Peer-reviewed version available at *Sensors* **2017**, *17*, 2109; doi:10.3390/s17092109

*Article*

# Co-simulation framework for on-chip LiDAR sensors in a cyber-physical system

**Fernando Castano [1,*], Gerardo Beruvides [1], Rodolfo Haber [1] and Antonio Artuñedo [1]**

[1] Centre for Automation and Robotics, UPM-CSIC, Arganda del Rey, Spain
* Correspondence: fernando.castano@car.upm-csic.es; Tel.: +34-918-717-050

**Abstract:** Collision avoidance is an important feature in advanced driver-assistance systems, aiming at providing correct, timely and reliable warnings before an imminent collision (objects, vehicles, pedestrians, etc.). A co-simulation framework is proposed in this paper to address the design and evaluation of collision avoidances in a cyber-physical system. The co-simulation framework is supported on the interaction between SCANeR and Matlab/Simulink. From the best of authors' knowledge, two main contributions are reported in this paper. Firstly, the modelling and simulation of virtual on-chip LIDAR sensors in a cyber-physical system (CPS) considering traffic scenarios is presented. The CPS is designed and implemented in SCANeR. Secondly, an obstacle recognition library with three specific Artificial Intelligence-based methods is also designed based on sensory information database provided by SCANeR. Three methods for collision avoidance detection are considered, i.e.; a multi-layer perceptron neural network, a self-organization map and a support vector machine. Finally, a comparison among these methods for detecting obstacles before different weather conditions is done with very promising results in terms of accuracy. The best results are achieved using the multi-layer perceptron in sunny and fog conditions, the support vector machine in rainy and self-organized map in snowy conditions.

**Keywords:** sensor-in-the-loop; co-simulation framework; virtual CPS; on-chip LiDAR; obstacle recognition library

## 1. Introduction

Recent developments demonstrate an increasing efficiency, availability and affordability of sensors, data acquisition systems, and computer networks [1]. Cyber-Physical Systems (CPSs) are still growing in different engineering fields supporting applications across industries, such as: manufacturing, healthcare, electric power grids, agriculture, and transportation. Nowadays, dozens of contributions are reported in the literature addressing key CPSs issues [2] from connecting topologies up to cognitive and self-configuration layers. A detailed review of CPS-architecture functionalities is presented in [3] describing the integration of sensors, actuators and protocols [4]. Big data analytics and cloud computing platforms are reported in [5] based on methods, software, and computer-based infrastructures. The analysis of current practices to predict future behaviour and provide security to components, machines and infrastructures are presented in [6]. The concept to achieve greater storage for historic data to predict future trends is plausible, however new sensory data processing and decision-making technologies are required [7]. The computational requirements in relation to operating systems, programming languages, user interfaces, and networking technologies have become more sophisticated in relation to software managing, information flow control, error control, redundancy, reliability and latency in heterogeneous global networks [8]. Furthermore, the knowledge acquisition, the learning [9] and its transformation into physical actions to help machines in decision-making activities [10] are priorities in the paradigm of CPSs.

One of the main application fields for sensoring systems and CPSs is the realistic scenarios for advanced driver-assistance systems (ADAS) and autonomous vehicles (AV) [11-13]. According to the Accenture LLP report, for the next ten years and beyond, the key areas into the automotive vehicle

industry are: (i) cyber security; (ii) product liability for sensors and software and/or algorithms and (iii) insuring AV infrastructure [14]. In particular, sensor-in-the-loop system has been reported in several studies, with promising expectations in relation to high accuracy and precise six degree-of-freedom position information for real-time navigation [15, 16]. Many vision based navigation algorithms are also available for AV systems [17, 18]. Among them, the Laser Imaging Detection and Ranging (LiDAR) and stereo vision cameras are widely used in computer vision for autonomous vehicle applications [19-21].

Obstacle recognition serves to represent the common patterns of road, lane marking, traffic signals, vehicles, pedestrians, etc. Nowadays, many classifiers rely on machine-learning approaches to exploit data redundancy and abundance to finding out patterns, trends and relations amongst the input attributes and the class labels [22]. Within obstacle recognition techniques, vector support machines have been widely applied for classification and regression problems [23, 24]. An interesting application using machine learning for the pedestrian detection in autonomous vehicles based on HD 3D LiDAR is reported in [25], providing more accurate data to be successfully used in any kind of lighting conditions.

By the other hand, co-simulation frameworks take into account physical dynamics, control software, computational platforms, and communication networks, which is crucial for designing CPSs for autonomous driving [26, 27]. Co-simulation is essential for CPSs due to virtual prototyping, capable of properly emulate actor-sensor nodes with their own hardware specifications [28]. Moreover, virtual prototyping can take advantage of different modelling languages/tools and integrate them together for evaluating the behaviour of CPSs. For example, processing elements with real-time operating systems [29], communication systems, sensors, actuators, model transformations to the final virtual prototype [30] and localization error estimation and compensation [31] can be efficiently represented and modelled.

This paper introduces a co-simulation framework for modelling and simulating a virtual sensor network in a cyber-physical system for obstacle recognition in driving assistance. From the best of authors knowledge, two contributions are reported in this paper. Firstly, a simulation framework of virtual sensors for improving the accuracy of on-chip LiDAR sensors is presented. This simulation framework enables to get in-parallel data from connected sensor networks in a CPS. Secondly, the design and application of a library with three Artificial Intelligence-based methods for obstacle recognition in the co-simulation framework is reported. A multilayer perceptron, a self-organized map and a support vector machine are chosen due to the solid mathematical foundations, demonstrated ability in modelling in complex scenarios and a wide range of successful applications reported in literature.

Finally, an obstacle database in the CPS for driving assistance was generated from the SCANeR simulator to assess the on-chip LiDAR sensor accuracy before different weather conditions. On-chip LiDAR concept has driven to a great technological challenge with regard to sensor networks in CPSs. Due to its limited measurement range and field of view, it is necessary to obtain in-parallel data from other connected sensors to set a more accurate scan of the whole environment.

The paper consists of five sections. Following this introduction, the second section explains the co-simulation framework composed by two modules: SCANeR and Simulink. Subsequently, a case study based on the interaction between SCANeR simulator and MatLab in driving assistance scenario is explained in section 3, as well as the first preliminary results obtained. After, experimental results and discussion by means of a comparative study are addressed in section 4. Finally, the conclusions and future research steps are presented.

## 2. CPS co-simulation framework description

The CPS co-simulation framework consists in a computer-aided system to enable an efficient interaction between SCANeR studio and Matlab/Simulink. A set of computational procedures in the computer-aided system is in charge of adapting and transferring sensory information from SCANeR to MatLab and vice versa. The transfer of data is carried out through different functionalities available in SCANeR studio.

