mode control. When automatic engine mode control was engaged, signals were forwarded from the smartphone, via Bluetooth, to a mechanical "finger" device that was mounted inside the Prius. The "finger" physically interacted with the Prius by pushing and releasing a drive mode switch in the vehicle. This mechanical device is shown in Fig. 4. The user interface of the plug-in was shown in [12].

Fig. 4 The mechanical device.

Given that the control algorithm itself had been initially developed in MATLAB, we again made use of our RPC framework to directly call MATLAB functions from our Python script that was deployed on the workstation computer. As such, we were again not forced to port the algorithm, developed in MATLAB, over to Python. For convenience, we also utilised, in the demonstration, the same road map and SUMO configurations (with respect to the number and type of virtual vehicles in the emulation) that we did for the speed recommender application. Again, updates to the Prius' position in the SUMO map were made based (only) on its real-time travelling speed.

In our demonstration, we applied a constant threshold, in regards to permissable global CO emissions, over the entire map. However, we recognised that obtaining positional information from the real car (e.g. GPS coordinates) would be necessary if a constant threshold was not applied over the whole map; but rather, to subregions. In such cases, positional data would be needed from the vehicle to determine accurately whether or not it was inside a region where a pollution threshold was enforced. Furthermore, we noted that incorporating event-driven logic into our Python script and plug-in, to activate the application when the real vehicle entered a geographical zone where pollution monitoring was ongoing, would be ideal. Finally, we noted that, in reality, the exact quantity of CO being emitted by each vehicle would not be known to a central infrastructure. Furthermore, in more realistic scenarios, not only hybrid vehicles participating in the service would be driving around on the map. So too, driving around, would be non-participating hybrids; and also vehicles that were not hybrids and thus could not have their engine modes affected by the application. Therefore, knowledge of CO levels in a region would likely come from external measurements; for example, from a meteorological agency utilising sensors. To imitate such in our VIL platform, we noted that another ideal future enhancement to the platform would be an ability to incorporate real-time and/or real-world information in order to bring more realism or proof-of-concept capabilities to our demonstrations; for example, live news events, traffic jam reports, and weather and pollution information from meteorology organisations, could be taken into account in generating or validating test scenarios.

3.3 On the Existing Architecture

While we succeeded, in [12], in developing a preliminary prototype of our VIL platform, and demonstrated some ITS applications, and thus exhibited the concept of the VIL platform that we aimed to construct, it is clear from the discussion above that a number of ideal components or enhancements to the VIL platform remained yet to be added. In particular:

(i) our original design permitted only a single, real vehicle to be embedded into the platform (this issue was not discussed above but will be explored in Section 5);

(ii) access to real-world information was limited to what the real vehicle's on-board diagnostics (OBD-II) and/or what the smartphone, carried on board the real vehicle with the driver, could provide;
(iii) our Python scripts were not modular, in that they did not exhibit a plug-and-play structure, in regards to what ITS applications were required by a user at any given point in time;
(iv) our Python scripts were not event-driven, with respect to what ITS applications were in-use, or active, in a traffic scenario at any given point in time;
(v) and our platform exhibited poor map-matching in regards to the incorporation of real-world positional information from the embedded hardware into the platform.

Thus, in this current work, we now aim to report on some improvements that we have made to our VIL platform since its inception in [13] and [12]. These improvements are as follows:

(I) multiple real vehicles can now be embedded through threading and a ticketing system;
(II) real-world information is available from a range of different sources;
(III) we have moved towards code modularity and event-driven capabilities;
(IV) and we have improved map-matching for those applications that are location-aware.

In the next section, we describe these improvements in more detail. Our aim is to maintain, as much as possible, a low cost in regards to platform construction, and ease of availability in terms of the components required, while also extending upon the range of ITS applications that the platform may be used to demonstrate and validate.

4 New Components and Enhancements

An account of the latest components added to the VIL emulation platform follows. As mentioned, our aim is to, through the addition of new features and enhancements, increase the platform's versatility and improve its functionality.

