D4.3

Situational Awareness Solution based on the Machine Learning Applications

Topic	SU-ICT-01-2018
Project Title	Artificial Intelligence-based Cybersecurity for Connected and Automated Vehicles
Project Number	833611
Project Acronym	CARAMEL
Contractual Delivery Date	M16
Actual Delivery Date	M18
Contributing WP	WP4
Project Start Date	01/10/2019
Project Duration	30 Months
Dissemination Level	Public
Editor	UCY
Contributors	8BELLS, ATOS, 0INF, AVL, PANA, UPAT

Version	Date	Modifications
0.1	09/04/2020	Initial ToC
0.2	15/06/2020	First document draft
0.3	17/08/2020	Second document draft
0.4	20/10/2020	Third document draft
0.5	18/12/2020	Fourth document draft
0.6	20/02/2021	Fifth document draft
0.7	18/03/2021	Final draft for internal review and SAB
0.8	25/03/2021	Final changes and revisions
1.0	29/03/2021	Final version

DISCLAIMER OF WARRANTIES

This document has been prepared by CARAMEL project partners as an account of work carried out within the framework of the contract no 833611.

Neither Project Coordinator, nor any signatory party of CARAMEL Project Consortium Agreement, nor any person acting on behalf of any of them:

- makes any warranty or representation whatsoever, express or implied,
    - with respect to the use of any information, apparatus, method, process, or similar item disclosed in this document, including merchantability and fitness for a particular purpose, or
    - that such use does not infringe on or interfere with privately owned rights, including any party's intellectual property, or
- that this document is suitable to any particular user's circumstance; or
- assumes responsibility for any damages or other liability whatsoever (including any consequential damages, even if Project Coordinator or any representative of a signatory party of the CARAMEL Project Consortium Agreement, has been advised of the possibility of such damages) resulting from your selection or use of this document or any information, apparatus, method, process, or similar item disclosed in this document.

CARAMEL has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 833611. The content of this deliverable does not reflect the official opinion of the European Union. Responsibility for the information and views expressed in the deliverable lies entirely with the author(s).

DISCLOSURE STATEMENT

"The following document has been reviewed by the CARAMEL External Security Advisory Board as well as the Ethics and Data Management Committee of the project. Hereby, it is confirmed that it does not contain any sensitive security, ethical or data privacy issues."

Table of Contents

List of Figures

List of Tables

List of Acronyms

ACC	Adaptive Cruise Control
AEB	Automatic Emergency Braking
AP	Average Precision
AUC	Area Under Curve
BPSK	Binary Phase Shift Keying
CAN	Controller Area Network
CAV	Connected and Autonomous Vehicle
CCD	Charge-Coupled Device
CDF	Cumulative Distribution Function
CMOS	Complementary Metal-Oxide Semiconductor
CNN	Convolutional Neural Networks
CV	Computer Vision
DR	Detection Rate
ECU	Engine Control Unit
EKF	Extended Kalman Filter
FDR	False Detection Rate
GAN	Generative Adversarial Network
GNSS	Global Navigation Satellite System
IoU	Intersection over Union
IR	Infrared
KM	Kinematic Model
LED	Light Emitting Diodes
LiDAR	Light Detection and Ranging
LKA	Lane Keep Assist
LMSE	Localization Mean Square Error
LSQ	Least-Squares
mAP	mean Average Precision
MBOC	Multiplexed Binary Offset Carrier
MIR	Mid-Infrared
ML	Machine Learning
MLE	Maximum Likelihood Estimation
MR	Misdetection Rate
MSE	Mean Square Error
NIR	Near-Infrared

OBU	On-Board Unit
OS-NMA	Open Service Navigation Message Authentication
PA	Parking Assist
PVT	Position, Velocity and Timing
QPSK	Quad Phase Shift Keying
RGB	Red, Green, Blue
RGCL	Robust Graph based Cooperative Localization
ROC	Receiver Operating Characteristics
RPS	Relative Positioning System
RTCL-MLE	Robust Traditional Cooperative Localization based on MLE
SDR	Software-Defined Radio
SoO	Signals of Opportunity
SSD	Single-Shot multibox Detection
SSIM	Structural similarity index measure
STD	Standard Deviation
TJA	Traffic Jam Assist
ToF	Time of Flight
V2V	Vehicle-to-Vehicle
V2I	Vehicle-to-Infrastructure
VANET	Vehicular Ad-hoc NETwork
VIS	Visible

Executive Summary

This document describes the work performed in task T4.3 towards understanding the autonomous vehicle's state at any time; thus, providing holistic situational awareness in the case of cyber-attacks on the vehicle's camera or GPS sensors. This is achieved through enhanced sensor fusion both at the sensor data level, as well as at the solutions' level, i.e., through the combination of standalone solutions to deliver higher robustness to measurement noise and increase the attack detection confidence.

First, the details of sensor fusion solutions are presented for detecting camera attacks, which are different from the solutions described in previous deliverables and may in fact complement those solutions developed in the context of T4.2/D4.2. Second, the details of diverse and highly complementary sensor fusion solutions are provided for detecting GPS location spoofing attacks. Third, detailed methodologies are presented for combining the respective camera-related and GPS-related attack detection solutions into powerful hybrid schemes to build advanced defensive mechanisms against camera attacks and GPS location spoofing attacks, respectively.

1 Introduction

Understanding the autonomous vehicle's state at any time is critical to identifying potential threats, reliably detecting on-going attacks, and applying proper mitigation actions to recover. The extensive literature on sensor fusion provides a wide variety of powerful tools and techniques for combining heterogeneous multi-source sensory data, which are readily available inside the sensor-rich autonomous vehicles or can be easily collected by leveraging Vehicle-to-Infrastructure / Vehicle-to-Vehicle (V2I/V2V) communication capabilities to enable cyber-attack detection. These data can be combined in an optimal way, where optimality may refer to increased robustness against measurement noise, higher detection accuracy, lower response time or any other performance criterion.

In task T4.3, different fusion-based solutions are developed to defend against cyber-attacks that target either the vision sub-system (and in particular the camera) or the location awareness sub-system (i.e., the GPS location) of the autonomous vehicle. In particular, one of the developed solutions, mitigates camera cyber-attacks by using the Light Detection and Ranging (LiDAR) sensor as an independent module that processes only raw 3D point clouds to assist the vehicle's perception engine, that typically uses data only from the camera, in delivering correct scene understanding. Another solution, fuses camera and GPS, which are commonly used in perception engines in autonomous vehicles. The fusion of such data can provide insights about cyber-attack incidents, as well as enable the robustification of an existing attack detection pipeline.

Regarding the detection of GPS spoofing attacks, one of the developed solution leverages in-vehicle multisensory data (e.g., accelerometer, gyroscope, compass, etc.) to compute a parallel GPS-free stream of estimated vehicle's locations using a fall-back localisation method based on Bayesian filtering. This enables the detection of potential location spoofing attacks by comparing the estimated vehicle position with the GPS location reading. The other solution is essentially a collaborative GPS spoofing defence mechanism that relies on multi-modal sensor fusion among the vehicles of a Vehicular Ad-hoc NETwork (VANET), while the measurements include the relative distances, the relative angles, and the relative azimuth angles among the vehicles, as well as the absolute position measurements (i.e., GPS positions) of all vehicles.

Going beyond the information fusion at the sensor level, it is possible to provide improved situational awareness for achieving higher attack detection accuracy and reliability by correlating the outputs of complementary detection modules. To this end, a framework is developed that combines the *DriveGuard* Convolutional Autoencoder developed in D3.2 with the traffic sign attack detection pipeline developed in D4.2 for detecting both internal and external attacks. Furthermore, a second framework is developed to achieve enhanced GPS location spoofing attack detection by combining the aforementioned spoofing detection solutions, i.e., the in-vehicle and the collaborative solutions.

1.1 Purpose of this Document

This document presents the details of fusion-based solutions for detecting attacks on camera and GPS sensors. Fusion may take place at the sensor measurements level, i.e., by combining heterogenous sensor data. To this end, two sensor fusion solutions for detecting camera attacks are described, namely one that leverages data from the LiDAR sensor to validate the observations of the camera sensor and another one that utilizes GPS data to assist the detection of camera attacks. These solutions are validated experimentally using realistic data obtained through the CARLA simulator. Notably, both solutions may complement the solutions developed in the context of T4.2/D4.2 to defend against this type of attacks. In addition, the methodology to combine the camera attack detection solutions developed in T4.3 is also presented and validated.

Along the line of fusing diverse multi-source sensory data, this document also describes two sensor fusion solutions for detecting GPS location spoofing attacks including one solution for fusing the data that are readily available from the vehicle's On-Board Unit (OBU), in an in-vehicle attack detection approach, and another solution that fuses information from neighbouring vehicles, in a centralised collaborative attack detection scheme. These solutions are validated experimentally by using realistic data obtained through the CARLA simulator, while the methodology to combine them in a hybrid approach is also presented.

1.2 Structure of this Document

The remainder of this document is structured as follows:

- **Section 2 – Sensor Fusion Solutions for Detecting Camera Attacks**: describes two solutions for detecting attacks on the camera sensors followed by respective mitigation mechanisms, namely i) one solution based on LiDAR as an auxiliary data source and ii) another solution based on imagery and GPS data that leverages Machine Learning (ML) techniques.
- **Section 3 – Sensor Fusion Solutions for Detecting GPS Location Spoofing Attacks**: elaborates on two standalone solutions for detecting GPS location spoofing attacks, namely i) by fusing in-vehicle sensor readings and ii) by means of a collaborative GPS spoofing defence mechanism that relies on multi-modal sensor fusion among the vehicles of a VANET.
- **Section 4 – Combination of Standalone Fusion Solutions**: discusses how the standalone attack detection solutions presented in Section 2 and Section 3 can be coupled to provide enhanced situational awareness against camera attacks and location spoofing attacks, respectively.
- **Section 5 – Conclusion**: provides concluding remarks and some directions for further research.

1.3 Relation to other Tasks and Deliverables

Deliverable D4.3 is related to the following tasks and deliverables:

- **Task 2.1 – Use Cases Elaboration / D2.1 – Report on Detailed Specification of Use Cases**: D4.3 receives the definition of the CARAMEL use cases, as well as the technical evaluation strategy deployed within the project.
- **Task 2.3 – Analysis of Security and Privacy Requirements / D2.3 – Specifications of CARAMEL Security and Privacy Requirements**: D4.3 considers the user, security and privacy requirements defined in D2.2 and D2.3, with focus on the definition of the various scenarios for each CARAMEL use case.
- **Task 2.4 – System Specifications and Architecture / D2.4 – System Specifications and Architecture**: D4.3 receives the overall system architecture from D2.4, including the definition of attack types related to the camera and GPS sensors, as well as the associated interfaces.
- **Task 3.1 – Automotive Threat Modelling / D3.1 – Automotive Threat Modelling:** D4.3 receives the threat analysis and the definition of assets, access points, and systems present in the automotive sector from D3.1, focusing on the threats against the camera and GPS sensor.
- **Task 3.2 – Cyberthreat Detection Using Sparse and Deep Priors / D3.2/D3.5 Cyberthreat Detection Using Sparse and Deep Priors:** D4.3 will receive the algorithms for camera image preprocessing and attack mitigation.
- **Task 3.3 – Cyberthreat Detection and Response Techniques for Cooperative Automated Vehicles / D3.6 – Cyberthreat Detection and Response Techniques for Cooperative Automated Vehicles:** D3.6 presents the workflow of the GPS spoofing attack detection solutions detailed in D4.3 and describes how an alert will be triggered, either at the vehicle's OBU for the in-vehicle solution (Section 3.2) or at the network side for the collaborative multi-vehicle solution (Section 3.3), after an attack is detected.
- **Task 4.2 – AI-based Context-rich and Context-aware Cybersecurity / D4.2 – Robust to Cyberattack Machine Vision Models Based on Improved Training Methods and Anomaly Detection Deep Networks**: D4.3 will integrate the outcome of anomaly detection and mitigation results with respect to attacks on camera sensors from D4.2.
- **Task 5.1 – Report on the Collection and Storage of Data from Smart Vehicle's Internal Network / D5.2 – Report on the Collection and Storage of Data from Smart Vehicle's Internal Network**: D5.2 will describe in detail the data types and processing steps for the measurements required as inputs to the both the standalone and combined attack detection solutions described in D4.3.
- **Task 5.2 – Advanced Algorithmic Detection of Attacks via passive Anti-hacking Device / D5.3 – Machine Learning based Detection of Attacks into Anti-hacking Device**: D5.3 will present how the attack detection solutions described in D4.3 will be implemented and integrated into the anti-hacking device.

2 Sensor Fusion Solutions for Detecting Camera Attacks

2.1 Detecting Camera Attacks using LiDAR

This section describes how to mitigate attacks in autonomous driving systems using LiDAR as an auxiliary data source. Considering that the perception module, that uses data only from the camera, has been attacked, an independent module using only raw 3D point clouds tries to provide the correct scene understanding.

2.1.1 Sensors and Measurements

The camera and LiDAR sensors are considered in this solution. The camera will detect objects or structural elements with some confidence and the LiDAR will provide auxiliary measurements for the same objects. Although, before explaining the detection part, a short introduction about the camera and LiDAR sensor is provided. The raw data used for training in these tasks can be accessed from Kitti[1]. Kitti vehicle uses a bunch of sensors but we use data from the colour camera FL2-14S3C-Cand a Velodyne HDL-64E

- **Camera:** PointGray Flea2 colour camera (FL2-14S3C-C), 1.4 Megapixels, 1/2" Sony ICX267 CCD, global shutter. Images files are 8-bit PNG files and the Kitti dataset provides the rectified images.
- **Velodyne:** 1 × Velodyne HDL-64E rotating 3D laser scanner, 10 Hz, 64 beams, 0.09-degree angular resolution, 2 cm distance accuracy, collecting ~ 1.3 million points/second, field of view: 360^o horizontal, 26.8^overtical, range: 120 m.

In general, cameras can be classified as visible (VIS) or infrared (IR) based on the wavelength received by the device. The element used by the camera to sense the environment is known as an imaging sensor and uses two technologies: Charge-coupled device (CCD) and complementary metal-oxide semiconductor (CMOS). VIS cameras capture wavelengths between 400 nm to 780 nm, the same as the human eye can. IR cameras are passive sensors that work in infrared (IR) wavelengths ranges between 780 nm to 1 mm. Many devices work in this spectrum because fewer light interferences exist (e.g., LiDARs). Perception systems that include IR cameras work in near-infrared (NIR: 780 nm–3 mm) or mid-infrared (MIR: 3 mm–50 mm, known as thermal cameras) ranges. The uses of NIR usually replace or complement VIS cameras [2]. Another category of camera sensors, is Time-of-Flight (ToF) cameras. ToF cameras are active sensors that use the time of flight principle to obtain a 3D representation of the objects in the scene. ToF cameras emit NIR light pulses of 850 nm with an LED (Light Emitting Diodes)

As for LiDAR systems, they were initially developed in the 70s to measure elements in sea or land from satellites or aeroplanes. The ToF principle is used in LiDARs to measure the distance between emission and reception. They can be classified according to the type of information they obtain from their environment in 2D or 3D LiDARs or they can be classified according to their construction rotary or solid-state LiDAR. A pulsed light emitted from a laser diode until it is received by an emitter and the set of diodes lasers used are mounted on a pod that rotates at high speed. This is graphically illustrated in Figure 2.1. 3D LiDARs are most commonly used in autonomous driving and they allow to obtain a 3D map of the surrounding environment with high accuracy. Currently, 3D LiDARs can integrate from 4 to 128 channels with a horizontal FOV of 360 grades and vertical FOV that oscillates between 20–45 grades with the accuracy of a few centimetres.

[1] http://www.cvlibs.net/datasets/kitti-360/

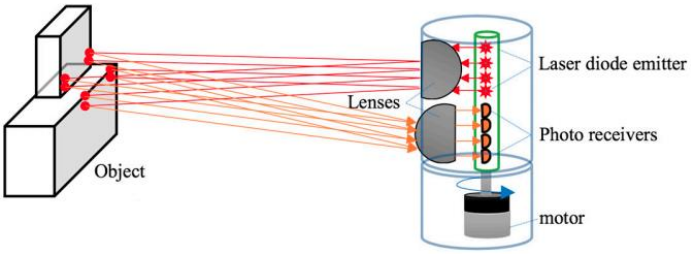


Figure 2.1. Rotating 3D LiDAR

2.1.2 Relevance to Attack Scenarios

In a computer vision system based on the camera sensor, image quality is the most important and moving automated vehicles brings a few challenges. Indeed, the image quality is affected by lenses, windshield, vibration, environmental conditions (e.g. light, rain, snow), potentially causing objects to get unnoticed. Exploiting the limitations of the camera sensor an attacker could corrupt the image. The most significant distinction between sensor attacks and cyber-attacks is the use of physical channels. Sensor attacks utilize the same physical channels as the targeted sensor in most cases, which can disrupt or manipulate the sensor readings. Since sensors are categorized as the lower layers of a control system and are normally trusted, falsified readings could lead to unexpected consequences of a system.

Assuming that the attacker is outside the vehicle (external attack) and targets sensor data acquisition, attacks on cameras can target its features such as automatic exposure controls, auto-focus or light sensitivity. Indeed, cameras normalize lighting conditions via an iterative process. When light is directed at the image sensor, it will tune down its sensitivity and exposure to improve the image quality according to predefined settings. This can lead to undesired effects, for instance when the auto exposure tunes down due to headlights at night. This could hide information in the background, such as traffic signs, road edges or pedestrians [3].

If we now assume that the attacker has access to the perception systems (internal attack) and its modalities these contain known bugs and vulnerabilities which can be exploited by the attackers. We are interested in attacks that can exploit the hardware and specifically camera sensors. A possible way to perform such an attack is through the ECU Firmware tampering attack. ECU (Engine control unit) is an electronic control module for the sensors and actuators of any sub-system in a vehicle and a typical vehicle consists of more than 100 ECU's [4]. Attackers target to reflash the ECU with custom firmware altering its state and inducing malicious and unintended actions. To perform this kind of attack physical access to the ECU is needed. Attacker updates the firmware of ECU using the external interface thus altering the functionality of ECU. By altering the ECU memory and tampering the security keys and maintaining the integrity of the ECU firmware code and its updates using the hashing techniques and authentication for software updating [4].