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



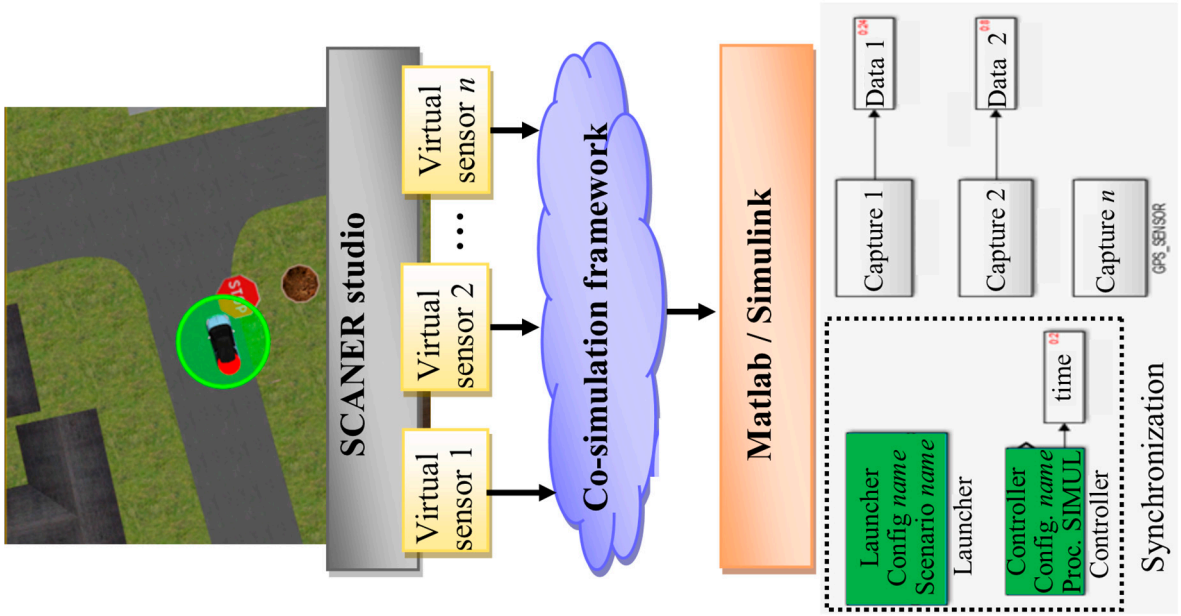



**Figure 1**. Co-simulation framework architecture. Interaction between SCANeR and SIMULINK

The co-simulation framework is implemented using the software developer kit of SCANeR. For example, some of the available functions are created with C/C++, C#, Labview, Matlab/Simulink or Python. LiDAR, 3D Stereo vison cameras and GPS sensors can be emulated with this software.

A two module co-simulation system is introduced in this work. The first module is used to generate multiple 3D traffic scenarios composed by different nodes that belong to a sensor network by using SCANeR. The second module with three specific Artificial Intelligence-based methods (i.e., artificial neural network, self-organized map and support vector machine) is implemented in Matlab/Simulink. A classifier is then derived from data cloud points given by virtual sensors in the CPS (see Figure 1). Matlab/Simulink can easily interact with SCANeR studio, even in real time, thought a SDK module. The main goal of the classifier system is the detection of multiple obstacles in traffic scenarios.

### *2.1. SCANeR studio module*

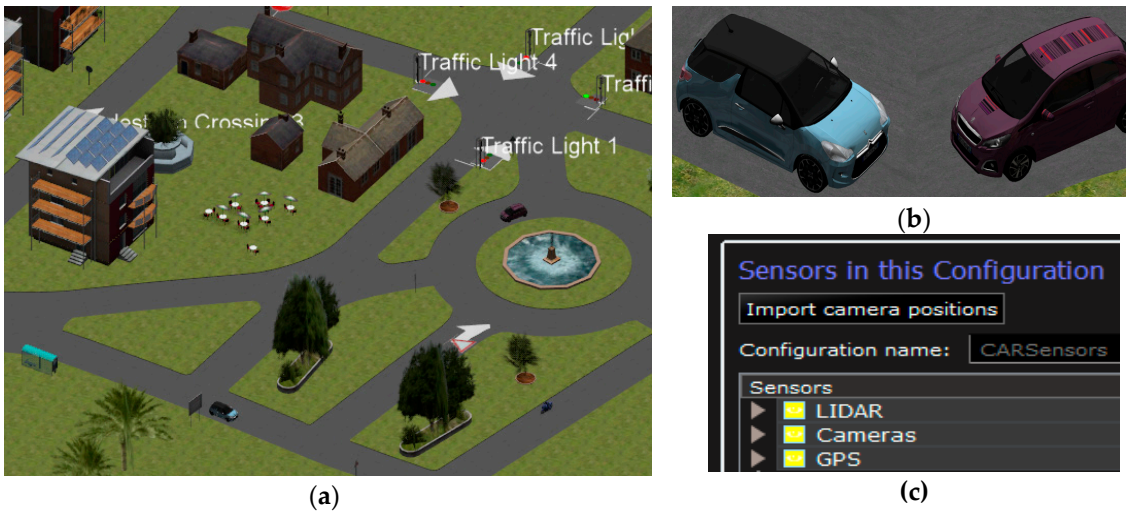
SCANER is a simulation engine for automotive applications in a virtual environment. The functionalities of SCANeR™ can be extended by an interface with third party modules by means of the SDK tool.

#### 2.1.1. 3D traffic scenario

A 3D traffic scenario can be created in order to simulate the behaviour of virtual sensor networks in CPSs for driver-assistance systems. For example, a simple scenario can be composed by elements that represent an urban environment (see Figure 2 (a)).

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109






(a) (b) (c)

**Figure 2.** Traffic scenario in SCANeR. (**a**) Aerial view of simulation scenario, (**b**) vehicle models and (**b**) virtual CPS sensors configuration.

Moreover, each vehicle model represented in this simulation tool can be equipped with specific sensors and actuators models (see Figure 2 (b)). Moreover, a control architecture can be also included in this simulation tool. For example, this architecture can be based on a fuzzy logic controller that manages individual actions on throttle, brake, and steering wheel, from sensory information [32].

2.1.2. Virtual models of CPS components.

In the same way that 3D simulation scenario and vehicles models, different sensors can be represented in order to equip vehicles with these on-board sensors (see Figure 2 (c)). Therefore, connected sensor networks can be created in order to emulate the behaviour of virtual sensors or actuators in the CPS. The co-simulation allows to design and include new features. For example, key steps such as pre-processing (invalid scan points), segmentation (clustered point clouds) and feature extraction (identify features for the individual scans) and classification (object type detection) for the LiDAR sensor can be designed, conditioned and tailored.