4.1 Scalability: Multiple Real Vehicle Embedding

The original design of our platform permitted only a single real vehicle to be embedded into a SUMO simulation. To add more real vehicles, we exploited the fact that, in the parent process of our Python scripts, TraCI provides access to SUMO via a single port; and multiple real vehicles can be represented in SUMO by different vehicle IDs. Thus, multiple real vehicles can be embedded in a single simulation.

Threading, and the use of a ticketing system, were implemented in the server subprocess of our Python application scripts (i.e. the subprocess that communicates with the smartphones). Threading permits the handling of multiple client calls at once. The procedure was as follows. A single port is reserved for communication with the smartphones in our server subprocess. When a new real vehicle joins the network and wishes to utilise the ITS application under test, it communicates its presence via its smartphone to the server. Once the server has accepted the connection, the real vehicle is provided with a unique ticket for identification purposes; parameters associated with the real vehicle (e.g. its speed and location) are initialised; and, from there, communication with the real vehicle is passed to and dealt with in a newly created, personal, background thread for continued interaction between the server and client. That is, the real vehicle can now send continued data regarding its state, and the server can pass back information continually, such as alerts and recommendations.

Finally, we note that this modification to the platform permits, not only multiple real vehicles to be added to a single simulation such that they may all drive around in it concurrently; but also enables single real vehicles that leave the scenario and need to re-enter it again later, at a future time step, to be successfully re-added when (or each time that) they do. We will explore a benefit of being able to embed multiple real vehicles in the use case presented in Section 5.

4.2 Information Exchange: Additional Sensors and Other Devices

In [12], our platform architecture utilised the OBD-II diagnostic connector of a real vehicle to access data from its engine control unit (ECU). The particular data accessed is typically application-driven, and might consist of the vehicle's speed, the state of charge of its high voltage battery, etc. The type and amount of data available to ITS application testing need not be limited to what the OBD system can provide, however. Real vehicles can be equipped with additional sensors and devices. Furthermore, other forms of real-world and/or real-time data can be obtained (for example, from meteorological organisations) and incorporated into testing, all in order to enable the available information about the real vehicle, driver, and the real vehicle and driver's surroundings, to be expanded upon. Some of these additional devices and/or the data that they provide can be incorporated into the VIL platform and used for validating a wider range of ITS applications in a more comprehensive or realistic manner. Thus, we now discuss some new sensors and other devices that we have utilised recently, in terms of extending the capabilities of our VIL platform.

Smartphones

Besides acting as devices through which to relay, to a workstation computer over a cellular network, information obtained from a real vehicle via its OBD-II diagnostic connector; and as devices through which to receive messages from a workstation computer and display them on a user interface for a driver; smartphones have the capacity to provide other uses as well. By themselves, for example, they carry their own sensor compliments, such as GPS and accelerometers, and can thus provide extended information in addition to what is available from the real vehicle's ECU. They can receive and process human input (e.g. voice, touch or text) through a user interface, as provided by a driver. Smartphones have processing power and can run a range of applications. Some applications may involve the preprocessing of data to be sent to a workstation computer; and/or involve computations, algorithms and actions concerning the data received from a workstation computer. Other applications (or subprocesses of an application) may involve processing information received from other, additional devices onboard a real vehicle, or carried by the driver, such as the wearable technology that we describe next.