So, if an attacker manages by accessing the software to alter the output of the camera with adversaries the integrity of the whole system would be compromised. Adversaries alter original inputs with perturbations, which may be imperceptible to the human eye, but can force a trained model to produce incorrect outputs. Szegedy et al. [5] first discovered that state-of-art deep neural networks are susceptible to adversarial attacks. Speculative explanations suggested it was due to the extreme non-linearity of deep neural networks, combined with insufficient model averaging and insufficient regularization of the purely supervised learning problem. Studies on adversarial attacks have developed attacks for image classification models [6][7], used for multiple vision tasks such as object detection [8][9], object tracking [10], and semantic segmentation [9].

Overall as the automation level of the vehicle increases using outputs from one sensor gives greater flexibility to the intruder to harm the system. Additional information is needed and is the deliverable we will try to robust the perception system of the AV taking additional information from LiDAR. Environmental and lighting conditions, changed or not by an attacker, may affect camera sensors. LiDAR, in contrast, offers precise 3D measurement data over short to long ranges, regardless of

weather and lighting conditions. In Table 2.1 the LiDAR adaptation is summed up according to the Level of Autonomy. The Key Benefits of LiDAR are:

- Resolution & Accuracy: LiDAR generates instantaneous, massive amounts of measurements, and can be accurate to a centimetre.
- 3D Mapping: LiDAR data can be easily converted into 3D maps to interpret the environment.
- Low Light Performance: LiDAR is unaffected by ambient light variations, and performs well in low any light conditions.
- Speed: LiDAR data are direct distance measurements that don't need to be deciphered or interpreted – thus enabling faster performance and reducing processing requirement.

Table 2.1. LiDAR Adaptation per Level of Autonomy.

Level of Autonomy and Description	LiDAR Adaptation
Level 1 - Driver Only: Driving tasks are being executed exclusively by human drivers. **Example of Applications:** Automatic Emergency Braking (AEB), Adaptive Cruise Control (ACC), Lane Keep Assist (LKA)	Little to no LiDARs
Level 2 - Driver Assistance: The driver is responsible for either longitudinal or lateral control. The rest of the tasks can be automated to a certain extent by the assistance system. **Example of Applications:** Parking Assist (PA), Traffic Jam Assist (TJA)	Some will use LiDARs
Level 3 - Partial automation: The control unit takes over longitudinal and lateral control, although the driver shall be aware to take over control at any time needed **Example of Applications:** Highway Pilot	Most will use LiDARs
Level 4 - High automation: The control unit takes over longitudinal and lateral control while the driver is no longer required to be aware of monitoring the system. In case of a take-over request, the driver must take-over control with a certain time degree of request, the driver must take-over control with a certain time buffer. **Example of Applications:** Automated Urban Mobility	LiDAR is necessary
Level 5 - Full automation: The control unit takes over longitudinal and lateral control completely and permanently. In case of a takeover request that is not followed, the system will return to the minimal risk condition by itself. **Example of Applications:** Full Automation	LiDAR is necessary

2.1.3 Methodology Description

To segment, detect, and classify objects in an autonomous vehicle scene with robust and discriminative performance there are quite a few challenges that need to be addressed. Despite current state-of-art methods based solely on camera data seem to achieve astonishing results under normal imaging conditions, they fail in adverse weather and imaging conditions. Existing training datasets are biased towards clear weather conditions, and detector architectures are designed to rely only on the redundant information in the undistorted sensory streams. So, in a scenario that a sensor fails on specific conditions or, as in our case, has been attacked using a system relying on multiple sensor modalities gives a more robust result. There are three different fusion strategies that have been used so far in literature in order to exploit the advantages that each modality offers. So, we have early, late and deep fusion.

- **Early fusion:** Modalities are combined at the beginning of the process, creating a new representation that is dependent on all modalities.
- **Late fusion:** Modalities are processed separately and independently up to the last stage, where fusion occurs. This scheme does not require all modalities to be available as it can rely on the predictions of a single modality.
- **Deep fusion:** Modalities are mixed hierarchically in neural network layers, allowing the features from different modalities to interact over layers, resulting in a more general fusion scheme.

We choose to go with a late fusion strategy. Given that the camera is attacked the data coming from it would be unreliable to use them in the feature extraction level. In late fusion, or else decision fusion, multiple classifiers are used to generate decisions that are then combined to form a final decision, as shown in Figure 2.2.

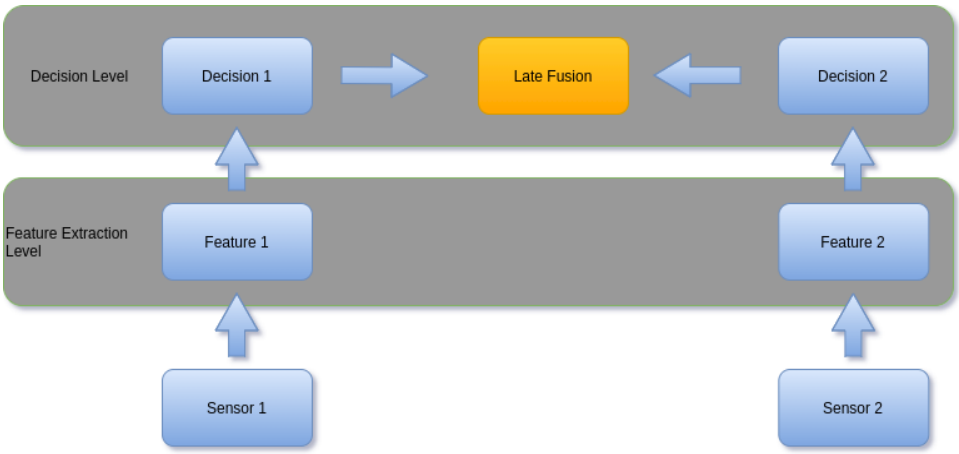


Figure 2.2. Abstract sensor fusion architecture describes where the late fusion occurs.

Specifically, in our case, we fuse the output of our segmentation model with the output of a deep learning model for 3D object detection based only on raw LiDAR frames. So, we can decide whether the camera has been attacked or not by correlating the two outputs. The overall architecture can be seen in Figure 2.3. Next, we will analyse the structure of the deep learning model that has been used for object detection.

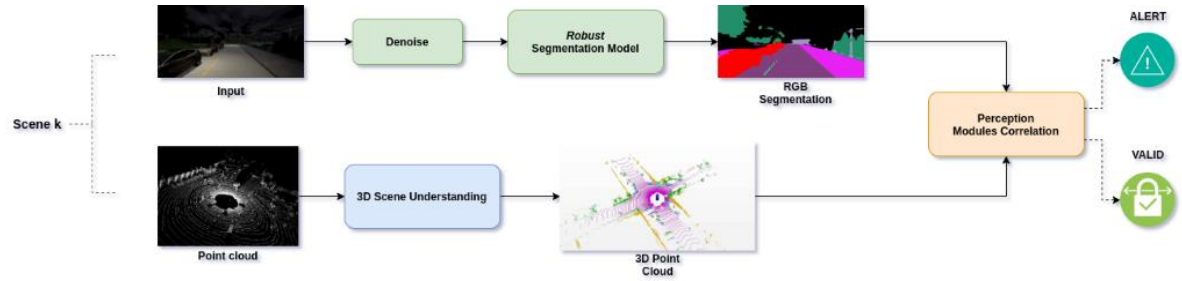


Figure 2.3. The overall pipeline of the perception module.

2.1.3.1 *Object detection Using DL on LiDAR*

A brief review of the state of art model on 3D object detection

Current state-of-art methods for 3D object detection proposed several ways to extract feature information from the sparse 3D point clouds. Many researchers tried to exploit pre-existing 2D CNN

architectures by projecting point cloud to bird's view for 3d box generation [11][12][13]. Another approach that has quite reliable results is to group the points into voxels and with the use of 3D CNN to learn the features of voxels to generate 3D boxes [14][15]. Due to the projection, and the voxelization stage the aforementioned methods suffer from loss of information. A significant step towards a better scene analysis of 3d raw point clouds was the network PointNet and PointNet++ [16][17] PointNet architecture directly learn point features from raw point clouds, which greatly increases the speed and accuracies of point cloud classification and segmentation, instead of representing the point cloud as voxels or multi-view formats. The network PointRCNN [18], that we used for object detection uses PointNet as a backbone. More details about PointRCNN [18], are provided in the next section.

PointRCNN architecture for Point Cloud 3D Detection

The model that we used is PointRCNN [18] and is one of the current state-of-art models for 3D object detection. The overall architecture of the model as it is proposed in [18] is illustrated in Figure 2.4. PointRCNN [18] is a bottom-up point cloud-based 3D bounding box proposal generation algorithm, which generates a small number of high-quality 3D proposals via segmenting the point cloud into foreground objects and background. So, the model consists of a bottom-up 3D proposal generation stage and a stage for the refinement of the bounding boxes.

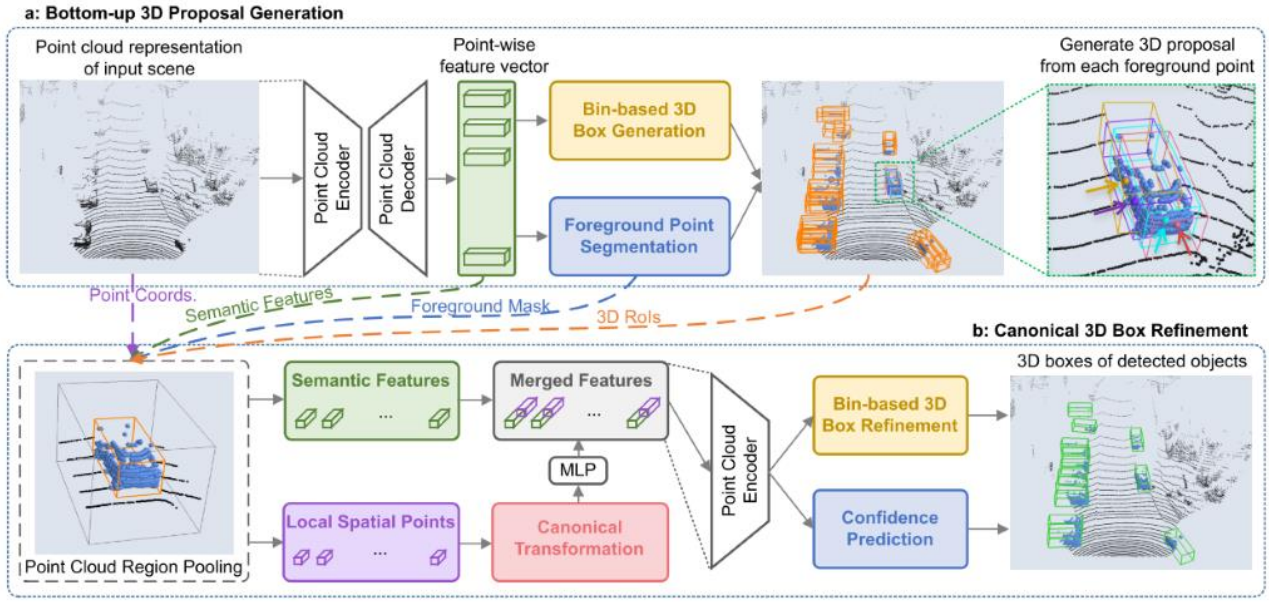


Figure 2.4. [18] The PointRCNN architecture for 3D object detection from point clouds. The whole network consists of two parts: (a) for generating 3D proposals from raw point cloud in a bottom-up manner. (b) for refining the 3D proposals in canonical coordinate.

Bottom-up 3D proposal generation via point cloud segmentation: In order to learn pointwise features for describing the raw point cloud, PointRCNN [18] uses as a backbone network PointNet++ [16]. The network uses foreground points to gain some knowledge of the locations and orientations of the associated objects.

- Foreground points: All 3D objects' segmentation masks could be directly obtained by their 3D bounding box annotations. 3D points inside 3D boxes are considered as foreground points.

Given the pointwise features encoded by PointNet++ [16] one segmentation head is appended, for estimating the foreground mask and one box regression head for generating 3D proposals. For point segmentation, the ground-truth segmentation mask is naturally provided by the 3D ground-truth boxes. Thus, the focal loss [19] is used to handle the class imbalance problem as:

$$L_{focal}(p_t) = -a_t(1-p_t)^{\gamma} log(p_t)$$

, where $p_t = p$ for foreground point and $p_t = 1 - p$ otherwise.

Simultaneously with the foreground point segmentation problem, a box regression problem is handled. A 3D bounding box is represented as $(x, y, z, h, w, l, \theta)$ in the LiDAR coordinate system, where (x, y, z) is the object centre location, (h, w, l) is the object size, and θ is the object orientation from the bird's view. For the estimation of the centre location, the neighbourhood for each foreground point is being split in discrete bins along the X and Z axes instead of handling a direct regression problem.

The localization loss for the Xor Z axis consists of two terms, one term for bin classification along each X and Z axis, and the other term for residual regression within the classified bin. The centre location y along the vertical Yaxis, using the $L1$ loss is enough for obtaining accurate y values because y values are within a very small range.

The localization targets are formulated as:

$$bin_x^{(p)} = \left\lfloor \frac{x^p - x^{(p)} + S}{\delta} \right\rfloor, bin_z^{(p)} = \left\lfloor \frac{z^p - z^{(p)} + S}{\delta} \right\rfloor,$$

$$res_u^{(p)} = \frac{1}{C}(u^p - u^{(p)} + S - (bin_u^{(p)} \cdot \delta + \frac{\delta}{2})), u \in \{x, z\}$$

$$res_y^{(p)} = y^p - y^{(p)}$$

- $(x^{(p)}, y^{(p)}, z^{(p)})$ are the coordinates of a foreground point of interest,
- (x^p, y^p, z^p) is the centre coordinates of its corresponding object,
- $bin_x^{(p)}$ and $bin_z^{(p)}$ are the ground-truth residual for further location refinement within the assigned X and Z axis,
- $res_x^{(p)}$ and $res_z^{(p)}$ are the ground-truth residual for further location refinement within the assigned bin,
- C is the bin length for normalization.

The estimation of the orientation θ and size (h, w, l) is done based on [20]. The overall 3D regression loss L_{reg} with different loss terms for training could then be formulated as:

$$L_{bin}^{(p)} = \sum_{u \in \{x,z,\theta\}} (F_{cls}(\widehat{bin}_u^{(p)}, bin_u^{(p)}) + F_{reg}(\widehat{res}_u^{(p)}, res_u^{(p)})),$$

$$L_{res}^{(p)} = \sum_{v \in \{y,h,w,l\}} (F_{reg}(\widehat{res}_v^{(p)}, res_v^{(p)}),$$

$$L_{res}^{(p)} = \frac{1}{N_{pos}} \sum_{p \in pos} (L_{bin}^{(p)} + L_{res}^{(p)})$$

N_{pos} is the number of foreground points

- $\widehat{bin}_u^{(p)}$ and $\widehat{res}_u^{(p)}$ are the predicted bin assignments and residuals of the foreground point p
- $bin_u^{(p)}$ and $res_u^{(p)}$ are the ground-truth targets calculated as above
- F_{cls} denotes the cross-entropy classification loss,
- F_{reg} denotes the smooth L1 loss.

- **Point cloud region pooling**

After having the region proposal for the 3D bounding box for refining the specific location PointRCNN [18] proposes to pool 3D points. So, each 3D box proposal $b_i = (x_i, y_i, z_i, h_i, w_i, l_i, \theta_i)$ is enlarged by a constant n and each point that is inside the enlarged 3D bounding box is being kept, alongside its features for refining the box bi. The features associated with the inside point p include its 3D point coordinates $(x^{(p)}, y^{(p)}, z^{(p)}) \in R$, its laser reflection intensity $r^{(p)} \in R$, its predicted segmentation mask $m^{(p)} \in \{0,1\}$ from stage-1, and the C-dimensional learned point feature representation $f^{(p)} \in R^C$ from stage-1.

- **Canonical 3D bounding box refinement**

The pooled points and their associated features for each proposal are given as input to the stage-2 sub-network. Each pooled point is being transformed into the canonical coordinate system of the corresponding 3D proposal. This means that the origin is located at the centre of the box proposal and that the local $X'$and Z' axes are approximately parallel to the ground plane with X' pointing towards the head direction of the proposal and the Z' axis is perpendicular to X'.Y' axis remains the same as that of the LiDAR coordinate system.

The canonical transformation enables robust local spatial features learning although because it loses depth information of each object the distance to the sensor is included in the features of point p. The local spatial features are first concatenated and fed to several fully connected layers to encode their local features to the same dimension of the global features $f^{(p)}$. Then the local features and global features are concatenated and fed into a network and a discriminative feature vector is obtained for the following confidence classification and box refinement. For box proposal refinement the bin-based regression losses for proposal refinement is adopted as in stage one.

2.1.3.2 *Fusion Scheme*

From LiDAR coordinate system to camera coordinate system: As presented in [21], for the sensors to be synchronized the timestamps coming from the LiDAR are used as a reference and each spin is considered as a frame. The LiDAR keeps rotating to collect the data and the camera is triggered every time it faces forward. To project a 3D point $X = (x, y, z, 1)^T$ in the rectified (rotated) camera coordinates to a point $Y = (u, v, 1)^T$:

$$Y = P_{rect} X,$$

$$P_{rect} = \begin{pmatrix} f_u & 0 & c_u & -f_u b_x \\ 0 & f_v & c_v & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}$$

b_x denotes the baseline (in meters) with respect to the reference camera.

Note that in order to project a 3D point x in reference camera coordinates to a point Y on the image plane, the rectifying rotation matrix of the reference camera R_{rect} must be considered as well:

$$Y = P_{rect}\, R_{rect}\, X$$

R_{rect} has been expanded into a 4x4 matrix by appending a fourth zero-row and column and setting $R_{rect}(4,4) = 1$

Velodyne laser scanner with respect to the reference camera coordinate system is registered using [22]. The rigid body transformation from Velodyne coordinates to camera coordinates are given from:

- $R_{velo}^{cam} \in R^{3x3}$..... rotation matrix: Velodyne → camera
- $t_{velo}^{cam} \in R^{1x3}$..... translation vector: Velodyne → camera
- using $T_{velo}^{cam} = \begin{pmatrix} R_{velo}^{cam} & t_{velo}^{cam} \\ 0 & 0 \end{pmatrix}$

Hence, a 3D point x in Velodyne coordinates gets projected to a point Y in the camera image as below:

$$Y = P_{rect} R_{rect}\, T_{velo}^{cam}\, X$$

Fusion Scheme Architecture: So, after we perform semantic segmentation to the attacked image, the output will resemble Figure 2.5 image (b). Most of the vehicles are hidden from the prescription engine due to the attack on the camera sensor. In Figure 2.5 image (a) we can see the output of the segmentation model in the non-attacked image.