*2.2. Matlab/Simulink module*

The second module consists on a library and classification models, implemented in Matlab/Simulink, in order to identify different object types. The framework is composed by a knowledge database and a library with three Artificial Intelligence-based methods by default (i.e., artificial neural network, self-organized map and support vector machine), although this library can be enriched at runtime from data received by all nodes that make up the virtual sensor network. The functional blocks are distributed in different nodes according to their functions. The distributed mobile nodes are in charge of capturing sensory data and run the classification with the required accuracy. Whereas, the main static node incorporates the generated runtime model, the library and knowledge database.

Thee flow diagram of the procedure is shown in Figure 3. The distributed mobile node is also composed by the "Cloud point" block that includes the occupancy grid generation and segmentation of ground plane and nearby obstacles. These nodes also include the classification block for object point detection and feature extraction.

On the other hand, the main static node contains the default library with some classification models. Later on, the library can be enriched from the process simulation. New traffic situations are generated providing new clouds of points (environment information) in each interaction between sensors and the obstacle detection procedure. Based on this continuous information flow and the previous classification (knowledge database), the library executes a parallel learning procedure for all the classification models to obtain a personalize setting for each particular scenario. Finally, once

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



a new best configuration is yielded the corresponding classifier in the distributed mode is then updated.

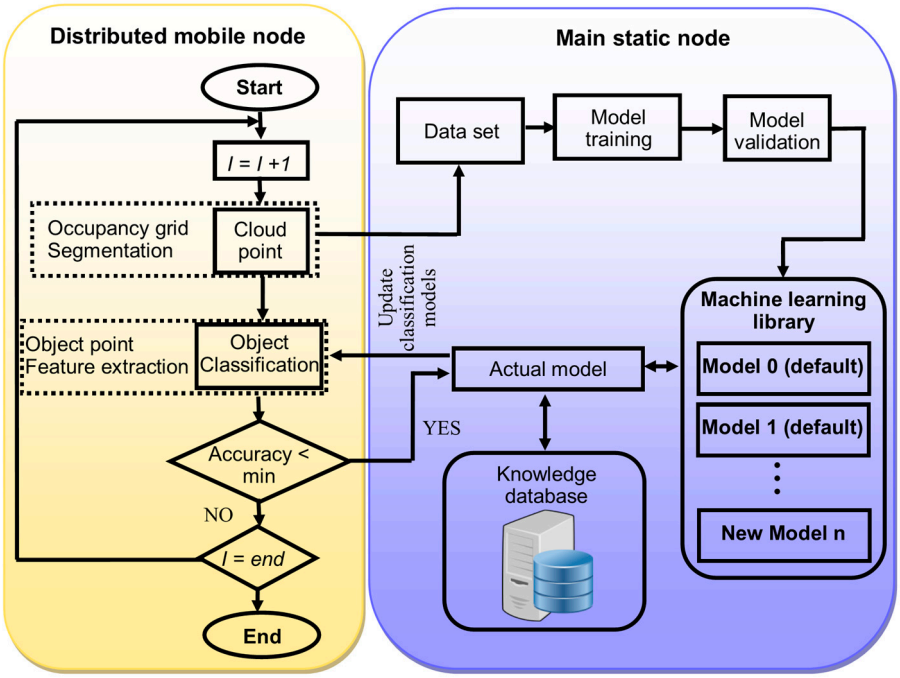



**Figure 3**. Block diagram of classification methodology procedure for object type identification

The decision to select the best classifier model included in the library is carried out by all virtual nodes, although some of them have not yet individually detected a decrease in their particular classification performance metrics. The performance metric chosen was correctly classified instances (CCI). This also generates a knowledge database with the performance of classifier models. By this way, the best models for each weather condition can be recorded.

In this work, the techniques included library are multi-layer perceptron neural network (MLP), self-organized map (SOM) and support vector machine (SVM).

MLP is one of the pioneering and most studied topology of artificial neural networks successfully applied in pattern recognition and modelling. The most popular supervised learning algorithm is the error backpropagation. The operation in a perceptron is described by the formula:

$$y = f\left(\sum_{i=1}^{n} Wp_i x_i + bp\right) \quad (1)$$

where, $f$ is a discontinuous step function; $n$ is number of inputs in a neuron; $X$ the input signals; $Wp$ the synaptic weights; $bp$ the threshold value or *bias* and $y$ the neuron output value.

For training algorithm, the steepest descent is applied:

$$\Delta Wp^n = -\alpha \frac{\partial L}{\partial Wp^n} \quad (2)$$

where $\Delta Wp^n$ is the n-th weight update, α is the learning rate. This process is repeated until some stopping criteria is met. A major problem with gradient descent is that it easily gets stuck in local minima. This can be mitigated by the addition of a momentum term [33], which effectively adds inertia to the motion of the algorithm through weight space, thereby speeding up convergence and helping to escape from local minima [34]:

$$\Delta Wp^n = m\Delta Wp^{n-1} - \alpha \frac{\partial L}{\partial Wp^n} \quad (3)$$

where $m$ is the momentum parameter.

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



The self-organizing map belongs to unsupervised learning methods, i.e., there are no explicit target outputs associated with each input and the goal is to build representations of the input that can be used for decision making [35]. The mapping of the SOM is done by feature vectors associated with each unit, $Wsom_i = (Wsom_i^1, Wsom_i^2, ..., Wsom_i^n)$. A sequential description of how to train a Kohonen SOM is as follows [36]:

1. Initialize all weights randomly;
2. Choose OP randomly in the training set;
3. Select the winning output unit as the one with the largest similarity measure between all weight vectors and the operating point $x$. The winning unit satisfies the following equation:

$$\left|x - Wsom_c\right| = \min\left|x - Wsom_i\right| \tag{4}$$

4. Define the neighbourhood of the winner, by using a neighbourhood function $\Omega_c(i)$ around a winning unit $c$. For instance, the Gaussian function can be used as the neighbourhood function as follows:

$$\Omega_c(i) = \exp\left(-\frac{|p_i - p_c|}{2\sigma^2}\right) \tag{5}$$

where $p_i$ and $p_c$ are the positions of the output units $i$ and $c$ respectively, and $\sigma$ reflects the scope of the neighbourhood. After the definition of the neighbourhood function the weight vector $\omega_i$ of the selected neuron and the weight vectors $\omega$ of its neighbours are updated according to the following formula:

$$\Delta Wsomi = \alpha \cdot \Omega_c(i) \cdot (x - Wsom_i) \tag{6}$$

5. Finally, if the neighbourhood function is bigger the allowed error go to 2; else, stop.

To achieve convergence, the learning rate and the width of the neighbourhood of the winner neuron must shrink to zero with time. A main problem of the SOM algorithm is the fact that the number of training steps of the convergence phase has to be fixed *a priori* and therefore must be set to a large value in order to ensure convergence.