Wearable Technology

With the rise of the Internet of Things (IoT), wearable technology (i.e. clothing and accessories incorporating computer and advanced electronics, such as Microsoft Band [26], Apple Watch [3] and Android Wear [2]) has become readily available. Wearables can provide additional information regarding the state of a real driver; for example, Microsoft Band and Apple Watch contain heart rate sensors. For instance, recently, at University College Dublin, in Belfield, Ireland, in collaboration with IBM Research – Ireland, one of our team's research projects has consisted of embedding an electric bicycle into a SUMO simulation and artificially creating regions where transport-related pollution is high (e.g. a busy section of road with many buses and trucks) [38, 35]. The bike is fitted with a controller that increases the output from the electric motor in order to aid the pedalling bike rider in powering the bike when the rider enters one of these "tail-pipe" polluted areas. The aim is to keep the bike rider's breathing rate lower so that they do not inhale as much of the air pollution. To monitor the heart and ventilation rates of the bike rider, they were fitted with ventilation masks (i.e. a COSMED Spiropalm 6MWT, with a fully integrated pulse oximeter) during the application testing, which took place both on a real road and in a laboratory. A video demonstrating the proposed system can be found at [39].

Gas Sensors and Weather Stations

In the emissions regulation example described in Section 3, we briefly mentioned the notion of incorporating real-world weather or pollution information, as mea-

sured by meteorology organisations for example, in order to create more realistic test case scenarios or proof-of-concept demonstrations. Gas sensors, e.g. [8], can be fitted to real vehicles embedded in simulations for validation purposes too. As an alternative example application, consider [46], wherein the source localisation problem for a natural gas leak in an urban setting was considered. As an extension to this work, to provide proof-of-concept and improve upon the theoretical models presented within it, some members of our research team, in collaboration with IBM Research – Ireland, have since performed experiments using real CO_2 sensors [8] positioned in a two-dimensional space (as well as fitted on parked cars), and demonstrated gas leaks coming from different point sources. Experimentation with the sensors was possible due to the ease of incorporating the compact modules as add-on components of microprocessor-based equipment via serial communication, and also due to their technical features such as low power consumption, wide measurement range, fast response rate and high accuracy. The solution approach to the source location problem then involved using the information obtained from the CO_2 sensors. The measurements from the sensors were combined with wind information from weather stations and processed with a source localisation algorithm based on advection-diffusion equations [46]. It is worth noting that an off-the-shelf professional weather station, such as the one used in our investigations [44], not only provides wind information from a wind metre, but also incorporates a temperature sensor, humidity sensor, barometric pressure sensor, light sensor and a rain gauge, which allows it to be used in applications such as weather estimation and weather forecasting.

RFID

Radio-frequency identification (RFID) is a generic term for technologies that use radio waves to automatically identify entities [34]. Several methods of identification exist, but the most common is to store a serial number that identifies an entity on a microchip that is attached to an antenna. The chip and antenna together are called an RFID transponder or tag. The antenna enables the chip to transmit the serial number to a reader. The reader converts the radio waves reflected back from the RFID tag into digital information that can then be passed on to a computer to make use of. A passive RFID tag draws power from the field created by the reader and uses it to power the microchip's circuits. The chip modulates the waves sent back to the reader by the tag, and the reader converts these new waves into digital data. In this way, passive tags do not require a local power source. In addition, unlike barcodes, RFID tags do not need to be within the line of sight of the reader.

In some recent experiments, we utilised RFID in combination with our VIL platform to study problems relating to the localisation of missing entities by considering parked vehicles as service delivery platforms [43, 9]. Specifically, the notion was to utilise vehicles that are parked for extended periods of time in dense, urban areas to detect and track moving, missing objects using RFID technology. Such entities

might consist of a missing patient with dementia, a lost pet or a stolen vehicle; e.g. a bicycle. The entities carry with them a passive RFID tag, while the readers are located with the parked vehicles. The advantages of using a network of parked vehicles to locate a missing entity include, first, the sheer number of vehicles that are owned by people, and the fact that, for 96% of the time on average, these vehicles are parked [33]. In other words, the network is large. The network also does not require dedicated maintenance, and technology upgrades are easy [24]. Additionally, energy infrastructure and planning permissions are not required to establish the network. The use of the VIL platform permits us to experiment with a range of algorithms; e.g. algorithms concerning the density of vehicles actively searching for an RFID tag at any point in time versus the time taken until localisation of the missing entity occurs. One such algorithm was provided in [43]. Realistic parking data from urban environments can be utilised and represented in an emulated scenario while proof-of-concept of the system in real life and in real time is being demonstrated. Preliminary results were provided in [43]. Further investigations are ongoing.