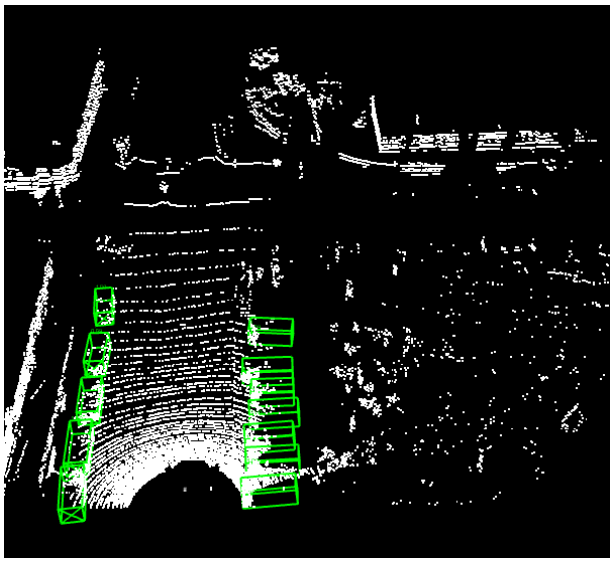
(a)	

(b)

Figure 2.5. Image (a) is the output of the segmentation model to the non-attacked image while in the image (b) is the output of the segmentation model to the attacked image. The red mask indicates the detected vehicles.

In a parallel module, the object detection model is running, which has been trained to identify only the moving objects which are mainly vehicles, pedestrians and cyclists. In Figure 2.6(a) we can see the 3D bounding boxes of the vehicles that have been detected on LiDAR sensor data. Most of the vehicles have been identified from the model. After obtaining the coordinates of the 3D bounding boxes, they are being projected to the images plane in order to correlate the 2D image segmentation output and the 3D object detection output (Figure 2.6(b)).

(a)



(b)

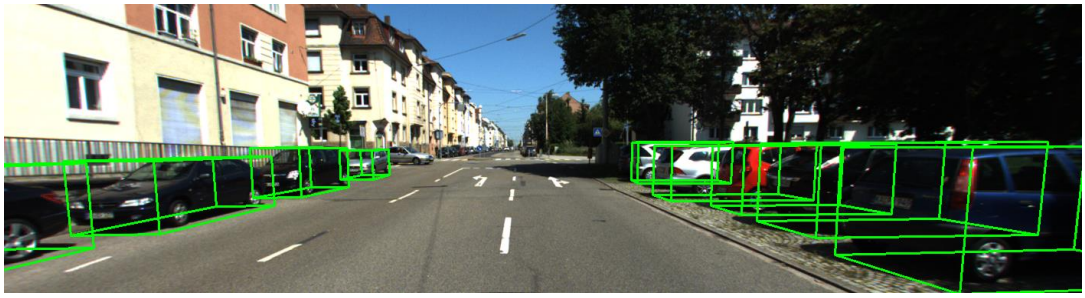


Figure 2.6. (a)The output of PointRCNN to 3D space in the LiDAR coordinate system. (b)The projected output of PointRCNN to the image plane.

More specifically in order to correlate the two outputs, we isolate the region of the projected 3d bounding box to the image. We consider that the isolated region belongs to a specific class (vehicle, pedestrian, cyclist). We isolate respectively the same region from the segmentation mask. Finally, we compare the two outputs in order to estimate the overlap between the two segmentation masks. If the object has been detected by both of the modalities the overlap should be high enough. Structural similarity index measure (SSIM) [23] is used for the comparison and should be above 0.6. The scheme for comparing the two outputs is presented in the Figure 2.7.

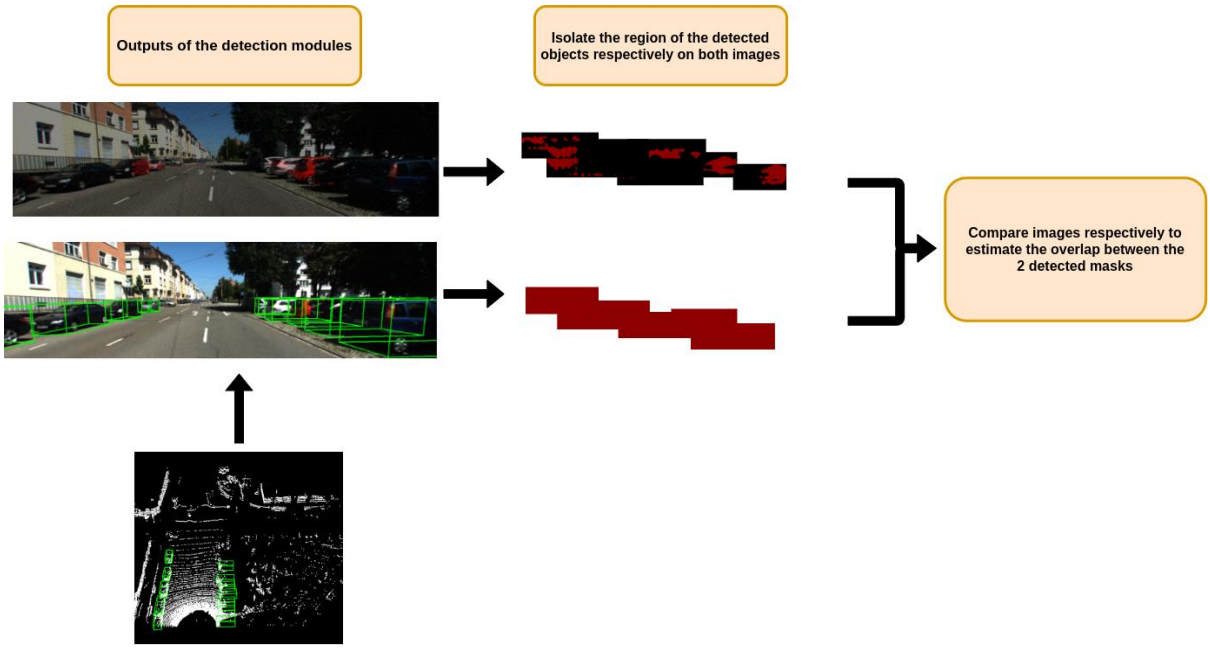


Figure 2.7. Comparison scheme. The 3D isolated projected objects are compared respectively with the segmentation output.

2.1.4 Validation of Methods

The proposed pipeline aiming to provide improved situational awareness to the user consists of two modules. The first one refers to the robust image segmentation model and the second to the 3D object detector using the LiDAR sensor. Hence, taking into consideration the first model, some evaluation metrics are shown below. Multiple attacks were implemented with different sizes of perturbation for evaluation purposes. Some of the implemented attacks on the camera sensor are shown below:

- Untargeted attacks
    - BIM
    - FGSM
    - PGD
- Targeted attacks referring to cars
    - LinfPGD
    - MomentumIterative

The Intersection over Union (IoU) is the primary metric used in evaluating segmentation outputs. The evaluation of the robust segmentation model of DeeplabV3 on the KITTI benchmark are illustrated in Table 2.2 and Table 2.3. Table 2.2 refers to untargeted attacks and Table 2.3 to targeted attacks.

Table 2.2. Evaluation results (IoU) for untargeted attacks between robust and original segmentation.

Resnet = 152	**BIM**				**FGSM**				**PGD**			
Model	e=2	e=4	e=8	e=16	e=2	e=4	e=8	e=16	e=2	e=4	e=8	e=16
Robust	**0.42**	**0.1**	**0.03**	**0.03**	**0.75**	**0.71**	**0.66**	**0.5**	**0.74**	**0.72**	**0.72**	**0.52**
Original	0.13	0.02	0.01	0.01	0.67	0.51	0.21	0.03	0.69	0.53	0.17	0.03

Table 2.3. Evaluation results (IoU) for targeted attacks between robust and original segmentation model.

Resnet = 152	LinfPGD							
	target = 2				target = 7			
Model	e=2	e=4	e=8	e=16	e=2	e=4	e=8	e=16
Robust	**0.75**	**0.7**	**0.62**	**0.5**	**0.75**	**0.71**	**0.63**	**0.46**
Original	0.69	0.58	0.37	0.17	0.69	0.53	0.26	0.1
Resnet = 152	MomentumIterative							
	target = 2				target = 7			
Model	e=2	e=4	e=8	e=16	e=2	e=4	e=8	e=16
Robust	**0.71**	**0.68**	**0.62**	**0.35**	**0.72**	**0.68**	**0.61**	**0.34**

Original	0.65	0.5	0.17	0.02	0.65	0.37	0.12	0.02

From Table 2.2 and Table 2.3, we observe that our goal to provide a robust segmentation model for adversarial attacks in 2D space has been achieved. Next, some evaluation results of the 3D object detector coming from the LiDAR data are illustrated.

The evaluation of the 3D object detection model PointRCNN tested on the KITTI benchmark is shown in Table 2.4. For the 3D detection of car and cyclist, PointRCNN method outperforms previous state-of-the-art methods with remarkable margins on all three difficulties and ranks first on the KITTI test board among all published works at the time of its submission. Although most of the previous methods use both RGB image and point cloud as input. For pedestrian detection, compared with previous LiDAR-only methods, our PointRCNN method achieves better or comparable results, but it performs slightly worse than the methods with multiple sensors.

Table 2.4. [18] Performance comparison of 3D object detection with previous methods on KITTI test split by submitting to the official test server. The evaluation metric is Average Precision (AP) with IoU threshold 0.7 for car and 0.5 for pedestrian/cyclist.

Method	Modality	Car (IoU=0.7)			Pedestrian (IoU=0.5)			Cyclist (IoU=0.5)		
		Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
MV3D	RGB+LiDAR	71.09	62.35	55.12	-	-	-	-	-	-
UberATG-ContFuse	RGB+LiDAR	82.54	66.22	64.04	-	-	-	-	-	-
AVOD-FPN	RGB+LiDAR	81.94	71.88	66.38	50.80	42.81	40.88	64.00	52.18	46.61
F-PointNet	RGB+LiDAR	81.20	70.39	62.19	51.21	44.89	40.23	71.96	56.77	50.39
VoxelNet	LiDAR	77.47	65.11	57.73	39.48	33.69	31.51	61.22	48.36	44.37
SECOND	LiDAR	83.13	73.66	66.20	51.07	42.56	37.29	70.51	53.85	46.90
PointRCNN	**LiDAR**	**85.94**	**75.76**	**68.32**	**49.43**	**41.78**	**38.63**	**73.93**	**59.60**	**53.59**

Finally, some more experiments on the KITTI dataset were conducted to evaluate the overall proposed pipeline, that combines the 2D image segmentation with the 3D object detector. Hence, the LiDAR module is triggered only when an external attack to the camera sensor has been detected. In a safe situation, the final result is coming only from the camera sensor, as we can observe from the green cells from Table 2.5. On the other hand, when a dangerous situation has been detected, the output of the 3D detector is given to the output of the perception engine, ignoring the camera results. In the latter cases, the model performs better as we can observe from orange cells in Table 2.5.

Table 2.5. Evaluation results (IoU) fusing multiple sensor data.

Data Type		IoU Camera		IoU Camera + LiDAR		Samples
Normal		**76**		**76**		250
Attacked		**eps**				
		2	16	2	16	
Untargeted Attacks	FGSM	**75**	**50**	**75**	**82**	250
	PGD	**74**	**52**	**74**	**81**	
	BIM	**42**	**3**	**82**	**81**	
Targeted Attacks	Momentum Iterative	**71**	**35**	**71**	**80**	
Fused Model Accuracy				**78**		500

2.2 Validating Traffic Sign Attacks with GPS measurements

The fusion of sensors such as LiDAR, Camera and GPS are commonly used in perception engines in autonomous vehicles [24]. Fusion of multisensory data can provide not only insights on the events but also can be used for robustification of an existing pipeline. There are a number of applications to apply sensor fusion such as 3D object detection [25], improved GPS area localisation [26], occupancy grid mapping [27], tracking of moving objects [28] and much more. Likewise, post-analysis of using such sensor fusion mechanisms also can provide an in-depth knowledge on vehicles' behaviours and in particular for the identification of anomalies.

GPS sensors are an essential part of autonomous vehicle systems. It provides geo-location information which enables autonomous vehicles to navigate from one place to another [29]. Likewise, camera sensors are another vital part of autonomous vehicles which capture images of the external environment. These images are used to perform scene analysis that is essential for autonomous vehicles, including tasks such as detection of objects and pedestrians [30], traffic lane [31], and traffic signs recognition [32] using Machine Learning approaches.

2.2.1 Relevance to Attack Scenarios

A specific scenario was considered to demonstrate the use of multi-sensor fusion mechanisms using Deep Learning architecture to detect anomalies. A vehicle simulator called CARLA [33] has been used to generate the training data as well as a validation environment. The study examines both internal components such as GPS as well as external environments such as traffic signs. As shown in the Figure 2.8 and Figure 2.9, six different scenarios are considered in this experiment and the proposed multi-model fusion will detect abnormal activities of vehicles based on GPS, speed and traffic signs. Figure 2.8, shows the correct behaviour of the autonomous vehicles where the vehicle manages to follow multiple traffic signs correctly such as correct direction under the speed limit. Figure 2.9 shows the abnormal behaviour of the vehicle where either the vehicle is moving over the speed limit or moving in the wrong direction.

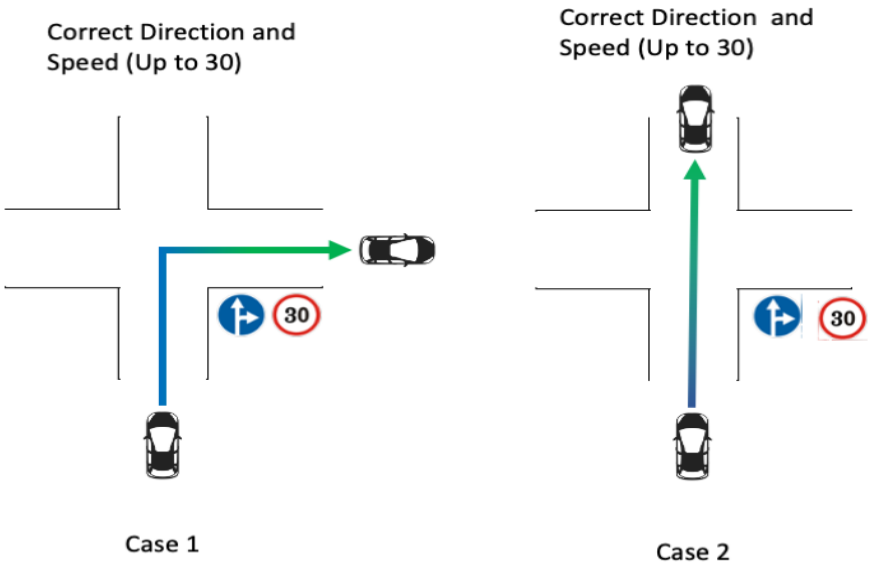


Figure 2.8. The correct cases of the autonomous vehicles' behaviours in regards GPS and the traffic sign data.

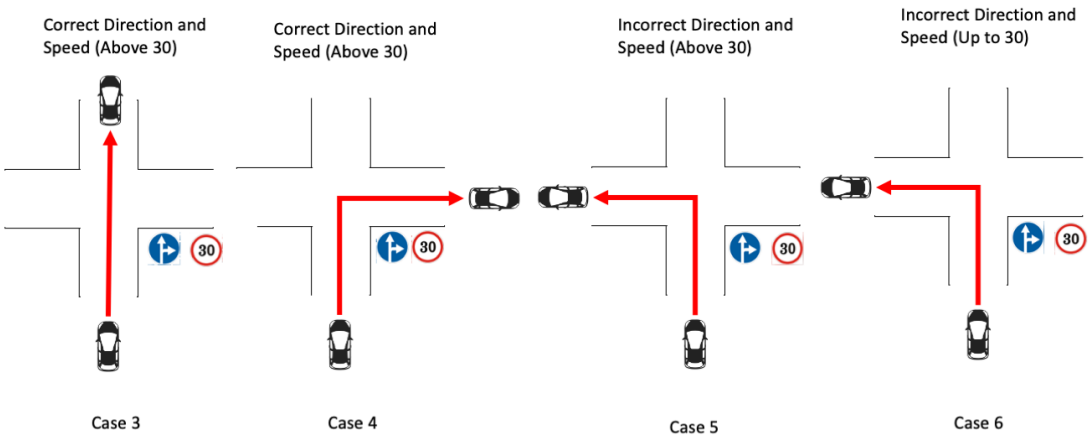


Figure 2.9. The incorrect cases of the autonomous vehicles behaviours in regards GPS and the traffic sign data.

2.2.2 Sensors and Measurements

The three types of sensors have been selected a) GPS, b) Speedometer and b) Camera. The data captured by GPS, Speedometer and Camera are processed further based on the requirement of the fusion model.