Finally, support vector machines are a group of supervised learning methods very cited in signal and image classification tasks. A kernel function ($K$) $K : \mathbb{R}^m \times \mathbb{R}^m \to \mathbb{R}$ can be described as [37]:

$$K(X_i, X_j) = \phi(X_i)^t \phi(X_j) \tag{7}$$

where, $X$ is a pattern text and $\phi$ the mapping representation.

Given a matrix $X$ the classification function can be represented:

$$K(X_i, X_j) = \sum_{X_i \in S} Ws^t \phi(X_j) + bs \tag{8}$$

where, $S$ is the set of supported vectors, $Ws$ is the weight and $bs$ the support bias.

In particular, the Gaussian kernel is one of the most reported SVM techniques in the literature to nonlinear decision boundary spaces.

$$K(X_i, X_j) = e^{-\frac{\|xi - xj\|^2}{\sigma^2}} \tag{9}$$

Without doubt, these methods are a powerful tool to carry out pattern classification in driving intensions and obstacle detection [38]. Nevertheless, the design, simulation and implementation of effective solutions beyond the academia is still a big challenge for researchers and engineers. Indeed, co-simulation environments play a key role due the high risks of getting actual data from real scenarios in realistic driving and traffic conditions.

## 3. Case study with a sensor network in CPS for driving assistance: LiDAR on-chip and GPS sensors.

A particular driving assistance scenario is defined in order to evaluate and to validate the proposed co-simulation framework. The scenario emulates the real setup available in the Centre of Automation and Robotics (CAR) in Ctra. Campo Real Km. 0.2, Arganda del Rey (Madrid, Spain) that is composed by a test track (a roundabout, traffic lights in the central crossing and additional curves on the main straight) that simulates an urban environment, a fleet of six fully-automated vehicles (distributed mobile nodes) and a main or central static node that is the communications tower [39].

In this use case, a LiDAR sensor is modelled, specifically, the 4- layer type Ibeo Lux. Table 1 shows the specifications of this sensor. The LiDAR specifications are inputs to the sensor model and distances between the detection point with regard to the vehicle are outputs of the LiDAR model. Furthermore, the relative localization (X, Y, Z) of the detect point are outputs to the sensor model.

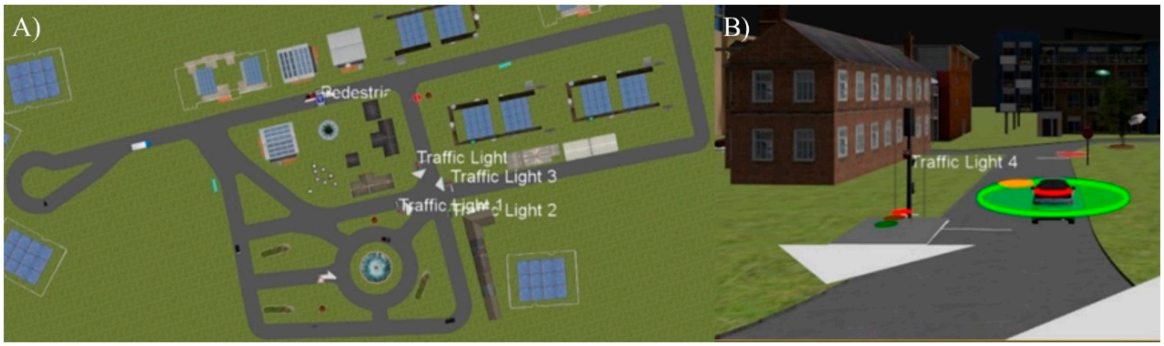



**Figure 4**: (a) Aerial view of simulation scenario of the CPS. (b) A fully-automated vehicle model.

In addition, in order to determine the vehicle localization within the scenario, a DGPS 20 Hz receiver (Trimble BD960) is also modelled. In this model configuration, outputs of this model are the global coordinates of the vehicle location (latitude, longitude and altitude). Both sensor models have been placed on the models of the three vehicles.

**Table 1.** Sensor model configuration. Virtual CPS sensor specifications (inputs to model).

| Specifications / inputs | Ibeo Lux 4 layers |
|---|---|
| Horizontal field | 120 deg. (35 deg. to -50 deg.) |
| Horizontal step | 0.125 deg. |
| Vertical field | 3.2 deg. |
| Vertical step | 0.8 deg. |
| Range | 200 m. |
| Update frequency | 12.5 Hz |

Once the CPS is defined, the next step is to define the object type to be identify in this particular simulation scenario. These virtual sensors mentioned early are incorporated in all vehicles models include in this scenario (see Figure 5 (b)) and can be found different kind of static and dynamic objects as obstacles: trees, traffic signs, traffic lights, vehicles, motorcycles, pedestrians, viewed from different directions and distances. Among common patterns which can be found in a driving assistance scenario, such as road, lane marking, traffic signals, vehicles, pedestrians, etc., only pedestrians were considered for the sake of clarity. The procedure to detect an obstacle for one virtual sensor in this particular use case is depicted in Figure 5.

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



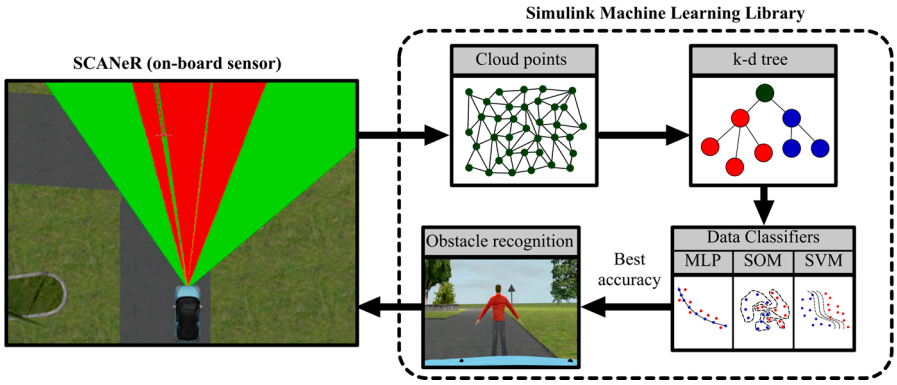



**Figure 5.** Obstacle detection close-loop for each virtual CPS sensor.

*3.1 Experimental set up*

The first step for implementing the initial obstacle recognition library aiming at collision avoidance is to create a training data set from the information collected by virtual sensors using the SCANeR software. The scenario for simulation was set up with three fully automated vehicles (distributed mobile nodes) with the on-board particular sensors. Four hours of sensors data provided by the virtual sensors data were recorded. The analysis consists in a data processing algorithm which begins with fitting the ground plane. It is necessary to search the ground plane and remove ground plane points, using a RANSAC algorithm [40]. It is important to note that the weather considered in these simulations is a sunny day.