Another, related, recent use of RFID in research concerned the implementation of a low-cost, low-power, easily integrated cyclist collision prevention system for in-car deployment. The result of another collaboration between University College Dublin and IBM Research–Ireland, this system sought to inform car drivers of nearby cyclists, based on cyclists being equipped with passive RFID tags, and cars containing readers. Further details can be found in [27]. Finally, we mention [18], wherein geo-fences (i.e. virtual geographic boundaries) were employed to specify areas of low pollution around cyclists. Similar to the emissions regulation application as discussed in Section 3, the emissions level inside a geo-fence was controlled via a coin tossing algorithm that determined the engine mode to be next employed by a participating hybrid vehicle. The algorithm was triggered when a vehicle detected a cyclist with RFID. The system was demonstrated with our VIL platform.

High-Precision GPS Information

We have utilised the high-precision, DGPS/AGPS-enabled, GPS travel recorder, Qstarz BT-Q1000XT [7], in our investigations to enhance the position estimation capabilities of real vehicles; and with the smartphone application Torque Pro, which permits the use of external (Bluetooth) GPS devices, we have been able to pass such information to the VIL emulation platform. This makes the localisation of avatars in SUMO a more reliable process, in terms of representing real-life events (i.e. the position of the embedded cars, in this case) with a higher level of accuracy.

4.3 Map Matching: Speed and Position Corrections

The incorporation of real-world information from the embedded hardware into the VIL emulation platform should be performed as accurately as possible, so that the real-world information is represented with high fidelity in the simulated experiments. Two important variables obtainable from an embedded vehicle are its position and speed. Thus, we pay special attention to those next.

Concerning information pertaining to a real vehicle's speed, it is been reported that the speed measurements obtained from an ECU do not generally match the speed on the car dashboard display, and this has two main causes: i) a typical (intentional) over-read in most standard speedometers; and ii) a different wheel diameter from the factory specification [42]. GPS speed measurements are also available, for example, from a smartphone, and optionally via Torque Pro, as an alternative.

In the case of position information, the main data source is GPS localisation. However, GPS measurements are highly sensitive to environmental and/or contextual factors, and they can be strongly degraded by atmospheric effects (e.g. ionospheric delay), multipath effects (e.g. from buildings), outdated ephemeris data, and the GPS receiver quality (e.g. inefficient algorithms/circuitry). Accordingly, we have proposed different methods for mitigating the position errors from traditional GPS measurements. These methods include the use of:

1. a high-precision, DGPS/AGPS-enabled GPS receiver, e.g. [7], which delivers more accurate GPS measurements than the average smartphone's GPS unit;
2. Kalman filtering, which improves the position estimation accuracy by fusing GPS information (e.g. latitude, longitude, precision) with sensor data (e.g. speed from OBD-II, bearing from smart device gyroscope) and aggregated data (e.g. bearing estimation from previous GPS locations, speed from GPS);
3. passive RFID tags, which are (intentionally) placed at dedicated road-side locations, and can transmit their position information once they are detected by the car;
4. map matching, with respect to the road network used in the simulated environment, using the method `traci.vehicle.moveToXY`[2];
5. and/or parked vehicles, used as anchor nodes for cooperative positioning (given the availability of V2V systems) [28].

Of the above methods, (1), (2) and (3) have been successfully implemented and tested, while (4) is currently being implemented, and (5) is planned for a future design.

[2] Under typical conditions, SUMO only allows vehicles to be on roads, and thus if an attempt is made to move a vehicle to an arbitrary location, then SUMO will place it on the nearest road to that location, taking into account certain filters. One of these filters uses bearing information so that the vehicle is map-matched to the nearest road with an orientation similar to the bearing provided by the user.