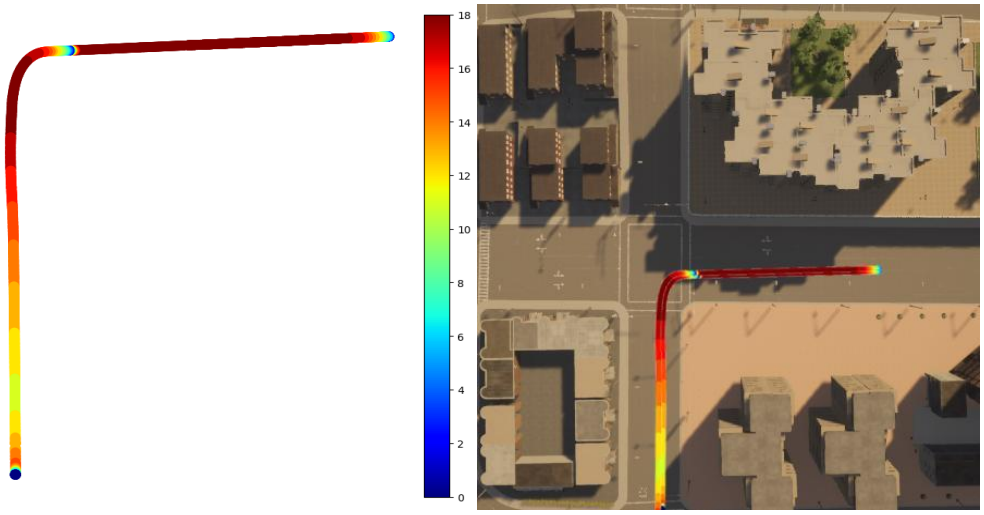


Figure 2.10. Left - 2D GPS location with colour indication of the vehicle's speed. Right - Top down view of the data overlaid on the CARLA [33] environment.

2.2.3 Methodology Description

There are a number of ways multi-sensor fusion can be achieved. However, image-based methods have been used to demonstrate the capability of fusion mechanisms. The image-based approach allows the architecture to be simple and effective using Convolutional Neural Networks (CNN). CNNs are widely used in Computer Vision (CV). It has also been shown that CNNs work well in other fields such as GPS trajectories prediction [34], Traffic Speed regression [35], and much more.

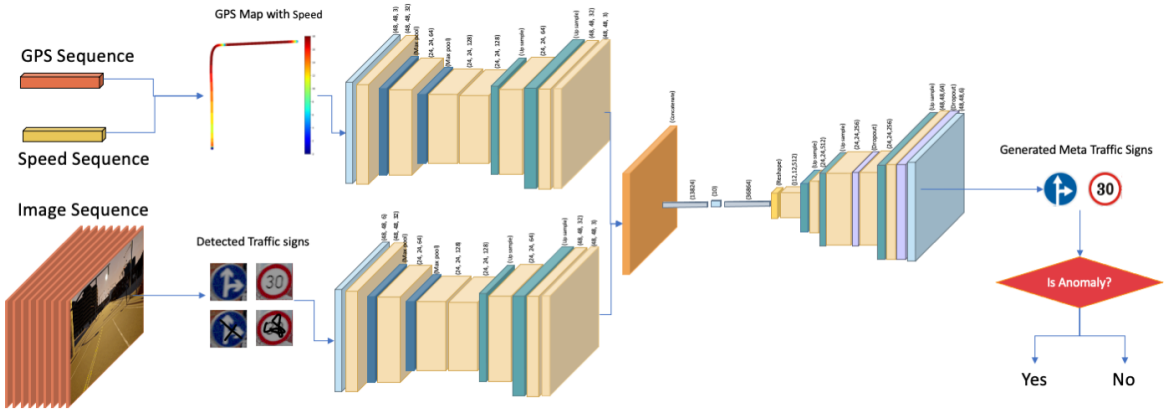


Figure 2.11. The overall flowchart of detecting anomaly behaviours using multiple-sensors in autonomous vehicles.

2.2.3.1 *Data Generation*

The CARLA [33] Simulator was used to generate data for training and testing purposes. Three types of data were captured: GPS location, vehicle speed and a sequence of image frames. By default, GPS values are in a longitude and latitude format including a timestamp and speed in number format. These values were converted into X and Y points/pixel coordinates while the speed was represented with coloured heatmap as shown in Figure 2.10 (left). Both the input data (GPS and traffic signs) are in an RGB (Red, Green and Blue) format with fixed resolution. Figure 2.12 shows the overall flow chart of the data generation process.

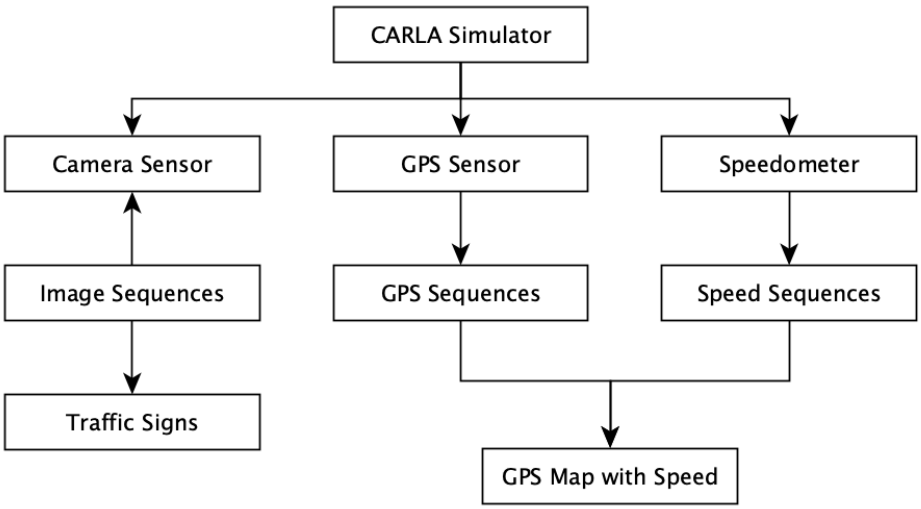


Figure 2.12. Flowchart of data generation process.

Table 2.6. Total data generated from CARLA [33] Simulator.

Type	Image Sequence Sets	GPS Sequence Sets	Speed Sequence Sets
Training	87	87	87
Testing	108	108	108

As shown in Table 2.6, a total of 87 sets of Image frames, GPS and Speed value sequences were generated for training purposes. All the sets for training represent the normal behaviours of the autonomous vehicles where the car follows the direction and the speed limit according to the traffic signs. Whereas, the test set includes 54 sets of normal and 54 sets abnormal behaviour data as shown in Figure 2.9. Likewise, each set has at least 1000 GPS points. In addition, a meta traffic sign has been generated to train the model. The meta traffic sign is a clean version of traffic signs without any artefacts.

2.2.3.2 *Architecture*

A Deep Learning model has been developed to identify the abnormal behaviour of the autonomous vehicle. The model is based on Generative Adversarial Network (GAN) architecture where two different networks generator and discriminator compete against each other during the training. The generator has two-head architecture which takes two different inputs: GPS maps and traffic signs. The primary goal of the generator is to map the GPS, Speed and traffic signs into appropriate meta traffic signs. Whereas the discriminator learns to distinguish the original meta traffic sign and generated ones. Figure 2.13 shows the overall architecture of the developed GAN and Table 2.7, shows the output shape of the model and the corresponding parameters in each network. In total there are 5,015,159 parameters in the proposed GAN. During the training phase, both models were utilised however only the generator was used for testing purposes.

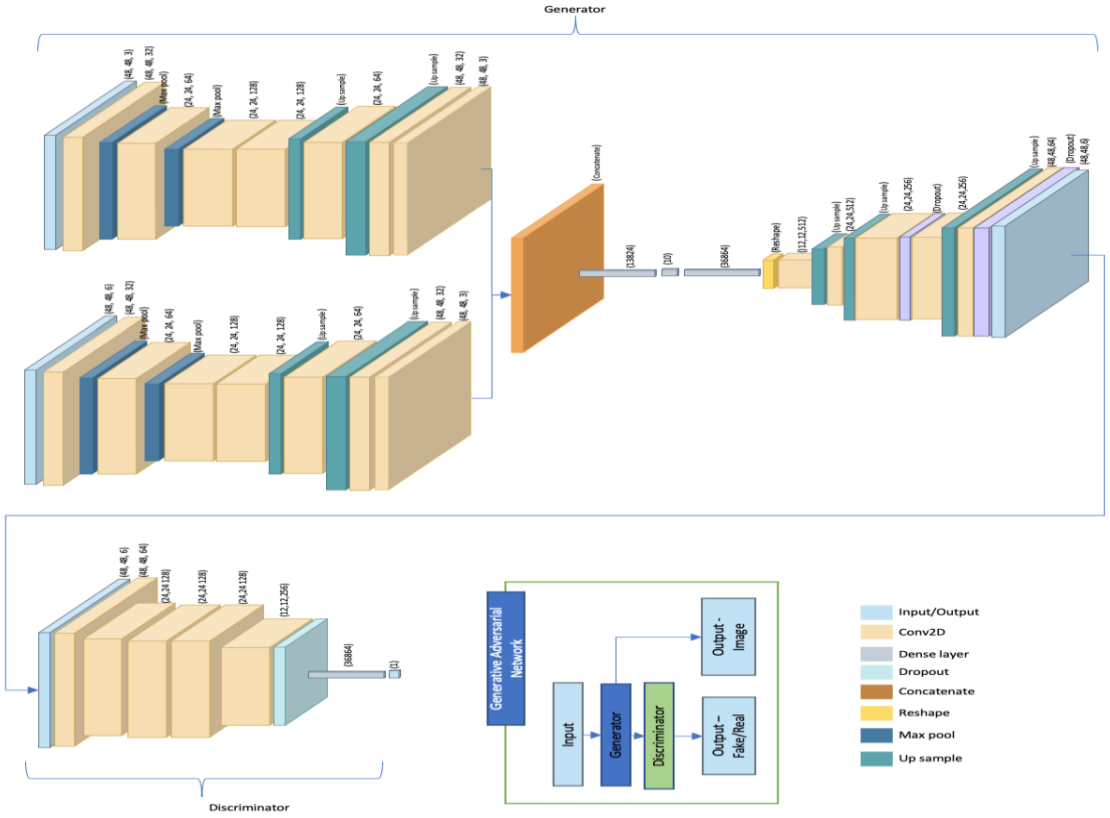


Figure 2.13. The GAN architecture of Anomaly detection model.

Table 2.7. Output and total number of parameters in GAN architecture.

Layer (type)	Output Shape	Parameters
input_1	48x48x3	0
input_2	48x48x6	0
Generator (Model)	48x48x6	4310582
Discriminator (Model)	1	704577
	Total parameters	5,015,159

The core attribute of the generator is the dimensionality reduction that is performed following the architecture of Autoencoder [36]. While the Autoencoder is an unsupervised method, the generator is a supervised one where inputs and outputs are provided during the training phase. The principal aim of the generator is to learn the mapping of the GPS data and the traffic signs, as precisely as possible. Once the network is trained, it is assumed that it has learnt the characteristics of the input and target data. Finally, only normal data were used to train the network and it is assumed the model can reconstruct with high accuracy data similar to the ones used during the training stage. Hence, this property of the proposed architecture can be used for detecting abnormal data considering them as failed reconstructions.

Table 2.8. The two-head architecture of the generator used the same layout till the concatenation layers.

Layer	Output Shape	Kernel size	Strids	Activation function	Filters
input_1	48x48			ReLU	3
conv2d_1	48x48	3		ReLU	32
max_pooling	24x24		2		
conv2d_2	24x24	3		ReLU	64
conv2d_3	24x24	3		ReLU	64
max_pooling	12x12		2		
conv2d_4	12x12	3		ReLU	128
dropout					
conv2d_5	12x12	3		ReLU	128
up_sampling	24x24		2		
conv2d_6	24x24	3		ReLU	64
up_sampling	48x48		2		
conv2d_7	48x48	3		ReLU	32
conv2d_8	48x48	3		ReLU	3
dropout_2	48x48				

Table 2.9. The overview of the tail of the generator which combines the two heads.

Layer	Output Shape	Kernel size	Strids	Activation function	Filters
concatenate_1	48x48				6
flatten_1	13824				13824
dense_1	10			ReLU	10

dense_2	36864			ReLU	36864
reshape_1	12x12				256
conv2d_1	12x12	**3**		ReLU	512
up_sampling	24x24				
conv2d_2	24x24	**3**		ReLU	256
dropout					
conv2d_3	24x24	**3**		ReLU	256
up_sampling	48x48				
conv2d_4	48x48	**3**		ReLU	64
dropout					
conv2d_5	48x48	**1**		Tanh	6

The Table 2.8, shows the overall structure of the generator. In total there are 42 layers including the input and output. The network used a 3x3 kernel for feature extraction and max pooling to reduce the shape of the features and upsampling to reconstruct them. In addition, L2 [37] and dropout [38] regularisation methods have been used throughout the network. Likewise, the ReLU activation function has been used except the last layer where the tanh function was used instead. The generator receives as input a 48x48x3 GPS map and 48x48x6 traffic signs (two different traffic signs stacked in channel dimension) and outputs 48x48x6 meta traffic signs. The 6D output is split into two parts (2 images of size 48x48x3) aiming to extra the two types of meta traffic signs (see Figure 2.11).

2.2.3.3 *Training*

The Deep Learning model was trained and evaluated on the NVIDIA GTX 2080Ti GPU using Keras-Tensorflow framework. The data were pre-processed before the training phase, such as resizing the GPS map and the traffic signs to an appropriate image resolution. During the training phase, additional data augmentation techniques were used. Table 2.10 shows all the augmentation methods that were used during the training phase.

Table 2.10. All the augmentation methods that were applied on the training data for the anomaly detection model.

Type	**Parameters**	**Description**
Rotation range	0 - 20 degrees	The training image i.e., GPS map is randomly rotated between 0 - 20 degrees.

Zoom range	0 - 40 percent	Zooming is applied on the training images by a random scaling factor in the range of 0 - 40 percent.
Width and height shift range	0 - 10 percent	The images are shifted left, right, up or down in random combinations and magnitudes in the range of 0 - 10 percent.
Shear	0 - 10 percent	The images are stretched randomly in the range of 0 - 10 percent.
Fill	Nearest mode	Missing pixel values are filled with nearby pixel values.
Horizontal and Vertical Flip	True	The images are flipped in a horizontal and vertical direction randomly. Only used for the GPS map

2.2.4 Validation of Methods

To validate the anomaly detection model, a test dataset was generated where 50% of data were normal and remaining abnormal. A total of 108 sequences of GPS maps and traffic signs were used for the model evaluation. The Mean Square Error (MSE) was selected to calculate the reconstruction errors. The threshold value was retrieved from the 50% quantile of reconstruction error. The value was then used to validate the performance of anomaly detection.

Table 2.11. The performance of the model with the threshold value of 0.09572.

Data Type	Precision	Recall	f1-score	Samples
Normal	0.82	0.78	0.80	54
Anomaly	0.79	0.83	0.81	54
Model Accuracy			0.81	108

Table 2.11 shows the performance of the anomaly detection model. The threshold value of 0.09572 produced the best performance of the model achieving 0.80 and 0.81 F1-score for normal and abnormal data respectively.

3 Sensor Fusion Solutions for Detecting GPS Location Spoofing Attacks

3.1 Relevance to Attack Scenarios

The security and intact operation of GPS sensor, is a crucial factor of the feasibility of VANET. Relying on GPS measurements and aided by a precise high definition map, vehicles can choose an optimized, shortest path from one location to another location. This is essential for vehicles to work correctly and autonomously without the assistance of human drivers [39]. However, as shown in [40][41][42], GPS is susceptible to attacks, such as jamming and spoofing. Jamming aims to fully block GPS operation by launching disruptive signals with the same frequencies as those of GPS. The more prominent threat of GPS spoofing is related to deceiving the user by the transmission of signals with the same characteristics of legitimate GPS satellite signals. There are multiple ways and resources (open source) available for spoofing, that pose a critical threat to GPS. Therefore, there is a great concern over the security and safety aspects. After encountering few unexpected and unprecedented attacks over the GPS sensors several research and studies are opening the ways as how these sensor systems can be made secured and risk free. Initially commercially available off the shelf receivers were used to study about the possible threats [43] which were not given much of a concern. The study [43] also gave an insight on topics such as GPS and speculations about the intended use, an alternative navigation system which can be used apart from GPS and the significant points which should be considered to have a secured and safe communication.

GPS vulnerability to spoofing can be seen on three distinctive levels:

1. Vulnerability in signal processing: The type of GPS signals (e.g. modulation type, transmit frequency, etc.) is publicly known. An attacker can generate signals with similar characteristics to the true ones and deceive the user.
2. Vulnerability in data bit level: Framing structure like satellite ephemeris and clock is also known and does not change rapidly. As such, the attacker can also exploit that information to generate deceiving signals.
3. Vulnerability in Navigation and Position Solution: The attacker can inject counterfeit pseudo range measurement and lead to wrong position, velocity, and time solution for the legitimate GPS receiver.

An advanced attacking strategy requires the attacker to be patient [44][45][46]. To launch an attack toward a legitimate receiver, the disruptive signals of the attacker should synchronize on the signals from the satellite. After synchronization, the attacker increases the power of signals, which makes the victim's GPS lock on spurious signals. Then, the attacker can manipulate the position of the victim by changing spurious signals. GPS spoofing techniques include Lift-off-delay, Lift-off-aligned, Meaconing or Replay, Jam and Spoof, Trajectory spoofing, etc. Techniques that try to defend against spoofing are based on Signal Power Monitoring, Signal Arrival Characteristics, Signal Correlation Peak, Antenna Array and Multi-Sensor Fusion. The latter defence approach fits with the rationale of Cooperative Localization in VANET. The nodes of the network collaborate between themselves, exchange measurements and aim not only to improve location accuracy but also, to detect and mitigate possible location attacks.