The next step during the data processing algorithm is to extract the points that correspond to nearby obstacles corresponding to specific point cloud sequence. Each scan of LiDAR data is stored as a 3-D point cloud. In order to process the sensory data, fast indexing and search capability are required. The procedure is performed by means of "pointCloud objects" from the perception with Computer Vision toolbox, which internally organizes data using a k-d tree structure [41]. After data processing, three sub-sets from experimental population were yielded for training (70% of total samples), validation (15% of total samples) and testing (15%). For manually labelling the most relevant objects from the sensory information, the frames captured by another model sensor have been used. This model sensor is a stereo vision camera which specifications are colour, 0.8 MP, resolution of 1032 x 776 and 20 FPS. The methodology is based on determining in the image captured by the camera the region of interest in which the object to be classified is located. On the other hand, the point cloud provided by the LiDAR are projected onto the image using a coordinate transformation and only the points within the region of interest are selected.

The full data set contains a total of 1500 examples divided in 1050 segments for training the classifier, 220 segments for validations and 230 segments for testing. The training part of the data set contains 525 segments manually labelled positives (class-1) (LiDAR cloud points segments of pedestrian in up-right entire body), and 525 segments without any pedestrian evidence (class-0). Instead, the validation part of data set contains 110 segments positives and 110 negatives, also labelled (class-0 and class-1).

Both segments of the training and validation set contains 18 predictors and one result (class label) for each observation. Therefore, a matrix or table for each set of $n$ rows by 19 columns has been generated, where $n$ is the number of observations corresponding to each set.

**Table 2.** Training and testing set for obstacle recognition library implementation.

| **Full data set** | | **Training set** | | **Validation set** | | **Test set** | |
|---|---|---|---|---|---|---|---|
| 1500 segm. | | 1050 segm. | | 220 segm. | | 230 segm. | |
| **Pos.** | **Neg.** | **Pos.** | **Neg.** | **Pos.** | **Neg.** | **Pos.** | **Neg.** |
| 750 | 750 | 525 | 525 | 110 | 110 | 115 | 115 |

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



*3.2 Training and initial test of obstacle recognition library*

As it was already mentioned, the library contains initially some classification models by default and later, the content will be enriched in runtime during the process simulation of the scenario. In this particular use case, three techniques are then considered, i.e., multi-layer perceptron neural network, a self-organization map and a support vector machine. The main rationale for their selection is the solid mathematical foundations, demonstrated ability in modelling in complex scenarios and a wide range of successful applications.

The first technique was a multi-layer perceptron neural network with an input layer with 18 neurons, a hidden layer of 40 neurons and an output layer with a single neuron and linear activation function. For training, the method used was gradient descent with momentum and adaptive learning rate backpropagation. The initial values of learning rate, and performance goal were $10^{-7}$ and $10^{-8}$, respectively. The network was trained during 50,000 iterations, after which it reached a best performance of 0.0216 and a gradient of 0.0021. Using the validation set, values of mean square error (MSE) of 0.0409 and correctly classified instances (CCI) equal to 95.91% was reached.

The second method for the obstacle recognition library is a self-organizing maps. Specifically, a topology function that creates neurons in an *N*-dimensional random pattern was used, and the dimensions were 22 x 2. Finally, the *Manhattan* function was applied as distance function. In addition, an input weight equal to the number of observations in the training set was set, i.e. $w = 1050$. The network was trained during a cover step of 10000 and an initial neighbour size of 4, after which it reached a MSE of 0.132 and a CCI equal to 89.55% was reached using the validation set.

Finally, a support vector machine was also implemented in the library. This nonlinear classifier uses a Gaussian Kernel function with a kernel scale $\sigma = 0.94$ and a box constrain of $9.78e^4$. The supervised learning method was trained during 1255 iterations, until its reason for convergence gradient reached a $\Delta < 0.001$. The results obtained during validation were a MSE of 0.0636 and a CCI equal to 93.64%.

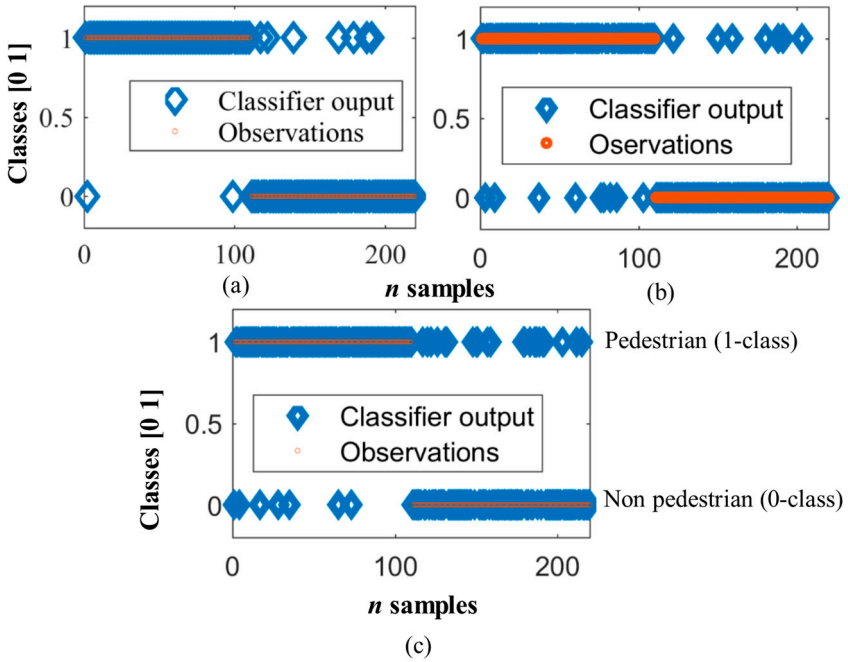



**Figure 6:** Validation results in pedestrian detection (a) MLP, (b) SVM and (c) SOM

Figure 6 shows the classifiers outputs of the three models vs. observed classes using the validation set. The classifiers outputs indicate whether the detected object is a pedestrian (class 1) or not (class 0). The three classifiers showed very good performance indices although the smallest error and the highest number of correctly classified instances corresponds to MLP, following by SVM and finally the worse result corresponds to SOM. This study is not conclusive and therefore a validation

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



with unknown data set is required using more performance indices in order to make a more complete comparative study among the three classifiers.

*3.3 Final validation of obstacle recognition library*

The current testing set in sunny weather conditions contains 230 segments (not known beforehand), and detailed annotations regarding the pedestrian appearances (in terms of occlusion), namely: occluded/partial pedestrians (class-0) and entire body pedestrians (class-1). A summary of the testing data set is shown in Table 2.