4.4 Transmission Frequency and Code Modularity: Application Logic

In the first iteration of our platform design, once a connection was formed, information between the real vehicle and the workstation computer was transmitted once per second. This steady, periodic rate was maintained throughout the duration of a simulation for simplicity. Changing the information transmission frequency, e.g. to different, event-inspired transmission frequencies, is something that makes sense for a number of applications, however. For instance, in the "detect and track a moving, missing object using RFID technology" application described in Section 4.2, one transmission frequency might be utilised while the detection stage of the application is active, whilst a different transmission frequency could be preferential once the object is located and the tracking function kicks in. Investigations regarding this are currently ongoing. As part of the ENABLE-S3 project [10] (an ongoing project funded by the ECSEL Joint Undertaking, which in turn receives support from the European Union's HORIZON 2020 research and innovation programme, as well as from a number of European countries), we are additionally continuing the development of the VIL platform such that ITS applications under test may be inserted in a 'plug and play' capacity and activated, deactivated or modified in behaviour on demand.

5 A Final Illustrative Use Case

In this section, we conclude the discussion on our VIL platform improvements by returning to the notion of embedding multiple real vehicles into a single emulation. These vehicles could be embedded either concurrently or consecutively, the latter instance also being applicable to where there might be only a single real vehicle, but it is to exit and enter a scenario multiple times. We will now provide a use case example of a recently proposed speed advisory system (SAS) [14, 15] in which embedding multiple real vehicles (as opposed to a single one) is a necessary requirement for adequately validating the system when utilising the VIL platform. The reasoning behind this is presented below, in Section 5.1.

The objective of the SAS is to implement a consensus algorithm to guide a set of vehicles towards a common driving speed [15]. An innovation of the SAS is that consensus is achieved over a multi-layer network, where parallel network topologies of connected vehicles are superimposed. The reason for the use of these parallel networks is that, in this way, state obfuscation is possible, with the benefit that common driving speed is attained with no vehicle knowing the exact state of other vehicles participating in the service. The SAS can be demonstrated using our VIL platform, where we would have (for instance) the two real field-test cars shown in Fig. 2 participating in the service along with many simulated vehicles. We direct the reader to

[15] for further details on the SAS algorithms, and focus our attention here on the characteristics of the VIL demonstration instead.

In our set-up, we considered the road network of the University College Dublin campus in Belfield, Ireland. A map of the campus was imported into SUMO from OpenStreetMap. A maximum allowed link speed of 30km/h was set on the campus roads to reflect real-world speed limits, and a maximum allowed link speed of 50km/h was set on roads on the approach to the campus (i.e. on links on the approach to the campus entrance gates). The road network is shown in [15, Fig. 4]. In regards to the simulated traffic in our demonstration, we emulated a morning rush, with passenger vehicles randomly being allocated to enter one of three possible entrance gates, and then making a random decision to head to one of three possible car parks (see [15, Fig. 4]). Vehicles entered the network at a rate of one vehicle per twenty seconds, for twenty minutes, at maximum permitted departure speed. Other attributes that we assigned to the simulated vehicles can be found in [15].

In a certain test case scenario of interest to our current discussion, no vehicle (simulated or real) was instructed to be a leader, and thus speeds were expected to converge towards the average. The time step size used in the simulation was 0.1s; however, information was only exchanged every 1s between the workstation computer and the smartphones carried in the real vehicles. A VIL emulation was performed to test the convergence of the SAS.

5.1 Discussion

In the case where only a single real vehicle is embedded in the VIL platform while the SAS described above is under test, a potential arising issue is that the real vehicle can unintentionally become a *leader*. This is because a real driver's ability to follow speed advice is often less precise than a simulated car's ability. Thus, the real vehicle pulls the convergence of the algorithm in an up or down manner depending on its own speed value. Having multiple real vehicles embedded, which is now possible with our VIL platform, paves the way to negating this issue, supposing that all real drivers are attempting to follow the speed advice.