3.2 Fusing in-vehicle measurements for detecting location spoofing attacks

3.2.1 Sensors and Measurements

When the safety and security related issues are discussed for the vehicle, monitoring the vehicle's location information or tracking the vehicles position is important. Likewise, the safety and security pertaining to the sensors involved in this scenario is of high importance. The machine learning algorithm which would be addressing these topics such as to detect the abnormalities with the sensor inputs and

mitigate the attacks relies on two aspects. The initial GPS sensor inputs which would be fed to the vehicle to navigate and then the vehicle parameters which are coming from the other relevant sensors in the vehicle. From GPS sensors the measurements related to the location calculation and number of satellites are considered as the important parameters. Since, the whole idea of the machine learning algorithm is to address the spoofing attack detection, as an alternate technique to calculate the position or location information, vehicle parameters like acceleration, yaw rate, velocity, etc., are considered. From start of the vehicle to until it reaches the desired destination GPS sensor measurements are being monitored for the spoofing attack. Parallelly, the vehicle parameters are considered, and the deviation pertaining to the position is evaluated. If the deviation is above a certain threshold then the GPS sensor measurements would no longer be considered for the navigation but vehicle parameters and other alternate techniques by using LiDAR, camera sensors would be used. More specific, the sensors and measurements used for this in-vehicle GPS location spoofing detection is listed below:

- Speed [*m/s*]; (*rotation speed of vehicle*).
- Compass [*deg*]; (*direction based North and South magnetic poles of the Earth*).
- GNSS; (*vehicle position: Lat, Long, Altitude and Time information*).
- Steering angle sensor [*deg*]; (*Steering angle of the vehicle*).
- IMU sensor; (*includes Gyroscope [rad/s] and Accelerometer [m/s2]*)

3.2.2 Methodology Description

Connected and Autonomous Vehicle (CAV) technology advancements led to the need for an accurate and robust localization system. However, due to satellite signal jamming and location spoofing attacks, the Vehicle-to-Vehicle/Infrastructure (V2V/V2I) communication and navigation disruptions are becoming a significant obstacle to CAV's operation. Many solutions to this problem have been proposed e.g. Galileo anti-spoofing service on the civil GNSS signal known as Open Service Navigation Message Authentication (OS-NMA), which enables the authentication of the navigation data. However, despite anticipation, no integrated circuit designs for OS-NMA on E1 frequency have been released to date and some experts question the usefulness of such solution if receivers can deliver anti-spoofing protection based on inertial sensors or signal processing [47].

In this section, the methodology of GPS spoofing attack detection will be analyzed and discussed. The framework of the attack identification based on in-vehicle GPS location integrity check, is illustrated in Figure 3.1. The CARAMEL system is able to compute a parallel GPS-free location stream using a fallback location solution based on Bayesian filtering which consists of two main steps namely (i) the prediction step and (ii) the update step. In the prediction step, by means of the on-board sensors through the vehicle's Controller Area Network (CAN) bus data and specifically the steering angle (α), the yaw rate (φ) and the wheel speed (υ), the system is able to predict the future location of the vehicle within a timestep δt.

In the update step, an Extended Kalman Filter (EKF) approach is applied to fuse the predicted vehicle's location with a GPS-free global location measurement. In our case, the GPS-free global location measurements are obtained through Signals of Opportunity (SoO), where the global location of the vehicle can be estimated with some uncertainty. More details about the SoO will be discussed in the Section 3.2.2.2. To identify a possible GPS location spoofing attack, the vehicle's location obtained through the fallback solution is compared with the current GPS measurement in the final comparison step (iii), as shown in Figure 3.1.

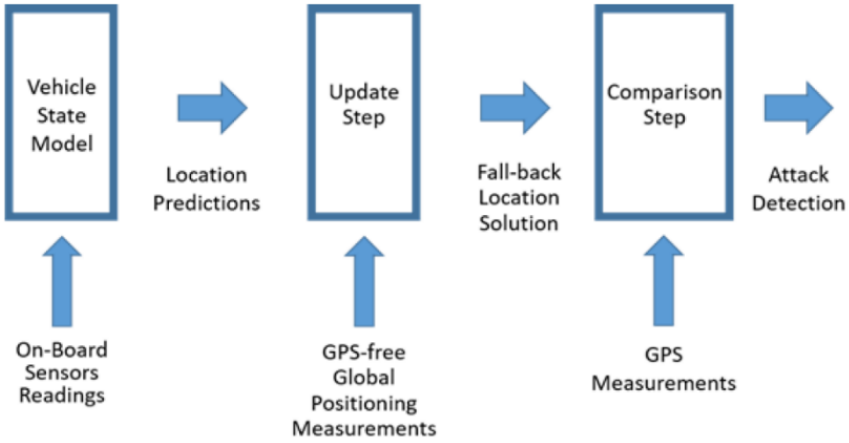


Figure 3.1. In-vehicle GPS location integrity check.

For the prediction step, using the sensor information data, it is possible to build a non-linear model of the vehicle system state following the underlying physical laws. As described in the deliverable D2.1 section 3.4.1, such a non-linear model exploits the assumption that the motion of the vehicle can be well approximated by a bicycle. If the body-frame of the vehicle is considered oriented as the x-axis, the one-step prediction of the location and the speed of the vehicle in its body-frame reference systems is:

$$\begin{pmatrix} x^u_{k+1} \\ \dot{x}^u_{k+1} \\ y^u_{k+1} \\ \dot{y}^u_{k+1} \end{pmatrix} = \begin{cases} \upsilon \Delta t \\ \upsilon \\ \frac{1}{2}\left(C_f\left(\alpha - \frac{l_f \dot{\varphi}}{\upsilon}\right) + C_r \frac{l_r \dot{\varphi}}{\upsilon}\right)\frac{1}{M}\Delta t^2 \\ \left(C_f\left(\alpha - \frac{l_f \dot{\varphi}}{\upsilon}\right) + C_r \frac{l_r \dot{\varphi}}{\upsilon}\right)\frac{1}{M}\Delta t \end{cases}$$

where l_f and l_r represent the distance of the front wheel and the rear wheel from the mass barycenter, respectively, M is the mass of the vehicle, and C_f and C_r represent the corner stiffness of the front and rear wheels, respectively. Applying a simple coordinate transformation, the one-step prediction in the global geographic reference system can be obtained as:

$$\begin{pmatrix} x_{k+1} \\ \dot{x}_{k+1} \\ y_{k+1} \\ \dot{y}_{k+1} \\ \varphi_{k+1} \end{pmatrix} = \begin{cases} x_k + x^u_{k+1}\cos\varphi_k - y^u_{k+1}\sin\varphi_k \\ \dot{x}^u_{k+1}\cos\varphi_k - \dot{y}^u_{k+1}\sin\varphi_k \\ y_k + x^u_{k+1}\sin\varphi_k + y^u_{k+1}\cos\varphi_k \\ \dot{x}^u_{k+1}\sin\varphi_k - y^u_{k+1}\cos\varphi_k \\ \varphi_k + \dot{\varphi}\Delta t \end{cases}$$

Under the assumption of uncorrelated and Gaussian measurement noise, the associated covariance of the estimated vehicle's system state is computed using the EKF approach.

3.2.2.1.1 Comparison Step

The attack detection is performed in the comparison step, as shown in the Figure 3.1. This attack detection mechanism takes as inputs the output of the estimated location, from the fallback solution and the current GPS location. First, we define the deviation of these two locations as the Euclidean distance

of the two vectors $e_d = ||x - \hat{x}||$ where x and $\hat{x}$, are the GPS location and the estimated location respectively. An indication of a potential attack occurs when the distance e_d exceeds a predetermined threshold T_d. The choice of the threshold value, is done by following an empirical approach where, an attack-free time period is assumed with the vehicle moving. At each time-step a set of n locations $\hat{x}_i$ and x_i is collected where $i = 1, \dots, n$ denotes the location samples. Figure 3.2(a) depicts the distance e_d for $n = 30$. The choice of the T_d cannot be arbitrary since there is a performance trade-off regarding the correct and false detection rates in the presence of attacks. For example, by setting a relatively low threshold value, our solution would probably detect the majority of the potential attacks; however, the false alarm rate will be high and as a result, the proposed solution be ineffective. On the other hand, by setting a relatively high threshold value (e.g., $T_d = max(e_d, i),\ i = 1, \dots, n$), the high false alarm rate can be addressed, but at the expense of misdetections that will increase. In our solution, we select the T_d as the a_{th} percentile of e_d distance errors, chosen from the cumulative distribution function of the distance e_d, which is depicted in Figure 3.2(b). Furthermore, a way to decrease the uncertainty on the decision level, the use of a sliding window can be applied. A sliding window of length w can be used for a more robust decision based on the majority vote approach. Therefore, a GPS location attack detection can be performed using a set of detection estimates of a size defined by the sliding window length w.

Alternatively, the comparison step can be performed in a different manner by exploiting the fact that a GPS receiver not only provides an approximated location for the vehicle, but also an uncertainty score that can be transformed into a covariance matrix. Based on that and under the assumption of Gaussian distribution for the GPS measurements, the two locations can be compared using the Bhattacharyya distance. Essentially the Bhattacharyya distance computes the amount of overlap of two statistical distribution, hence, measuring their similarity. Assuming the vehicle's location estimation and the corresponding covariance matrix as $[\mu_x, \mu_y]$ and $\Sigma_{x,y}$ and the GPS location and covariance matrix as $[\mu_x^G, \mu_y^G]$ and $\Sigma_{x,y}^{\mathrm{G}}$, the Bhattacharyya distance for each time slot k can be computed as:

$$D = \sum_{n \in [k-W,k]} \frac{1}{8}\mu[n]\left(\frac{\Sigma_{x,y}[n] + \Sigma_{x,y}^{\mathrm{G}}[n]}{2}\right)^{-1}\mu[n]^T + \frac{1}{2}\ln\left(\frac{\det\frac{\Sigma_{x,y}[n] + \Sigma_{x,y}^{\mathrm{G}}[n]}{2}}{\sqrt{\det\Sigma_{x,y}[n]\,\det\Sigma_{x,y}^{G}[n]}}\right)$$

where $\mu[n] = \left(\mu_x[n], \mu_y[n]\right) - \left(\mu_x^{\mathrm{G}}[n], \mu_y^{\mathrm{G}}[n]\right)$, and the Bhattacharyya distance is averaged over the samples collected inside a sliding window of W seconds. The sliding window mechanism allows reducing the number of false alarm due to spurious GPS measurement errors. Nevertheless, the trade-off between the length of the sliding window and the ability of the described attack detection mechanism to react to a GPS spoofing attack has to be considered when setting W.

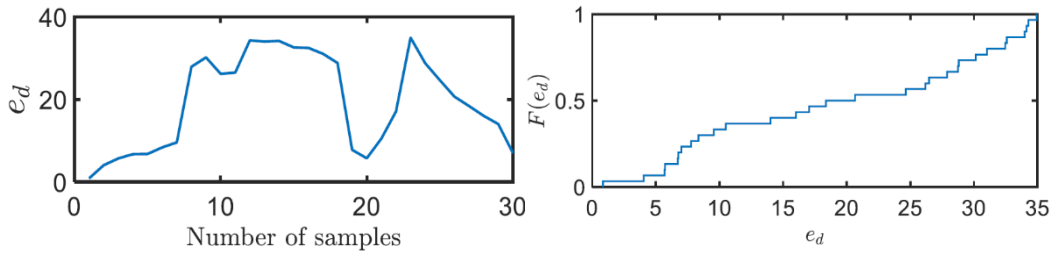


Figure 3.2. a) Deviation between a vehicle's estimated and GPS location. b) The cumulative distribution function of e_d.

3.2.2.2 *Signals of Opportunity*

The current navigation systems are fundamentally relying on GNSS for localization purposes. However, GNSS signals frequently become unavailable in the presence of jamming or spoofing attacks [48][49], pointing to the exploration of SoO as an alternative method for positioning [50]. Specifically, in [51], a novel Relative Positioning System (RPS) is introduced, requiring no information about the transmitters' location or any GNSS signal knowledge, that localizes the moving vehicle accurately and uses the SoO dataset in an EKF solution.

Figure 3.3(a) depicts the components of RPS, where a software-defined radio (SDR) receiver is applied on a moving vehicle gathering SoOs. The assumption is that there are several relative SoO transmitters

in the vehicle's area, with their locations initially unknown. Also, the vehicle's position evolves following a known discretized model, where **F** is defined as the system dynamics and **w** the process noise.

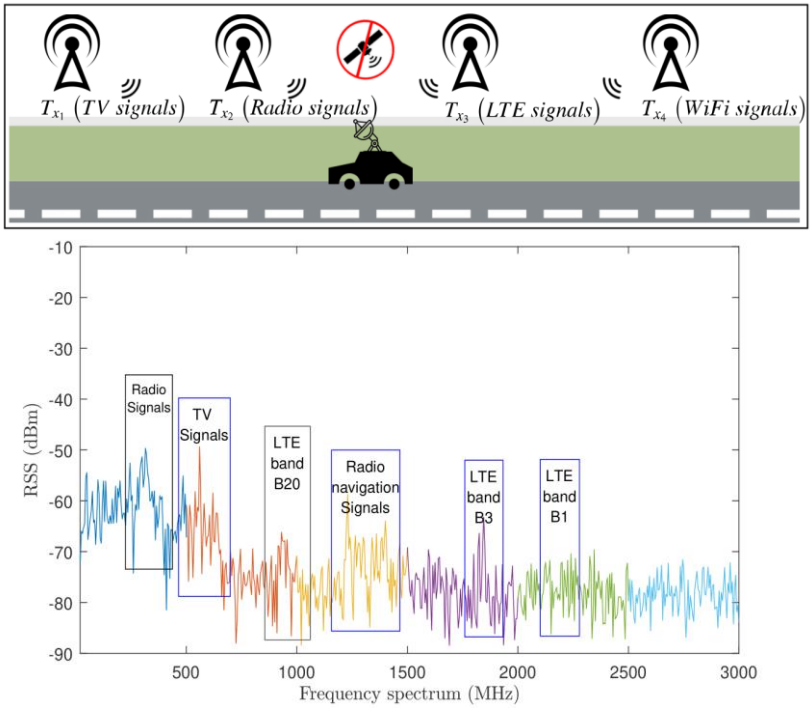


Figure 3.3. (a) System model [51]. (b) Sample of RSS measurements for the 0-3000 MHz frequency spectrum.

Initially, the physical distance between the transmitters sensed in the environment (e.g., radio, TV, Wi-Fi, LTE) and the CAV is computed using a log-normal signal propagation model.

$$P_L = P_{T_{dBm}} - P_{R_{dBm}}$$

$$P_L = P_{L0} - 10n_{PL}\, log_{10}\left(\frac{d}{d_0}\right) + X_G$$

where, $P_{T_{dBm}}$ and $P_{R_{dBm}}$ are the transmit and receive powers in dBm and P_L is the total path loss. P_{L0} is the RSS at a reference distance d_0 and n_{PL} is the path loss exponent. X_G is defined as the log-normal shadowing term and is modelled as a normal Gaussian variable with zero mean.

$$P_{L0} = 20\, log_{10}(d) + 20 log_{10}(f_c) - 27.55$$

The log-distance path loss model is defined as:

$$d = d_0 10 \frac{P_{L0} - P_L}{10n_{PL}}$$

The distance is necessary to estimate the vehicle's position, employing a multilateration method and using the estimated transmitters' location prior to GNSS information loss. Multilateration is a conventional method to calculate a receiver node's unknown location, as discussed in [51][52]. For this

method, it is assumed that each node transmits the information in a circle around itself. The radius of the circle is the distance to the receiver node, and the unknown node's position can be found at the intersection of these circles. As the distances are computed, using the log-normal model, an overdetermined system is created. Subtracting one equation from the others leads to the system linearization, which can be solved by employing the least-squares (LSQ) method.

$$\boldsymbol{A} = 2\begin{bmatrix}(x_1 - x_2) & (y_1 - y_2)\\ \dots & \dots \\ (x_1 - x_n) & (y_1 - y_n)\end{bmatrix},\ \boldsymbol{b} = \begin{bmatrix}x_1^2 - x_2^2 + y_1^2 - y_2^2 + d_2^2 - d_1^2\\ x_1^2 - x_n^2 + y_1^2 - y_n^2 + d_n^2 - d_1^2\end{bmatrix},\quad \boldsymbol{x_k} = \begin{bmatrix}x\\ y\end{bmatrix}$$

Subsequently, the best-approximated solution is computed, employing the following equations:

$$e = \boldsymbol{A}\boldsymbol{x_k} - \boldsymbol{b}$$
$$\boldsymbol{x}^* = argmin_{x_k} e$$
$$\boldsymbol{x}^* = (\boldsymbol{A^T A})^{-1}(\boldsymbol{A^T b})$$

Finally, an EKF is then applied in a multimodal fusion approach to derive the vehicle's location $\hat{\boldsymbol{x}}$ and the measured vehicle system state at time k. At first, the vehicle's position at time $k+1$ is computed, following a vehicle motion model [53]. Afterwards, the predicted position is updated with the estimated SoO-based position to obtain, at each timestamp, the GPS-free vehicle's location $\hat{\boldsymbol{x}}$.

$$\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}} = \boldsymbol{F}\widehat{\boldsymbol{P}}_{\boldsymbol{k}}\boldsymbol{F^T} + \boldsymbol{Q}$$
$$\boldsymbol{K} = \widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}\boldsymbol{H^T}(\boldsymbol{H}\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}\boldsymbol{H^T} + \boldsymbol{R})^{-1}$$
$$\hat{\boldsymbol{x}}_{\boldsymbol{k+1}} = \hat{\boldsymbol{x}}_{\boldsymbol{k}} + \boldsymbol{K}(\boldsymbol{z}_{\boldsymbol{k+1}} - \boldsymbol{H}\boldsymbol{x}_{\boldsymbol{k+1}})$$
$$\widehat{\boldsymbol{P}}_{+} = (\boldsymbol{I} - \boldsymbol{KH})\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}$$

where error covariance estimate is defined as $\widehat{\boldsymbol{P}}_{\boldsymbol{k+1}}$, the Kalman gain is denoted as $\boldsymbol{K}$ and the next error covariance matrix is $\widehat{\boldsymbol{P}}_{+}$. The next state is $\hat{\boldsymbol{x}}_{\boldsymbol{k+1}}$ while $\boldsymbol{z}_{\boldsymbol{k+1}}$ denotes the relative (SoO-based) measurements. $\boldsymbol{H}$ is the measurement matrix, $\boldsymbol{R}$ is the matrix consisting of the variances of the process noise vector, and **Q** is the covariance matrix of the observation noise.

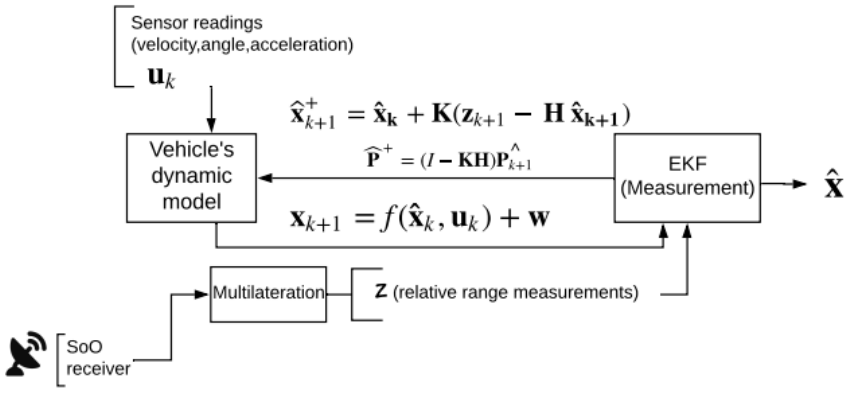


Figure 3.4. Block diagram of the Relative Position System [51].