A total of six performance indices were considered in the validation study on the basis of experimental run as follows: number of correctly classified instances (CCI), the number of incorrectly classified instances (ICI), the mean absolute error (MAE), the root mean squared error (RMSE), the relative absolute error (RAE) and the root relative squared error (RRSE). The results of the comparative study of the classifiers (MLP, SVM and SOM) are summarized in Table 3.

**Table 3.** Comparative study of MLP, SVM and SOM

| Performance Index\Approach | MLP | SVM | SOM |
|---|---|---|---|
| CCI(%) | 88.19 | 91.36 | 90.91 |
| ICI (%) | 11.81 | 8.64 | 9.09 |
| MAE | 0.12 | 0.09 | 0.09 |
| RMSE | 0.34 | 0.29 | 0.30 |
| RAE (%) | 23.64 | 17.29 | 18.68 |
| RRSE (%) | 9.274 | 7.93 | 8.36 |

The application of the MLP yielded 23.64% of RAE. On the contrary, SVM and SOM achieved an excellent accuracy, for instances 17.29% and 18.68% in RAE, respectively, although not much lower in percentage than the MLP error. The excellent behaviour is also endorsed with high correct classified instances of 91.36% and 90.91%. However, SOM and in particular SVM do not outperforms significantly MLP with regard to all figures of merits considered in this study.

It should be noted that this study has been carried out in good weather climatological conditions.

## 4. Experimental results with other weather conditions

Additional experimental test for evaluating the co-simulation framework and the performance of the library for obstacle detection before different weather conditions was also conducted. Sunny, fog, rainy and snowy were taken into account.

The same simulation time (i.e., 2 h) for each weather condition are considered in each running of virtual sensors in the CPS. All virtual objects and its corresponding position are previously known. Some of these dynamic objects in scenario are 205 pedestrians, 10 bicycles, 60 motorbikes, 213 small and medium vehicles and 20 trucks. The goal is to assess the accuracy for detecting and identifying pedestrians in spite of the other obstacles that can be recognized but not classified in this case study.

Another test set was created for each weather condition (sunny, fog, rainy, and snowy). Each data set consists in 1010 samples with 205 samples positively labelled pedestrian detections and 805 negatively labelled. The three classifiers were evaluated with these four data sets.

In order to assess the performance of classifiers, some performance indices are used such as Correct Classify Samples or Correct Rate (CCR), Incorrectly classified samples or Error Rate (ECR), Correctly Classified Positive Samples / True Positive Samples or Sensitivity (*Sn*), Correctly Classified Negative Samples / True Negative Samples or Specificity (*Sp*), Correctly Classified Positive Samples / Positive Classified Samples or Positive Predictive Value (PPV), Correctly Classified Negative Samples / Negative Classified Samples or Negative Predictive Value (NPV), $Sn / (1 - Sp)$ or Positive Likelihood (PL) and $(1 - Sn) / Sp$ or Negative Likelihood (NL). The resulting performance indices (PI) for the four weather conditions (WC) are shown in Table 5.

Preprints (www.preprints.org) | NOT PEER-REVIEWED | Posted: 4 August 2017 doi:10.20944/preprints201708.0014.v1

Peer-reviewed version available at Sensors 2017, 17, 2109; doi:10.3390/s17092109



**Table 5.** Comparative study of Artificial Intelligence-based methods before different weather conditions

| PI/WC | *Sunny* | | | *Fog* | | | *Rainy* | | | *Snowy* | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | MLP | SVM | SOM | MLP | SVM | SOM | MLP | SVM | SOM | MLP | SVM | SOM |
| **CCR (%)** | 88.70 | 87.92 | 80.79 | 85.40 | 84.75 | 80.79 | 79.21 | 80.40 | 80.00 | 62.22 | 77.03 | 79.90 |
| **ECR (%)** | 11.30 | 12.08 | 19.21 | 14.60 | 15.25 | 19.21 | 30.79 | 19.60 | 20.00 | 37.78 | 22.97 | 20.10 |
| ***Sn*** | 0.886 | 0.886 | 0.768 | 0.852 | 0.868 | 0.767 | 0.805 | 0.859 | 0.764 | 0.595 | 0.906 | 0.760 |
| ***Sp*** | 0.888 | 0.854 | 0.966 | 0.849 | 0.766 | 0.971 | 0.668 | 0.585 | 0.942 | 0.532 | 0.239 | 0.951 |
| **PPV** | 96.90 | 95.96 | 98.92 | 95.74 | 93.57 | 99.01 | 90.53 | 89.06 | 98.14 | 83.33 | 82.37 | 98.44 |
| **NPV** | 66.42 | 65.54 | 51.41 | 59.44 | 59.70 | 51.42 | 46.62 | 51.50 | 50.43 | 25.11 | 39.20 | 50.32 |
| **PL** | 7.894 | 6.05 | 22.48 | 5.635 | 3.709 | 26.19 | 2.427 | 2.073 | 13,05 | 1.271 | 1.19 | 15.56 |
| **NL** | 0.129 | 0.134 | 0.241 | 0.174 | 0.172 | 0.241 | 0.797 | 0.240 | 0.251 | 0.762 | 0.395 | 0.252 |

In the case of CCR, there is clear tendency to decrease the number of correctly classified instances due to the interference of weather conditions in the sensors field of view. In sunny and fog conditions, MLP and SVM showed better results than SOM. However, MLP showed a more evident deterioration with regard to the weather conditions change, unlike SVM and specially SOM that remain more stable in spite of the weather condition. In fact, SOM outperforms other topologies in the most extreme weather condition that is snowy with the highest specificity value (*Sp*).

On the other hand, SVM produces the best classification in rainy conditions although in sunny and fog, the results are worse than those given by MLP. Figure 7 depicts the behaviour of CCR, PPV and *Sp* of the three classifiers with regard to the four weather conditions.