In addition, as mentioned prior, our platform now also permits for vehicles that leave a scenario and return to it later, to be "re-embedded" when (and each time) that they do. Not only is this an important feature for fault tolerance, like intermittent connection loss between a real vehicle and the workstation computer; but it also permits for us to accommodate situations, for example, where an application (like the above SAS) is active in a particular geographical zone, and real drivers are entering and exiting that zone multiple times throughout a demonstration. Each time that they do enter the zone, they can be reincorporated into the scenario as participants.

Finally, the methods for mitigating positional errors obtained from traditional GPS measurements that were described in Section 3 give us an improved capacity to map match and thus track the positions of the real vehicles on the road network in SUMO.

6 Conclusions and Future Work

The addition of new features and enhancements to our VIL emulation platform increases its versatility and improves its functionality in regards to ITS application demonstration and validating. Future features and enhancements that we would like to add include the capacity for our platform to be able to incorporate and handle V2V communications. To achieve this, the integration of our platform with a network simulator, and an ability of the platform to process information from real onboard units and roadside units (i.e. communication devices), will be necessary.

References

1. Albers A, You Y, Klingler S, Behrendt M, Zhang T, Song K (2014) Supporting globally distributed product development with a new validation concept. Procedia CIRP 21:461–466
2. Google Inc. (2017) Android Wear. In: Wear. Android. https://www.android.com/wear Cited 9 Oct 2017
3. Apple Inc. (2017) Watch. http://www.apple.com/uk/watch Cited 9 Oct 2017
4. Berg G, Nitsch V, Färber B (2015) Vehicle in the loop. In: Winner H, Hakuli S, Lotz F, Singer C (ed) Handbook of Driver Assistance Systems: Basic Information, Components and Systems for Active Safety and Comfort, Springer International Publishing Switzerland
5. Bock T, Maurer M, Färber G (2007) Validation of the vehicle in the loop (VIL) – a milestone for the simulation of driving assistance systems. In: IEEE Intelligent Vehicles Symposium 612–617 Istanbul, Turkey
6. Bock T, Maurer M, Färber G (2007) Vehicle in the loop (VIL) – a new simulator set-up for testing Advanced Driving Assistance Systems. In: Driving Simulation Conference – North America (DSC–NA) Iowa City, Iowa, USA
7. Qstarz International Co., Ltd. (2013) BT-Q1000XT. In: GPS Travel Recorder. Products. http://www.qstarz.com/Products/GPS%20Products/BT-Q1000XT-F.htm Cited 11 Oct 2017
8. CO2Meter.com (2017) K-30 10,000ppm CO2 Sensor. In: Products. http://www.co2meter.com/products/k-30-co2-sensor-module Cited 9 Oct 2017
9. Cogill R, Gallay O, Griggs W, Lee C, Nabi Z, Ordonez R, Rufli M, Shorten R, Tchrakian T, Verago R, Wirth F, Zhuk S (2014) Parked cars as a service delivery platform. In: International Conference on Connected Vehicles and Expo (ICCVE) 138–143 Vienna, Austria
10. ENABLE-S3. http://www.enable-s3.eu Cited 11 Oct 2017
11. Galko C, Rossi R, Savatier X (2014) Vehicle-hardware-in-the-loop system for ADAS prototyping and validation. In: International Conference on Embedded Computer Systems: Architectures, Modeling and Simulation (SAMOS XIV) 329–334 Samos, Greece
12. Griggs WM, Ordóñez-Hurtado RH, Crisostomi E, Häusler F, Massow K, Shorten RN (2015) A large-scale SUMO-based emulation platform. IEEE Trans Intell Transport Syst 16(6):3050–3059