3.2.3 Validation of Methods

3.2.3.1 *Preliminary results based on drone data*

Experiments utilizing real GPS location data were conducted, for the proposed solution validation. A drone is set to follow a predefined path as shown in Figure 3.5 (red line). The drone equipped with a GPS receiver is able to provide real GPS location measurements along its flight journey. For the fallback solution, one broadband antenna was mounted on an SDR module on the drone to acquire SoOs over a large frequency spectrum (via HackRF-one SDR) [54]. The spoofing attack is simulated by adding Gaussian noise $\boldsymbol{w}$ on the GPS measurements with $\boldsymbol{\mu} = [\boldsymbol{20m}\ \boldsymbol{20m}]$ and $\boldsymbol{\Sigma} = \boldsymbol{\sigma^2 I}$ where $\sigma = 3m$. The first scenario is emulated for the attack to start on the second half of the simulation time (i.e., $n \in [16{,}30]$) whereas, for the second scenario, subsets of GPS locations were attacked (i.e., $n < 6,\ n > 26$).

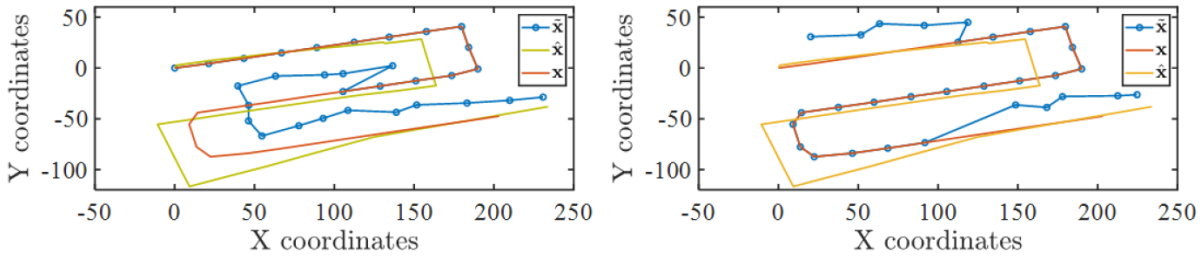


Figure 3.5. Drone trajectory of spoofed (blue), estimated (yellow), and original GPS (red) locations. (a) Scenario with second half of the GPS locations altered and (b) Scenario with a subset of GPS locations (5 at the start and 5 at the end of the trajectory) altered.

The performance assessment is conducted by observing a number of metrics, namely Detection Rate (DR), Misdetection Rate (MR), and False Detection Rate (FDR), defined as:

$$DR = \frac{\#\ of\ detected\ attacks}{\#\ of\ attacked\ locations},\ MR = \frac{\#\ of\ attacks\ missed}{\#\ of\ attacked\ locations},\ FDR = \frac{\#\ of\ falsely\ detected\ attacks}{\#\ of\ non\ attacked\ locations}.$$

Two different values are chosen for the threshold T_d from the CDF function in Figure 3.2(b). One at the 90-th percentile ($T_d = 32m$) and one at the 60-th percentile ($T_d = 22m$), for the first and second scenario respectively.Figure 3.6, shows the performance of this approach. In Figure 3.6(a), the DR reaches up to 90% with $MR = 10\%$ and $FDR = 15\%$ whereas, by setting a lower threshold as shown in Figure 3.6(b), DR tops 100% but in the expense of the increase of FDR up to 45%. Therefore, a proper choice of the threshold T_d must be considered for optimum performance.

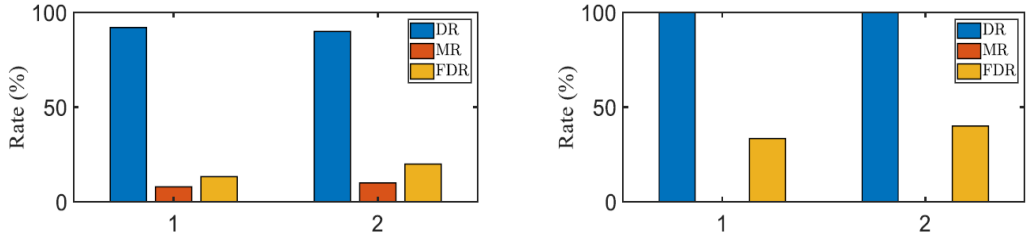


Figure 3.6. Performance evaluation in terms of the DR, MR, and FDR metrics. (a) Using the 90-th percentile of e_d as the detection threshold and (b) Using the 60-th percentile of ed as the detection threshold.

3.2.3.2 *Bhattacharyya Distance in CARLA Simulator*

To assess the performance of the detection algorithm utilizing the Bhattacharyya distance, the described mechanism has been implemented in CARLA simulator [55]. In Figure 3.7(a), one can see the ground truth trajectory of the vehicle as well as the fallback solution estimator where, the SoO measurements are simulated as the ground truth locations with a very large variance, i.e., $N(0, \mathrm{diag}(225\, m^2, 225\, m^2))$ and finally the associated GPS measurements received by the vehicle (distributed as a $N(0, \mathrm{diag}(9\, m^2, 9\, m^2))$) . The malicious attack takes place after the first half of the simulation, by introducing a constant bias of $15m$ on the GPS measurements.Figure 3.7(b), shows the output of the detection scheme. As expected, the instantaneous Bhattacharyya distance presents high variability, which increases the uncertainty of the reliable detection. However, the average Bhattacharyya distance, with a window length of $W = 4s$ provides a smoother output. By choosing the threshold value at the 99-th percentile of the Bhattacharyya distance under an attack-free environment, the proposed solution is able to detect up to 97% of the attacks. A more detailed analysis about the impact of the window and the window length will be discussed in the **Deliverable D5.2**.

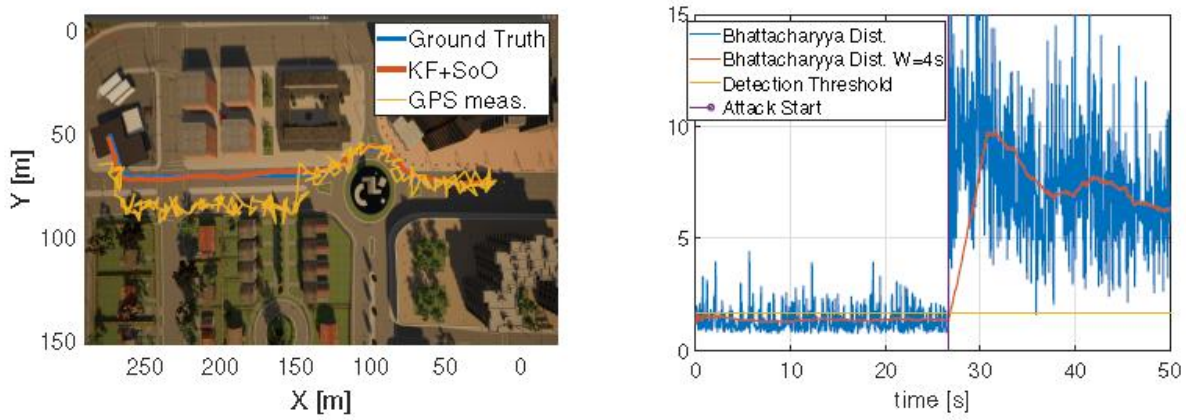


Figure 3.7. Example of a GPS spoofing attack and real-time detection in CARLA simulator.

3.2.3.3 *CARLA Simulator Extensive Results*

In this section, simulation data from CARLA simulator are used to assess the performance of the proposed detection algorithm. Three different trajectories are considered for this analysis which are depicted in Figure 3.8. The performance assessment is based on the output DR, MR, FDR rates for different values of the attack bias in the GPS measurement introduced by the attacker as well as, on the standard deviation (STD) of SoO measurements.

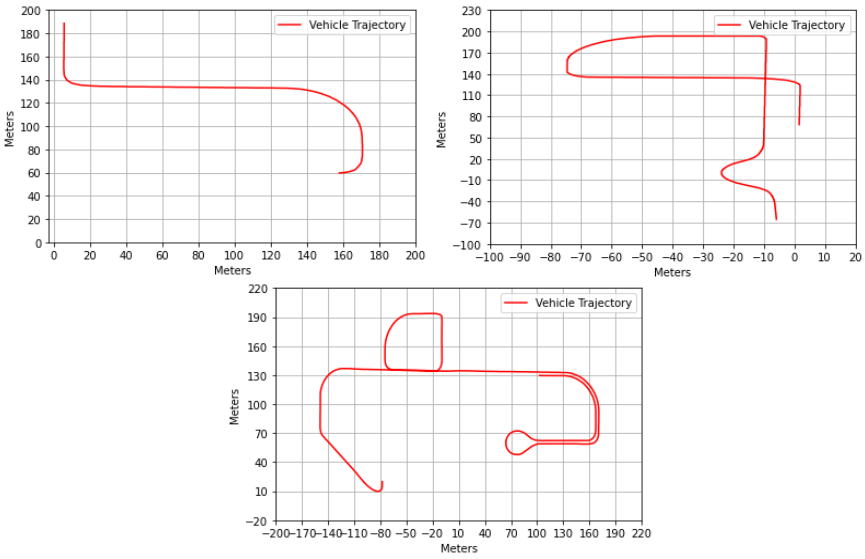


Figure 3.8. Vehicle Trajectories.

Three different values are considered for both STD of SoO and attack bias as shown in Table 3.1, which also includes all relevant parameters of the simulation. The threshold value is set on the 95 percentile of the Euclidean distance based on an attack-free scenario. Results using the sliding window approach are also included in the analysis.

Table 3.1. Simulation Parameters.

Parameter Name	Units	Value
Standard Deviation of Signals of Opportunity	[m]	[10, 15, 20]
Standard Deviation of Signals of Opportunity Orientation	[degrees]	20
Standard Deviation of GPS	[m]	3
Attack Bias in the GPS Measurement	[m]	[5, 9, 12]
Percentile to Select Threshold Value	%	95
Size of Sliding Window	-	5

For the first simulation scenario, the STD of SoO is set to $10m$ and the performance is assessed for all three attack bias values (see Table 3.1). Figure 3.9 shows, the estimated locations using SoO, the fallback solution and the ECDF of the error in the attack-free scenario. The attack is performed on the half simulation time instances in a consecutive way as illustrated in Figure 3.10. The DR, MR and FDR output values are shown in Figure 3.11.

As shown in the results, the overall performance of this method is highly depended on the attack bias values, whereas the algorithm does not perform well for small attack bias. More specific, for GPS attack bias of $5m$, the detection rate of the algorithm drops down to 10%. Furthermore, the impact of the sliding window is clearly shown in Figure 3.11(b), where DR is slightly enhanced. The proposed method can achieve more than 90% of DR for high attack bias with low MD rates less than 10%.

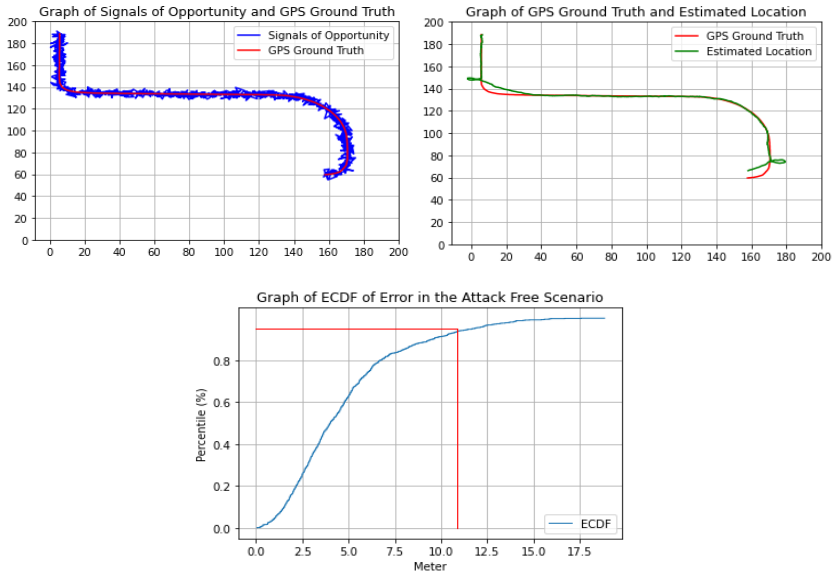


Figure 3.9. (a) SoO location estimation (b) Estimated Location using fall-back solution (c) ECDF of Error in the attack-free Scenario; T_d @ 95%.

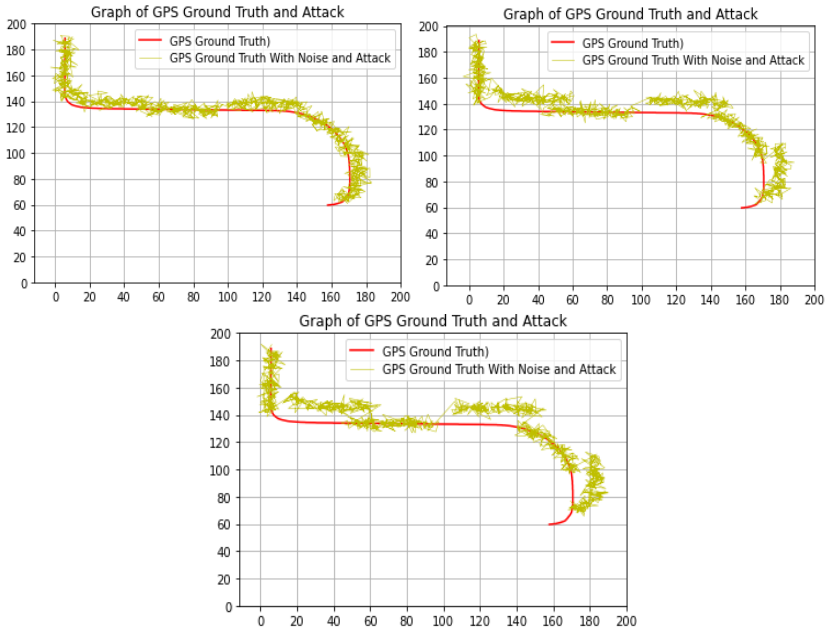


Figure 3.10. Attack scenario (a) Bias of 5m (b) Bias of 9m (c) Bias of 12m.

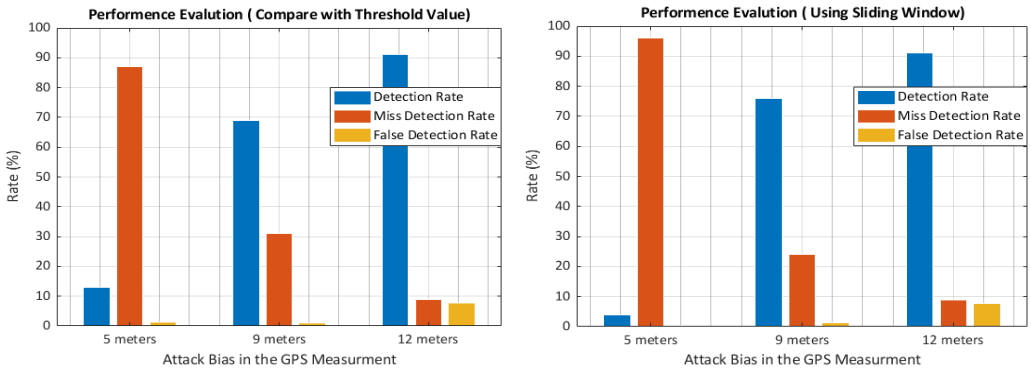


Figure 3.11. Performance Evaluation (a) Without sliding window (b) Using sliding window.

The same simulation procedure is conducted in the second scenario where the STD of SoO is increased to $15m$ and a different vehicle trajectory is chosen. Figure 3.12 shows, the estimated locations using SoO, the fallback solution and the ECDF of the error in the attack-free scenario and the attack is performed on the half simulation time instances in a consecutive way. The DR, MR and FDR output values are shown in Figure 3.13. Similarly, the resulting DR stays low for small attack bias, with maximum performance of 90% of DR for attack bias of $12m$ using sliding window. The increase of the uncertainty in SoO measurements also affected the performance where lower DR achieved for all GPS attack bias values. However, DR can be enhanced by varying the comparison threshold accordingly but with the cost of increasing the FDR.

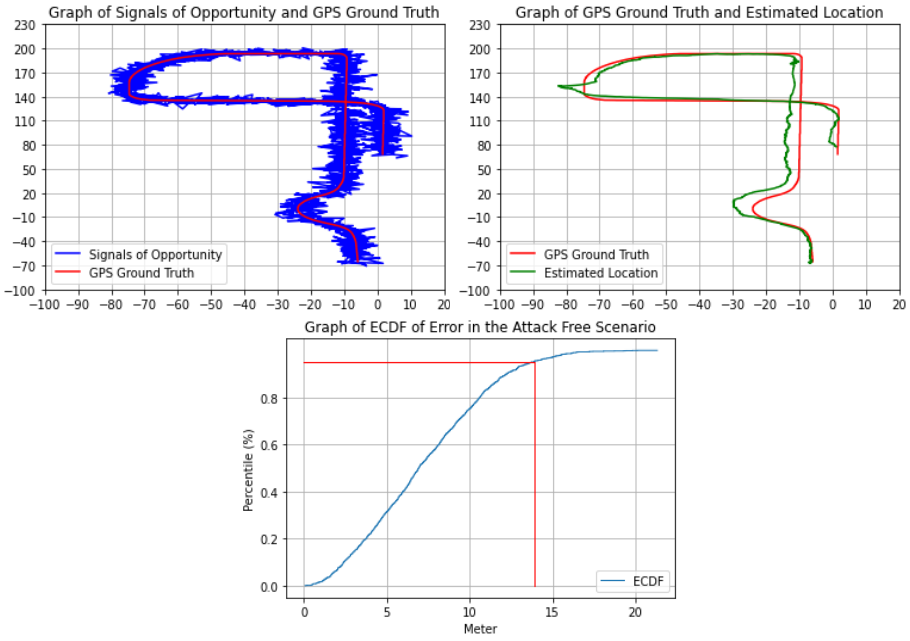


Figure 3.12. (a) SoO location estimation (b) Estimated Location using fall-back solution (c) ECDF of Error in the attack-free Scenario; Td @ 95%.