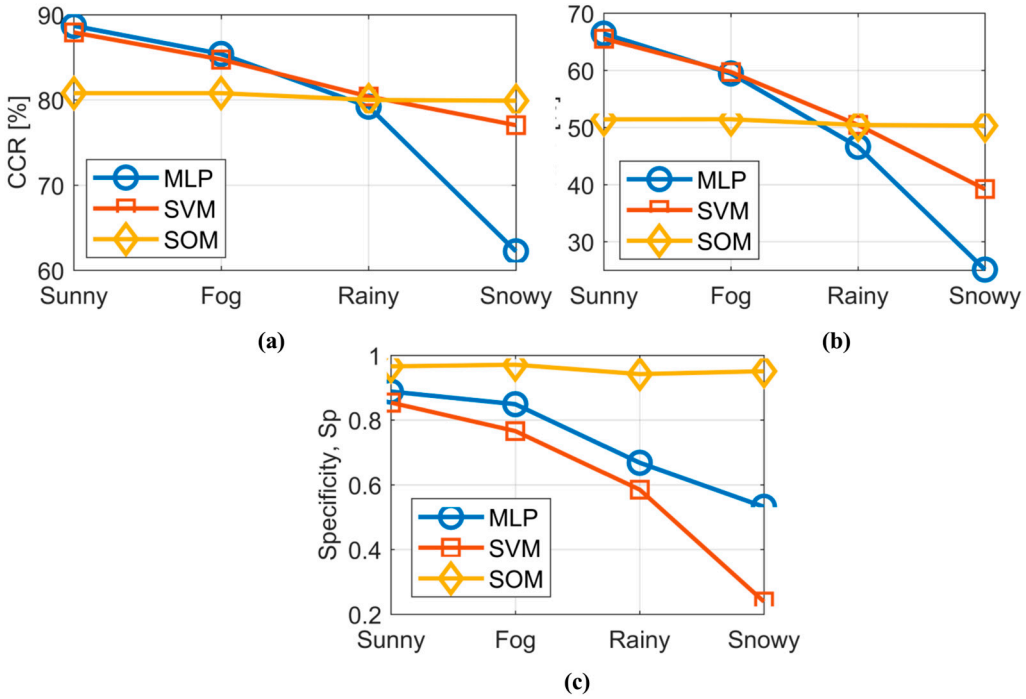



**Figure 7:** Behaviour of the performance indices for each classifier with regard to weather conditions (a) CCR, (b) NPV and (c) *Sp*

It is evident that the best classifier differs according to the weather conditions. The classifier based on MLP behaves better than SVM and SOM for sunny and fog conditions, whereas for rainy conditions, SVM-based model is the most appropriate. However, for the most extreme weather condition (snowy) the SOM-based classifier is the most suitable. Overall, the SOM-based classifier depicts the most regular behaviour before all weather conditions.

## 5. Conclusions

This work presents a co-simulation framework for obstacle recognition on the basis of sensory data provided by a virtual sensor network in a cyber-physical system. This co-simulation framework is designed and built using SCANeR studio and Matlab/Simulink. An assistance driving scenario is created in SCANeR in order to represent and emulate the behaviour of a cyber-physical system. On other hand, a library of Artificial Intelligence-based methods for obstacle detection is designed and implemented in Matlab/Simulink. The library is composed by three methods i.e., multi-layer perceptron neural network, a self-organizing map and a support vector machine.

The whole system is evaluated in a particular use case built from two types of sensory data (LiDAR on-chip and GPS sensors) within a defined scenario. The comparative study demonstrates that the proposed obstacles detection methods are powerful strategies for pedestrian detection. In the training and validation phase of the classification models, the best results were achieved with the multi-layer perceptron and the support vector machine, but not so remarkable to discard self-organizing map.

In addition, a second evaluation is also performed which consists in capturing sensory data provided by sensors but with different weather conditions. In this second evaluation, all methods are able to adequately classify pedestrians. Multi-layer perceptron provides very good results in sunny and fog conditions but at the same time they have a tendency to deteriorate its performance before other weather conditions. The support vector machine also produces the best result in rainy conditions. On the other hand, the self-organizing map produces the worst figures of merits, showing a more regular performance from data provided by all virtual on-chip LiDAR sensors.

The results of this investigation corroborate the high influence of the weather conditions on the classifiers accuracy for detecting and classifying pedestrians. Further research is focused on an optimal tuning of the library' methods and the development of a self-organization procedure to select the most appropriate method among those available in the library in each particular scenario. Finally, the proposed co-simulation system will be embedded and validated in real driving environments as part of the contributions to the IoSENSE project[1].

**Acknowledgments:** Authors wish to thank the support given by the project [1]IoSENSE: Flexible FE/BE Sensor Pilot Line for the Internet of Everything, funded by the Electronic Component Systems for European Leadership Joint (ECSEL) Undertaking under grant agreement No 692480. http://www.iosense.eu/

**Author Contributions:** Rodolfo Haber contributed with a technical and scientific review of the article whole. Antonio Artuñedo was in charge of review of technical words related to automotive applications and he collaborated with the design of co-simulation framework. Fernando Castano and Gerardo Beruvides are designed and implemented the scenario, co-simulation framework architecture and machine learning library, and wrote the paper.

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

## References

1. Trappey, A. J. C.; Trappey, C. V.; Govindarajan, U. H.; Sun, J. J.; Chuang, A. C., A Review of Technology Standards and Patent Portfolios for Enabling Cyber-Physical Systems in Advanced Manufacturing. *IEEE Access* **2016,** 4, 7356-7382.

2. Bures, T.; Weyns, D.; Klein, M.; Haber, R. E. In *1st International Workshop on Software Engineering for Smart Cyber-Physical Systems (SEsCPS 2015)*, 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 16-24 May 2015, 2015; 2015; pp 1009-1010.

3. Lee, J.; Bagheri, B.; Kao, H.-A., A cyber-physical systems architecture for industry 4.0-based manufacturing systems. *Manuf. Let.* **2015,** 3, 18-23.

4. Sanislav, T.; Miclea, L., Cyber-physical systems-concept, challenges and research areas. *Journal of Control Engineering and Applied Informatics* **2012,** 14, (2), 28-33.

5. Bhave, A.; Krogh, B. H.; Garlan, D.; Schmerl, B. In *View consistency in architectures for cyber-physical systems*, Proceedings of the 2011 IEEE/ACM Second International Conference on Cyber-Physical Systems, 2011; IEEE Computer Society: 2011; pp 151-160.