13. Griggs WM, Shorten RN (2013) Embedding real vehicles in SUMO for large-scale ITS scenario emulation. In: International Conference on Connected Vehicles and Expo (ICCVE) 962–963 Las Vegas, Nevada, USA
14. Griggs W, Russo G, Shorten R (2016) Consensus with state obfuscation: an application to speed advisory systems. In: 19th IEEE International Conference on Intelligent Transportation Systems (ITSC) 2506–2511 Rio de Janeiro, Brazil
15. Griggs W, Russo G, Shorten R (2017) Lead and leaderless multi-layer consensus with state obfuscation: an application to distributed speed advisory systems. IEEE Trans Intell Transport Syst, doi: 10.1109/TITS.2017.2700199
16. Hammerschmidt C (2015) Vehicle-in-the-loop speeds automotive design cycles. In: News. eeNews Europe Automotive. http://www.eenewsautomotive.com/news/vehicle-loop-speeds-automotive-design-cycles Cited 11 Sep 2017
17. Häusler F, Ordóñez-Hurtado RH, Griggs WM, Radusch I, Shorten RN (2014) Closed-loop flow regulation with balanced routing. In: International Conference on Connected Vehicles and Expo (ICCVE) 1054–1055 Vienna, Austria
18. Herrmann A, Liu M, Shorten R (2017) A new take on protecting cyclists in smart cities. Available at: https://arxiv.org/abs/1704.04540v3 Cited 10 Oct 2017
19. Hwang T, Roh J, Park K, Hwang J, Lee KH, Lee K, Lee S-J, Kim Y-J (2006) Development of HILS systems for active brake control systems. In: SICE-ICASE International Joint Conference 4404–4408 Busan, Korea
20. IPG Automotive GmbH (2017) VIL systems. In: Products & Services. IPG Automotive. https://ipg-automotive.com/products-services/test-systems/vil-systems Cited 15 Sep 2017
21. IPG Automotive GmbH (2016) Vehicle-in-the-Loop. In: News. IPG Automotive. https://ipg-automotive.com/news/article/vehicle-in-the-loop-1 Cited 15 Sep 2017
22. King PJ, Copp DG (2004) Hardware in the loop for automotive vehicle control systems development. In: UKACC Control Mini Sumposia 75–78 Bath, UK
23. Krajzewicz D, Erdmann J, Behrisch M, Bieker L (2012) Recent development and applications of SUMO – Simulation of Urban MObility. International Journal on Advances in Systems and Measurements 5(3,4):128–138
24. Liu N, Liu M, Lou W, Chen G, Cao J (2011) PVA in VANETs: stopped cars are not silent. In: 30th IEEE International Conference on Computer Communications (INFOCOM) 431–435 Shanghai, China
25. Mangharam R, Weller D, Rajkumar R, Mudalige P, Bai F (2006) GrooveNet: a hybrid simulator for vehicle-to-vehicle networks. In: 3rd Annual International Conference on Mobile and Ubiquitous Systems – Workshops (MobiQuitous) San Jose, California, USA
26. Microsoft Corporation (2017) Microsoft Band. https://www.microsoft.com/Microsoft-Band/en-us Cited 9 Oct 2017
27. Ó Faoláin C, Souza M, Verago R, Shorten R (2016) Cyclist collision prevention system for in-car deployment. In: 21st Annual Society for Design and Process Science International Conference (SDPS-2016) 284–290 Orlando, Florida, USA
28. Ordóñez-Hurtado RH, Crisostomi E, Shorten RN (2017) An assessment on the use of stationary vehicles to support cooperative positioning systems. Int J Contr, doi: 10.1080/00207179.2017.1286537
29. Ordóñez-Hurtado RH, Griggs WM, Massow K, Shorten RN (2014) Intelligent speed advising based on cooperative traffic scenario determination. In: Waschl H, Kolmanovsky I, Steinbuch M, del Re L (ed) Lecture Notes in Control and Information Sciences: Optimization and Optimal Control in Automotive Systems 455:77–92 Springer International Publishing Switzerland
30. Pfeffer R, Leichsenring T (2016) Continuous development of highly automated driving functions with vehicle-in-the-loop using the example of Euro NCAP scenarios. In: Gühmann C, Riese J, von Rüden K (ed) Simulation and Testing for Vehicle Technology, 7th Conference, Berlin, Springer International Publishing Switzerland