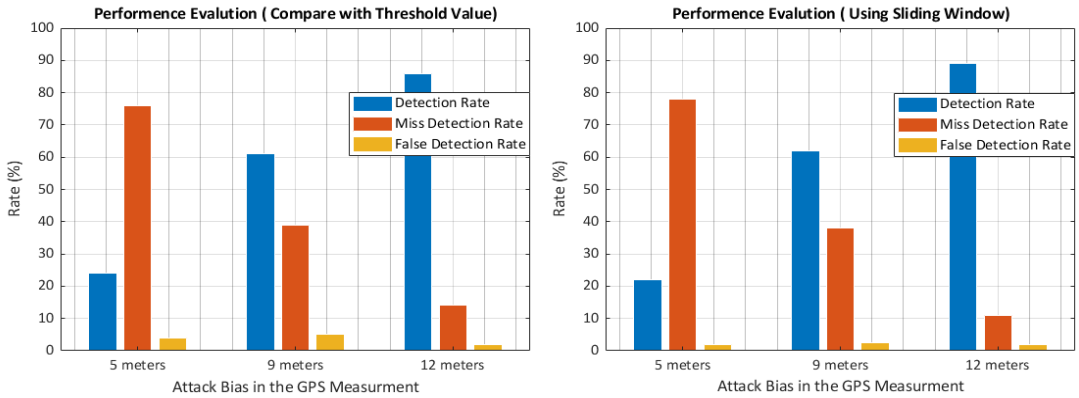


Figure 3.13. Performance Evaluation (a) Without sliding window (b) Using sliding window.

In the final scenario, the impact of the uncertainty of SoO measurements is assessed. Using the final trajectory, the simulation is conducted by fixing the GPS attack bias at $9m$ and the output results are extracted by varying the STD of SoO. The attack scenario is applied in the same manner as in the previous scenarios as shown in Figure 3.14.

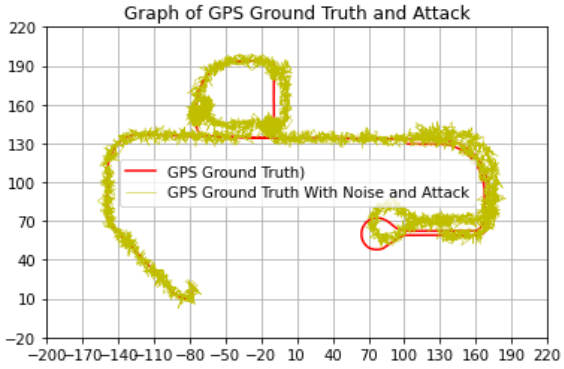


Figure 3.14. Attack scenario, Attack bias 9m.

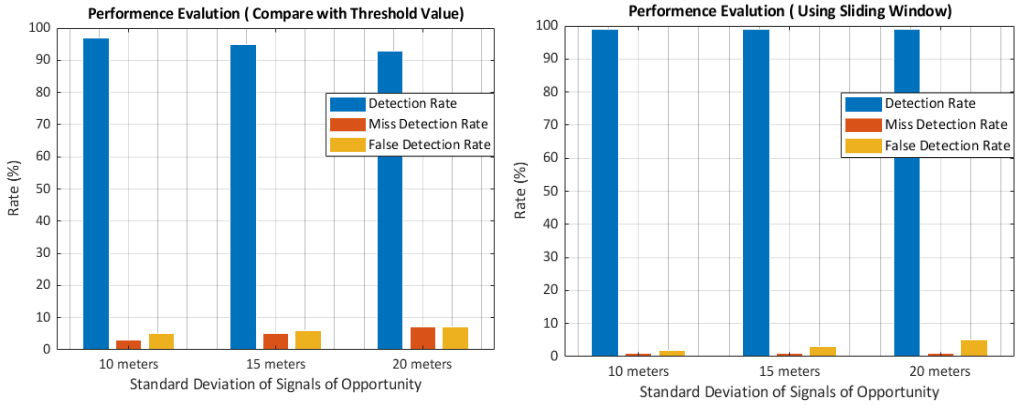


Figure 3.15. Performance Evaluation (a) Without sliding window (b) Using sliding window. Attack bias fixed @ 9m

Figure 3.15 shows the DR, MR and FDR for different STD of SoO. As expected, the increase in the uncertainty of SoO measurements reduces the DR of the algorithm and also increases the FDR. The use of a sliding window slightly improves both DR and FDR achieving DR up to 99% and FDR down to 5%.

3.3 Multi modal fusion between vehicles for detection location spoofing attacks

3.3.1 Sensors and Measurements

As it has been extensively reported in [56], the GPS sensor, originally developed in the USA, is widely known as an example of Global Navigation Satellite System (GNSS). Other types of GNSS include GLONASS (Russia), Galileo (European Union) and BeiDou (China). The group of satellites assigned to each GNSS, is termed constellation. The principal messages transmitted by each system include:

1. Position, velocity and timing (PVT) signal information
2. Ephemeris data, i.e. exact location of operating satellites
3. Almanac, i.e. location and orbit of all constellation's satellites, along with status information.

The interoperability allows GNSS receivers to read signals from the four main satellite constellations and so avoid blackouts in urban canyons or other areas of poor reception by taking into consideration satellites from other constellations that may be visible. All types of GNSS satellites transmit on at least two bands: using the predominant GPS terminology, on frequency L1 an encrypted military code, called P(Y), and an unencrypted civilian code, called C/A, while on the L2 band the P(Y) code is repeated.

At the lowest level a GNSS navigation signal can be seen as an analog sinusoidal wave at a frequency that varies roughly between 1.2 and 1.6 GHz. In order to carry digital information, segments of this basic signal are phase-shifted, in the case of GPS or GLONASS L1 by radians. The usual method is BPSK (binary phase shift keying), which encodes 1 bit per phase-shift. Other variations are QPSK (quad phase), which uses four phase shifts to encode 2 bits per shift, as used in Beidou-2, and MBOC (multiplexed binary offset carrier), as used in Galileo E1, which is designed to interoperate with the existing GPS L1 signal.

The calculations performed by a GNSS receiver to compensate for various forms of signal delay, such as relativistic effects or tropospheric delay, end up as corrections to the difference between the time stamped on the packet and the time it was received, but do not affect the basic nature of the position calculation. When a GNSS receiver reads a time and location signal from a single satellite, it cannot compute the actual range because its local clock will be offset from the satellite clock by an unknown amount. Therefore, it employs the so-called pseudo-ranges p_i for four satellites, representing four non-linear equations with four unknowns (the x, y, z coordinates of the receiver and time offset ΔT). These equations correspond to three hyperboloids, whose intersection, or simultaneous solution, is the location of the receiver.

Finally, it is assumed that range sensors like Camera or LIDAR, which are capable of providing measurements like relative distances and angles between neighbouring vehicles, are utilized for the derivation of our defense mechanism against GPS spoofing.

3.3.2 Methodology Description

The proposed collaborative GPS spoofing defence mechanism relies on multi-modal sensor fusion among the vehicles of a VANET. It is assumed that an attacker compromises and spoofs the GPS of a subset of VANET. However, by means of cooperation and measurements exchange (V2V communication), the vehicles manage to achieve three major tasks: 1) estimate quite accurately their locations, regardless if they were attacked, 2) estimate the possible impact on actual GPS measurement in the form of position outliers, 3) high classification accuracy of vehicles as spoofed or non-spoofed. The two first goals are related to the mitigation stage of our defence mechanism, while at the classification stage actually we detect which vehicles were spoofed. Although the first stage of the mechanism is described on the deliverable **D4.4: Report on the Fallback Actions for Minimal Risk Condition** of *CARAMEL*, we will review it again since it serves as input to the detection phase.

Consider a 2-D region where N connected vehicles collect measurements while moving. An example of such a VANET, is shown in Figure 3.16. The location of the i-th vehicle at k-th time instant is given by $\boldsymbol{x_i^{(k)}} = \left[x_i^{(k)}\ y_i^{(k)}\right]^T$.

Each vehicle knows its absolute position from GPS and measures its relative distances and angles with respect to neighbouring vehicles using LIDAR or Camera. The true relative distance $z_{d,ij}^{(k)}$ between connected vehicles i and j is given by $z_{d,ij}^{(k)} = \|x_i^{(k)} - x_j^{(k)}\|$, where $\|\cdot\|$ is the l^2 norm. The true angle $z_{a,ij}^{(k)}$ between neighbouring vehicles i and j is given by $z_{a,ij}^{(k)} = arctan\frac{y_j^{(k)} - y_i^{(k)}}{x_j^{(k)} - x_i^{(k)}}$.

The acquired measurements are assumed to be described by the following models:

- Relative distance measurement: $\tilde{z}_{d,ij}^{(k)} = z_{d,ij}^{(k)} + w_d^{(k)},\quad w_d^{(k)} \sim N(0, \sigma_d^2)$ (1)
- Relative angle measurement: $\tilde{z}_{a,ij}^{(k)} = z_{a,ij}^{(k)} + w_a^{(k)},\quad w_\alpha^{(k)} \sim N(0, \sigma_a^2)$ (2)
- Relative azimuth angle measurement: $\tilde{z}_{az,ij}^{(k)} = \lambda\pi + arctan\frac{|x_j^{(k)} - x_i^{(k)}|}{|y_j^{(k)} - y_i^{(k)}|} + w_{az}^{(k)},\ \lambda = 0{,}1\ or\ \tilde{z}_{az,ij}^{(k)} = \lambda\pi + arctan\frac{|y_j^{(k)} - y_i^{(k)}|}{|x_j^{(k)} - x_i^{(k)}|} + w_{az}^{(k)},\ \lambda = \frac{1}{2}, \frac{3}{2}, w_{az}^{(k)}\ N(0, \sigma_{az}^2)$ (3)
- Absolute position measurement:

$$\tilde{z}_{p,i}^{(k)} = x_i^{(k)} + w_p^{(k)},\quad w_p^{(k)} \sim N\left(0, \Sigma_p\right) \quad (4)$$

Covariance matrix Σ_p is a diagonal matrix equal to *diag* (σ_x^2, σ_y^2).

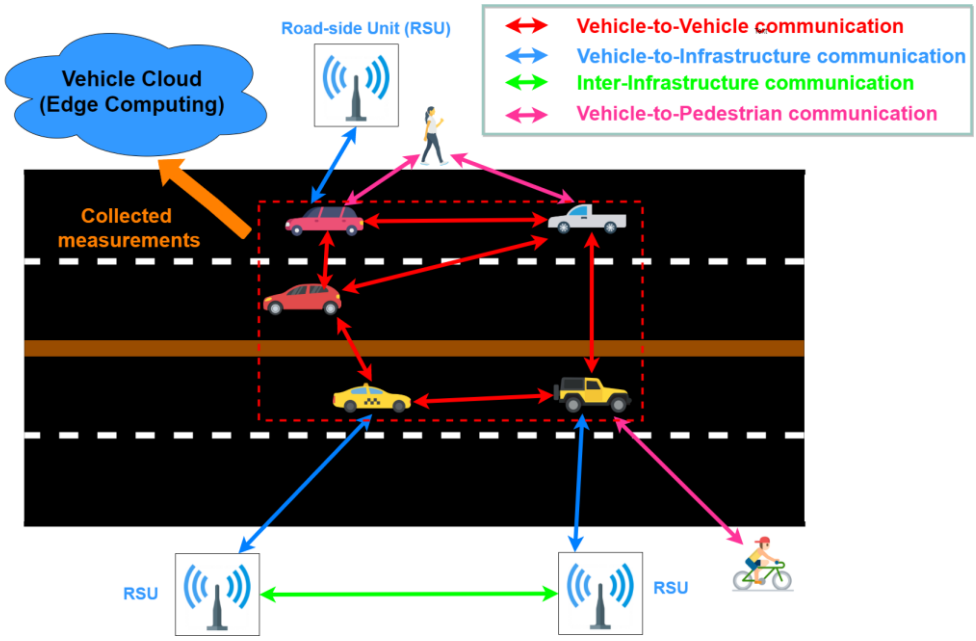


Figure 3.16. VANET

A typical approach in the area of CL is to formulate an objective cost function $C(x)$ according to Maximum Likelihood Estimation (MLE) criterion [57],[58] and to minimize it with respect to locations x_i in order to reduce the error of absolute position measurement. The likelihood function of the measurement models can be written as:

$$L(x) = \prod_{i\in N, j\in N(i)} P\left(x_i^{(k)}, x_j^{(k)}\right) \prod_{i\in N, j\in N(i)} P\left(x_i^{(k)}, x_j^{(k)}\right) \prod_{i\in N} P\left(x_i^{(k)}\right) \quad (5)$$

, where $N(i)$ denotes the set of neighbours of the i-th vehicle and $P(\cdot)$ are the probability density functions of the measurement models. If we take the logarithm of (5), then the objective cost function (same as in [58] and similar to that of [57]) is given by:

$$C(x^k) = \sum_{i\in N, j\in N(i)} \left(\tilde{z}_{d,ij}^{(k)} - z_{d,ij}^{(k)}\right)^2 / 2\sigma_d^2 \quad + \sum_{i\in N, j\in N(i)} \left(\tilde{z}_{a,ij}^{(k)} - z_{a,ij}^{(k)}\right)^2 / 2\sigma_a^2 \quad +$$
$$\sum_{i\in N} \frac{1}{2} \left[\left(\tilde{z}_{p,i}^{\,x,(k)} - x_i^{(k)}\right)^2 / \sigma_x^2 + \left(\tilde{z}_{p,i}^{\,y,(k)} - y_i^{(k)}\right)^2 / \sigma_y^2 \right] \quad (6)$$

The GPS spoofing attack impacts on the absolute position measurement that is provided to vehicles. It may result in dozens, hundreds or even thousands of meters away from the true location. Let $O_i^{(k)} = \left[o_i^{x,(k)}\ o_i^{y,(k)}\right] \in R^{N\times 2}$ be the unknown matrix of outliers to the true x and y locations of vehicles, which models the spoofing attack. Our main goal is to retrieve that impact and to substitute it from cooperative locations estimation approach. The spoofed absolute position measurement is now provided by the following model:

- Spoofed absolute position measurement: $\underline{z}_{p,i}^{(k)} = \tilde{z}_{p,i}^{(k)} + O_i^{(k)}$ (7)

The main hypothesis of the robust cooperative localization solutions that will be developed, relies on the fact that only a small number of 20-25% of VANET's vehicles can be compromised. That property facilitates the exploitation of l^1norm minimization approaches [57], since the outliers matrix is actually sparse, because it corresponds to the vehicles being spoofed. The new cost function, based on MLE criterion and the sparsity properties of outliers matrix, can be formulated according to (8):

$$C(x^k) = \sum_{i\in N, j\in N(i)} \left(\tilde{z}_{d,ij}^{(k)} - z_{d,ij}^{(k)}\right)^2 / 2\sigma_d^2 \quad + \sum_{i\in N, j\in N(i)} \left(\tilde{z}_{a,ij}^{(k)} - z_{a,ij}^{(k)}\right)^2 / 2\sigma_a^2 \quad + \sum_{i\in N} \frac{1}{2} \left[\left(\hat{z}_{p,i}^{x,(k)} - o_i^{x,(k)} - x_i^{(k)}\right)^2 / \sigma_x^2 \; + \; \left(\hat{z}_{p,i}^{y,(k)} - o_i^{y,(k)} - y_i^{(k)}\right)^2 / \sigma_y^2 \right] + \lambda_1 \|o^{x,(k)}\|_1 + \lambda_2 \|o^{y,(k)}\|_1 \qquad (8)$$

, where $\|\cdot\|_1$ is the l^1 norm. The interior point methods provided by CVX software can be applied in order to minimize the cost function. We named this approach as Robust Traditional Cooperative Localization based on MLE (RTCL-MLE).

An alternative approach is to treat the VANET as an undirected graph, using the connected vehicles as its vertices and the communication links between them as its edges. The associated Extended Laplacian Matrix $\tilde{L}^{(k)}$ of the VANET graph and the differential coordinate $\delta_i^{(k)} = \left[\delta_i^{x,(k)} \; \delta_i^{y,(k)}\right] \in R^{N\times 2}$ of each vehicle, can be derived according to that graph modelling and the previously discussed measurement models. See [58][59] for more details. The differential coordinates are equal to:

$$\delta_i^{x,(k)} = \frac{1}{d_i^{(k)}} \sum_{j\in N(i)} -\tilde{z}_{d,ij}^{(k)} sin\tilde{z}_{az,ij}^{(k)}$$

$$\delta_i^{y,(k)} = \frac{1}{d_i^{(k)}} \sum_{j\in N(i)} -\tilde{z}_{d,ij}^{(k)} cos\tilde{z}_{az,ij}^{(k)}$$

, where $d_i^{(k)}$ is the number of connected neighbors to i-th vehicle. Afterwards, the two following vectors are formed:

$$b^{x,(k)} = \left[\delta^{x,(k)} \; \tilde{z}_p^{x,(k)}\right]^T \in R^{2N}$$

$$b^{y,(k)} = \left[\delta^{y,(k)} \; \tilde{z}_p^{y,(k)}\right]^T \in R^{2N}$$

assuming that the noisy GPS positions of the vehicles of the network act as the anchors. Thus, the two following minimization problems have been formulated, based on the graph representation of VANET and the sparsity properties of outliers vectors, in order to estimate the locations of N vehicles, while at the same time to detect and mitigate possible attacks on GPS measurements:

$$\underset{x^{(k)}, o^{x,(k)}}{\operatorname{argmin}} \left\| \tilde{L}x^{(k)} - \left(b^{x,(k)} - q^{x,(k)}\right)\right\|^2 + \lambda_3 \left\|o^{x,(k)}\right\|_1$$

$$\underset{y^{(k)}, o^{y,(k)}}{\operatorname{argmin}} \left\| \tilde{L}y^{(k)} - \left(b^{y,(k)} - q^{y,(k)}\right)\right\|^2 + \lambda_4 \left\|o^{y,(k)}\right\|_1$$

Once again, the interior point methods provided by CVX software can be applied in order to solve the two minimization problems. Note that vectors $q^{x,(k)}$ and $q^{y,(k)}$ are equal to:

$$q^{x,(k)} = \left[0 \; o^{x,(k)}\right]^T \in R^{2N}$$

$$q^{y,(k)} = \left[0 \; o^{y,(k)}\right]^T \in R^{2N}$$

, where zero vector $0 \in R^N$. The outliers of the position must be removed only from the anchors part of vectors $b^{x,(k)}, b^{y,(k)}$. We named this approach as Robust Graph based Cooperative Localization (RGCL). Note that in both cooperative robust methods, regularizing parameters $\lambda_{1,2,3,4} > 0$ control the minimization of location estimation term and the outliers estimation term.