6. Gölzer, P.; Cato, P.; Amberg, M. In *Data Processing Requirements of Industry 4.0-Use Cases for Big Data Applications*, ECIS, 2015; 2015.
7. Yuan, X.; Anumba, C. J.; Parfitt, K. M., Review of the potential for a cyber-physical system approach to temporary structures monitoring. *International Journal of Architectural Research: ArchNet-IJAR* **2015,** 9, (3), 26-44.
8. Watteyne, T.; Handziski, V.; Vilajosana, X.; Duquennoy, S.; Hahm, O.; Baccelli, E.; Wolisz, A., Industrial wireless ip-based cyber–physical systems. *Proceedings of the IEEE* **2016,** 104, (5), 1025-1038.
9. Wan, J.; Zhang, D.; Zhao, S.; Yang, L.; Lloret, J., Context-aware vehicular cyber-physical systems with cloud support: architecture, challenges, and solutions. *IEEE Communications Magazine* **2014,** 52, (8), 106-113.
10. Ruan, J.; Yu, W.; Yang, Y.; Hu, J. In *Design and realize of tire production process monitoring system based on cyber-physical systems*, Computer Science and Mechanical Automation (CSMA), 2015 International Conference on, 2015; IEEE: 2015; pp 175-179.
11. Iovine, A.; Valentini, F.; De Santis, E.; Di Benedetto, M. D.; Pratesi, M., Safe human-inspired mesoscopic hybrid automaton for autonomous vehicles. *Nonlinear Analysis: Hybrid Systems* **2017,** 25, 192-210.
12. Zhao, X.; Mu, K.; Hui, F.; Prehofer, C., A cooperative vehicle-infrastructure based urban driving environment perception method using a D-S theory-based credibility map. *Optik - International Journal for Light and Electron Optics* **2017,** 138, 407-415.
13. Reiser, D.; Paraforos, D.; Khan, M.; Griepentrog, H.; Vázquez-Arellano, M., Autonomous field navigation, data acquisition and node location in wireless sensor networks. *Precision Agriculture* **2016**, 1-14.
14. Rapberger, W., The emergence of autonomous vehicles. In Accenture: 2017.
15. Martínez, C.; Richardson, T.; Thomas, P.; Du Bois, J. L.; Campoy, P., A vision-based strategy for autonomous aerial refueling tasks. *Robotics and Autonomous Systems* **2013,** 61, (8), 876-895.
16. Morrison, A.; Renaudin, V.; Bancroft, J. B.; Lachapelle, G., Design and testing of a multi-sensor pedestrian location and navigation platform. *Sensors* **2012,** 12, (3), 3720-3738.
17. Yang, S.; Scherer, S. A.; Schauwecker, K.; Zell, A., Autonomous landing of MAVs on an arbitrarily textured landing site using onboard monocular vision. *Journal of Intelligent & Robotic Systems* **2014,** 74, (1-2), 27.
18. Lee, H.; Kim, H. J., Trajectory tracking control of multirotors from modelling to experiments: A survey. *International Journal of Control, Automation and Systems* **2017**, 1-12.
19. Sim, S.; Sock, J.; Kwak, K., Indirect Correspondence-Based Robust Extrinsic Calibration of LiDAR and Camera. *Sensors* **2016,** 16, (6), 933.
20. Pandey, G.; McBride, J. R.; Savarese, S.; Eustice, R. M., Automatic extrinsic calibration of vision and lidar by maximizing mutual information. *Journal of Field Robotics* **2015,** 32, (5), 696-722.
21. Chen, C.-I.; Koseluk, R.; Buchanan, C.; Duerner, A.; Jeppesen, B.; Laux, H., Autonomous Aerial Refueling Ground Test Demonstration—A Sensor-in-the-Loop, Non-Tracking Method. *Sensors* **2015,** 15, (5), 10948-10972.
22. Kala, R., 3 - Perception in Autonomous Vehicles. In *On-Road Intelligent Vehicles*, Butterworth-Heinemann: 2016; pp 36-58.
23. Taghavifar, H.; Mardani, A.; Karim Maslak, H., A comparative study between artificial neural networks and support vector regression for modeling of the dissipated energy through tire-obstacle collision dynamics. *Energy* **2015,** 89, 358-364.
24. Apatean, A.; Rogozan, A.; Bensrhair, A., Visible-infrared fusion schemes for road obstacle classification. *Transportation Research Part C: Emerging Technologies* **2013,** 35, 180-192.
25. Navarro, P. J.; Fernández, C.; Borraz, R.; Alonso, D., A machine learning approach to pedestrian detection for autonomous vehicles using high-definition 3D range data. *Sensors* **2016,** 17, (1), 18.
26. Zhang, Z.; Eyisi, E.; Koutsoukos, X.; Porter, J.; Karsai, G.; Sztipanovits, J., A co-simulation framework for design of time-triggered automotive cyber physical systems. *Simulation Modelling Practice and Theory* **2014,** 43, 16-33.
27. Artuñedo, A.; Godoy, J.; Haber, R.; Villagrá, J.; Toro, R. M. d. In *Advanced Co-simulation Framework for Cooperative Maneuvers Among Vehicles*, 2015 IEEE 18th International Conference on Intelligent Transportation Systems, 15-18 Sept. 2015, 2015; 2015; pp 1436-1441.
28. Ferracuti, F.; Freddi, A.; Monteriù, A.; Prist, M., An integrated simulation module for cyber-physical automation systems. *Sensors* **2016,** 16, (5), 645.

29. de Souza, I. D.; Silva, S. N.; Teles, R. M.; Fernandes, M. A., Platform for Real-Time Simulation of Dynamic Systems and Hardware-in-the-Loop for Control Algorithms. *Sensors* **2014,** 14, (10), 19176-19199.

30. Mozumdar, M.; Song, Z. Y.; Lavagno, L.; Sangiovanni-Vincentelli, A. L., A model-based approach for bridging virtual and physical sensor nodes in a hybrid simulation framework. *Sensors* **2014,** 14, (6), 11070-11096.

31. Lee, B.-H.; Song, J.-H.; Im, J.-H.; Im, S.-H.; Heo, M.-B.; Jee, G.-I., GPS/DR error estimation for autonomous vehicle localization. *Sensors* **2015,** 15, (8), 20779-20798.

32. Godoy, J.; Pérez, J.; Onieva, E.; Villagrá, J.; Milanés, V.; Haber, R., A driverless vehicle demonstration on motorways and in urban environments. *Transport* **2015,** 30, (3), 253-263.

33. Plaut, D. C., Experiments on Learning by Back Propagation. **1986**.

34. Graves, A., Neural Networks. In *Supervised Sequence Labelling with Recurrent Neural Networks*, Springer Berlin Heidelberg: Berlin, Heidelberg, 2012; pp 15-35.

35. Kohonen, T.; Schroeder, M.; Huang, T., *Self-Organizing Maps*. Springer-Verlag New York: Inc., Secaucus, NJ, 2001; Vol. 43, p 2.

36. Voumvoulakis, E. M.; Hatziargyriou, N. D., Decision trees-aided self-organized maps for corrective dynamic security. *IEEE Transactions on Power Systems* **2008,** 23, (2), 622-630.

37. Murty, M. N.; Raghava, R., Kernel-Based SVM. In *Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks*, Springer International Publishing: Cham, 2016; pp 57-67.

38. Kim, I.-H.; Bong, J.-H.; Park, J.; Park, S., Prediction of Driver's Intention of Lane Change by Augmenting Sensor Information Using Machine Learning Techniques. *Sensors* **2017,** 17, (6), 1350.

39. Artuñedo, A.; del Toro, R.; Haber, R., Consensus-Based Cooperative Control Based on Pollution Sensing and Traffic Information for Urban Traffic Networks. *Sensors* **2017,** 17, (5), 953.

40. Derpanis, K. G., Overview of the RANSAC Algorithm. *Image Rochester NY* **2010,** 4, (1), 2-3.

41. Hunt, W.; Mark, W. R.; Stoll, G. In *Fast kd-tree construction with an adaptive error-bounded heuristic*, Interactive Ray Tracing 2006, IEEE Symposium on, 2006; IEEE: 2006; pp 81-88.