31. Poon JJ, Kinsy MA, Pallo NA, Devadas S, Celanovic IL (2012) Hardware-in-the-loop testing for electric vehicle drive applications. In: 27th Annual IEEE Applied Power Electronics Conference and Exposition (APEC) 2576–2582 Orlando, Florida, USA
32. Quinlan M, Au T-C, Zhu J, Stiurca N, Stone P (2010) Bringing simulation to life: a mixed reality autonomous intersection. In: IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 6083–6088 Taipei, Taiwan.
33. RAC Foundation (2012) Spaced out: perspectives on parking policy. In: Press. http://www.racfoundation.org/media-centre/spaced-out-press-release Cited 10 Oct 2017
34. RFID Journal (2017) Frequently asked questions. In: FAQs. Tools & Resources. http://www.rfidjournal.com/site/faqs Cited 10 Oct 2017
35. Russo G, Shorten R (2017) Hacking an e-bike to help cyclists avoid breathing in polluted air. In: Blog. IBM Reserach. https://www.ibm.com/blogs/research/2017/08/hacking-an-e-bike Cited 9 Oct 2017
36. Schlote A, Häusler F, Hecker T, Bergmann A, Crisostomi E, Radusch I, Shorten R (2013) Cooperative regulation and trading of emissions using plug-in hybrid vehicles. IEEE Trans Intell Transport Syst 14(4):1572–1585
37. Sieber M, Berg G, Karl I, Siedersberger K-H, Siegel A, Färber B (2013) Validation of driving behavior in the vehicle in the loop: steering responses in critical situations. In: 16th International IEEE Conference on Intelligent Transportation Systems (ITSC) 1101–1106 The Hague, The Netherlands
38. Sweeney S, Ordonez-Hurtado R, Pilla F, Russo G, Timoney D, Shorten R (2017) Cyber-physics, pollution mitigation, and pedelecs. Available at: https://arxiv.org/abs/1706.00646v2 Cited 9 Oct 2017
39. Sweeney S (2017) Cyber-physics, pollution mitigation and pedelecs. In: YouTube. https://www.youtube.com/watch?v=265u9KO-9QE Cited 9 Oct 2017
40. TASS International (2017) Vehicle hardware-in-the-loop. In: Services. TASS International. https://www.tassinternational.com/vehicle-hardware-loop Cited 14 Sep 2017
41. Tielert T, Killat M, Hartenstein H, Luz R, Hausberger S, Benz T (2010) The impact of traffic-light-to-vehicle communication on fuel consumption and emissions. In: Internet of Things (IoT) Tokyo, Japan
42. Torque (2012) Speedometer. In: Torque Wiki. https://torque-bhp.com/wiki/Speedometer Cited 11 Oct 2017
43. Verago R, Naoum-Sawaya J, Griggs W, Shorten R (2015) Localization of missing entities using parked cars. In: International Conference on Connected Vehicles and Expo (ICCVE) 40–41 Shenzhen, China
44. Birstall Garden & Leisure (2017) Weathereye professional touch screen weather station with PC interface. In: Products. http://www.birstall.co.uk/products/wea22.html Cited 10 Oct 2017
45. Wegener A, Piórkowski M, Raya M, Hellbrück H, Fischer S, Hubaux J-P (2008) TraCI: an interface for coupling road traffic and network simulators. In: 11th Communications and Networking Simulation Symposium (CNS) 155–163 Ottawa, Canada
46. Zhuk S, Tchrakian TT, Moore S, Ordóñez-Hurtado R, Shorten R (2016) On source-term parameter estimation for linear advection-diffusion equations with uncertain coefficients. SIAM J Sci Comput 38(4): A2334-A2356