During the detection phase of either of the two robust schemes, a vector containing the Euclidean distances between the initial GPS locations and the estimated locations is formed. Afterwards, a small threshold equal to 10 is set, implying that distances bellow 10m do not correspond to attacked vehicles, while distances greater than 10m may be indicative of an attack. In the latter case, k-means clustering

algorithm, with k = 2, is applied on the corresponding distances, producing two clusters with associated centres. The cluster with the largest centre contains in fact, the distances that correspond to attacks. As such, the ids of spoofed vehicles can be identified.

Apparently, the collaboration towards multi-modal fusion among the vehicles of VANET can lead to estimating their locations more accurately than GPS, as well as defending against GPS spoofing attack. More specifically, exploiting the sparsity properties of outliers matrix, our defence mechanism mitigates the impact of spoofing and detects the compromised vehicles.

3.3.3 Validation of Methods

We have validated the collaborating and robust to GPS spoofing attack approaches in a simulated environment. A number of vehicles, say N = 20, constitute the VANET. For a reduced computational load, two vehicles communicate if and only if their distance is up to 20 m and the maximum number of connected neighbours is 6. We have chosen $\sigma_x = 3\,m$, $\sigma_y = 2.5\,m$, $\sigma_d = 1\,m$ and $\sigma_a = \sigma_{az} = 4°$. The true trajectories of the first 3 vehicles moving for 500 time instances are depicted on Figure 3.17. They have been created according to the bicycle kinematic model (KM) of [60]. The spoofing attack is simulated by adding a bias (sampled uniformly in the interval of [5,40]) to randomly chosen vehicles at each time instant, resulting in an average deviation of the true location equal to 34 m. The experiments were conducted for a number of 5%, 10%, 20% and 30% compromised vehicles. Initially, we constructed the Cumulative Distribution Function (CDF) of Localization Mean Square Error (LMSE) of RTCL-MLE, RGCL, spoofed GPS and normal GPS without outliers. In Figure 3.18, the CDFs of LMSE for 1 (5%) and 4 (20%) compromised vehicles are being presented. It is clearly evident that both the robust schemes significantly reduced the error of spoofed GPS. Moreover, RGCL achieves much better performance than RTCL-MLE and even the GPS. Based on that, the reduction of LMSE of RGCL and RTCL-MLE with respect to GPS, was 77% and 56%, respectively, for 5% compromised vehicles. However, for 20% compromised vehicles, LMSE was reduced by 65% with RGCL, but increased by 1.02% with RTCL-MLE. As it was expected, when the number of attacked vehicles increased, the performances have been degraded. The two proposed cooperative approaches achieved significant reduction of spoofed GPS error, by estimating accurately the locations of vehicles. Furthermore, RGCL proved to outperform RTCL-MLE.

During the detection stage, we measured True Positives, False Positives, True Negatives, False Negatives, True Positive Rate and False Positive Rate for the entire simulation horizon (500 time instances). Afterwards, we constructed the Receiver Operating Characteristics (ROC) curves for 5%, 10%, 20% and 30% spoofed vehicles and measured the Area Under Curve (AUC). The ROC curves for the two schemes are depicted on Figure 3.19. In Figure 3.19(a), RGCL and RTCL-MLE both achieved 99% AUC. In Figure 3.19(b), they achieved 95% AUC. In Figure 3.19(c), RGCL achieved 95% AUC, while RTCL-MLE 94% AUC. Finally, in Figure 3.19(c), they achieved 95% and 93%, respectively. Regardless the number of attacked vehicles, the two robust schemes were able to detect the spoofed vehicles, since the classification accuracy was very high, i.e. AUC greater than 90%. We notice also that as the number of compromised vehicles increases, the classification accuracy is slightly reduced. RGCL performs the same or even better than RTCL-MLE. However, due to its much better performance during the mitigation stage, RGCL proves its superiority as a collaborating defence mechanism against GPS spoofing.

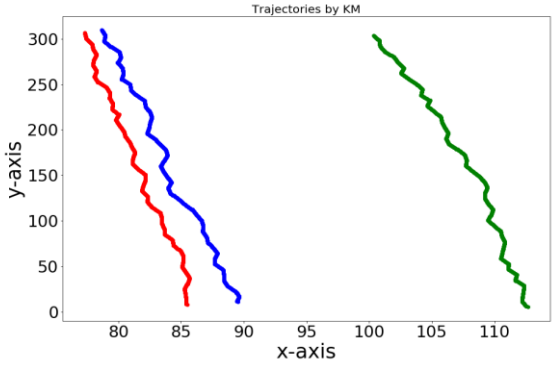


Figure 3.17. True trajectories of three vehicles.

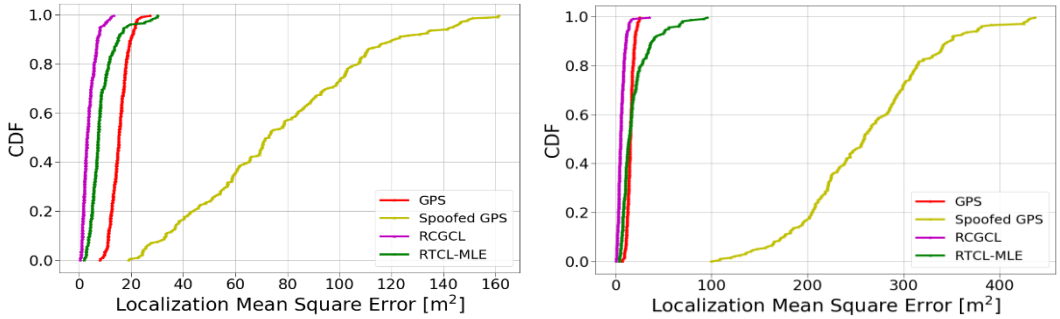


Figure 3.18. CDFs of LMSE of the proposed cooperative and robust schemes for different number of compromised vehicles (a) 1 attacked vehicle, (b) 4 attacked vehicles.

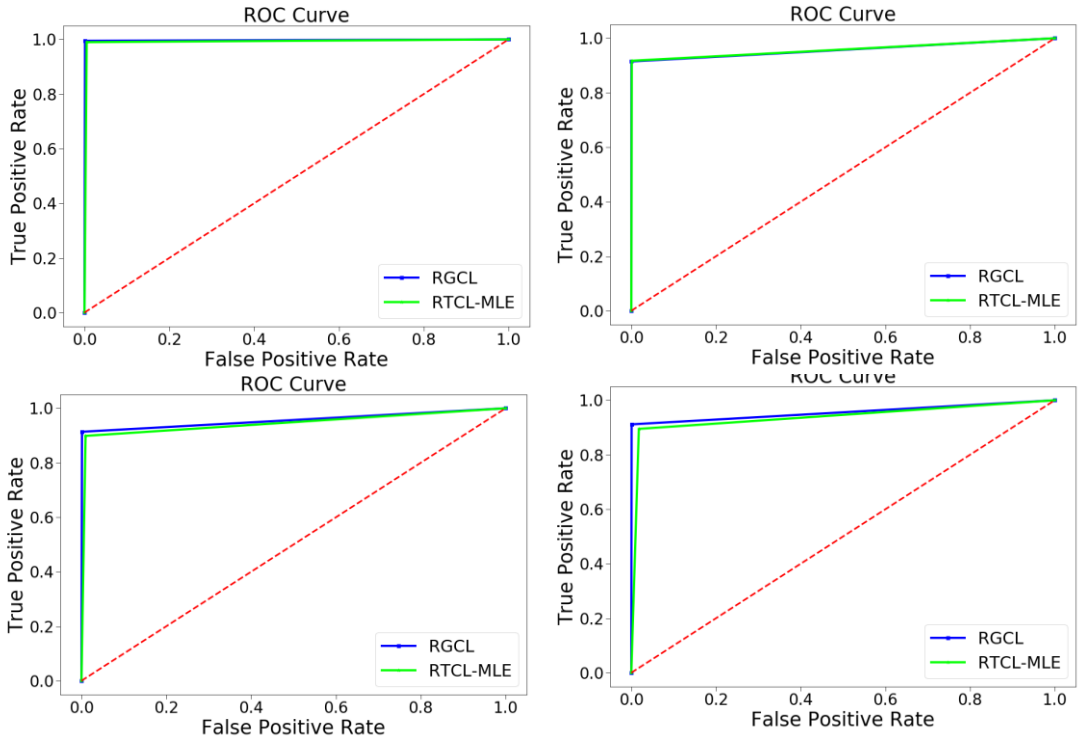


Figure 3.19. ROC curves for different number of compromised vehicles, (a) 1 attacked vehicle, (b) 2 attacked vehicles, (c) 4 attacked vehicles, (d) 6 attacked vehicles.

4 Combination of Standalone Fusion Solutions

This section provides a description of how standalone fusion solutions can be combined in order to increase the overall situational awareness. The standalone fusion solutions that are combined are listed below:

i) Combination of camera attack detection solutions
   a. the *DriveGuard* Convolutional Autoencoder described in D3.2 and
   b. the traffic sign attack detection pipeline described in D4.2

ii) Combination of GPS location spoofing attack detection solutions
   a. Fusing in-vehicle measurements for detecting spoofing attacks described in Section 3.2 and
   b. Multi-modal fusion between vehicles for detection location spoofing attacks described in Section 3.3

4.1 Combination of External and Internal Camera Attack Detection Solutions

4.1.1 Relevance to Attack Scenarios

This section will explore the recent progress in artificial intelligence and deep learning to provide holistic situational awareness for the camera sensor. Specifically, we describe how to formulate a solution that can detect attacks at two different levels thus providing a more robust and holistic situational awareness. It can simultaneously detect attacks that manipulate the camera data itself (internal attack) or external attacks that attempt to fool the camera perception system by manipulating an external structural element such as a traffic sign. The solution is formulated by combining the *DriveGuard* Convolutional Autoencoder described in D3.2 and the detection mechanisms from D4.2 with the traffic sign attack detection pipeline described in D4.2.

4.1.2 Sensors and Measurements

Vision sensors are the most essential sensors in autonomous systems. Sensors such as cameras provide rich information related to the environment and such information can be further analysed to extract useful semantic information such as detection of vehicles [65], pedestrians [66], traffic signs [67] and much more. This analysis is primarily focused on the camera sensors, in particular on the effects of camera sensor attacks and the mitigation strategies on the perception engine. In this instance, traffic sign detection as well as traffic sign anomaly detection models were chosen as part of the perception engine analysis.

Mainly two types of data were captured in this evaluation, camera frames and traffic signs. Both of these data were acquired from a vehicle simulator called Carla [68]. Carla simulator [68] is an open source urban driving simulator and can be used to capture various synthetic data such as camera frames, segmentation information, depth maps and much more.

4.1.3 Methodology Description

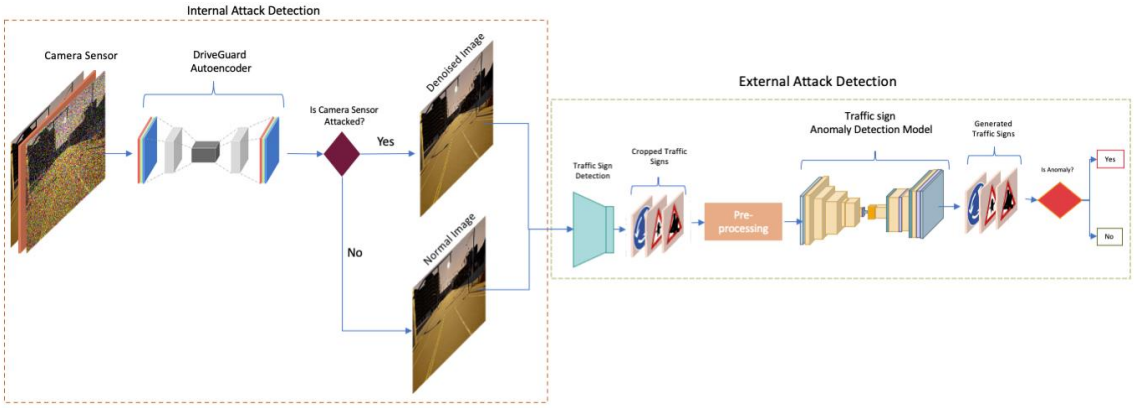


Figure 4.1. Illustration of the fusion of internal and external camera attack detection modules.

As shown in Figure 4.1, the approach follows a hierarchical where first we check for attacks that affect the image frame prior to looking for a more specialized form of attack. The rational being that any form of general attack will render the whole frame unreliable and thus make it more difficult to identify potential attacks on structural elements.

Thus, the approach is first initiated by passing the input image (attacked or otherwise) through the *DriveGuard* autoencoder model that attempts to reconstruct the input image in case of an attack. The detection process of whether the input image is attacked, is done parallel to our model's mitigation process and employs fully connected layers stacked after the generation layer to provide a prediction as a second output. This fully connected branch has been trained by first being provided with pairs of reconstructed/attacked images and reconstructed/normal images. This branch outputs a probability score of whether the image was attacked, which when compared to a threshold provides a detection mechanism. Values closer to 1 demonstrate higher attack likelihood, where values closer to 0 indicate the opposite. In the case of an attack the detector should be triggered and the output of the should be the mitigated generated image and in the opposite scenario the input Image. The input or generated image is then propagated to the next stage to check whether any environmental structure has been tampered with (e.g., traffic sign).

4.1.4 Validation of Methods

I. Validation of internal attack detection

First for the validation of the internal attack detection we gather images from the CARLA simulator [68] by navigating a path with a traffic sign present. These images were then used to construct a dataset of attacked images which are considered for the evaluation. Different types of attacks are considered simulated such as more traditional ones, as gaussian noise, or by adding artefacts addition similar to the ones presented in deliverable D3.2. An example of the visual results is shown in Figure 4.2. Overall, the framework manages to have an overall accuracy of 96% and the detection statistics are shown in **Table** 4.1.

Table 4.1. Overall Detection Statistics

True Positive Rate	0.9285714285714286,	False Positive Rate	0
True Negative Rate	1	False Negative Rate	0.071428571

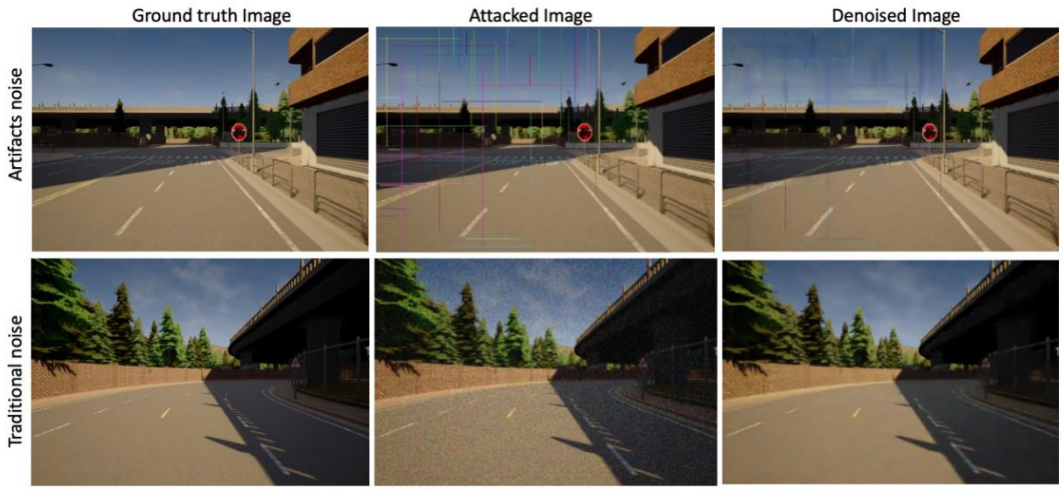


Figure 4.2. First column - Examples of ground truth images with attacked traffic signs, Second columns - images with global attacks (i.e. camera sensor level attack) and third columns - reconstructed images.

II. Validation of external attack detection

For the validation purposes, two experiments were established mainly focusing on the external attacks. First is the validation of the traffic sign detection model on attacked and reconstructed images. Second, is the validation of the traffic sign anomaly detection for denoised images. The core idea is to demonstrate the effectiveness of the denoising process images and the improvement on the performance of algorithms used in perception engines, such as traffic sign detection model and anomaly detection model used for the external attacks**.** Table 4.2 and Table 4.3 shows the total samples that were used for evaluation purposes. Total of 100 images were used for traffic sign detection and a total of 82 images were used for traffic sign anomaly detection.

Table 4.2. Total samples generated images for the evaluation of traffic sign detection evaluation purposes.

Types	**Samples**
Ground truth images	25
Traditional attack images	25
Artefacts attack images	25
Denoised images	25
Total	100

Table 4.3. Total samples generated for the evaluation of traffic sign anomaly detection evaluation purposes.

Types	**Samples**
Traffic signs - Attacked x 2 (types of attacks)	56

Traffic signs - Clean	26
Total	82

III. Traffic sign detection

A MobileNet Single-Shot multibox Detection (SSD) v2 [61] was used for the traffic sign detection task. The model was trained with the synthetic traffic signs and the overall performance of the model on the attack free frames is shown in Table 4.4. The mean Average Precision (mAP) was used as the evaluation metrics. The primary characteristic of the metrics is the IoU which determines the overlapping between the ground truth bounding box and the predicated one. In Table 4.4, "Ground truth images" are the images captured from Carla simulator [68]. The "all attacked images" include both the traditional and artefacts attacked ones and the "denoised images" are the images generated by the *DriveGuard* autoencoder model. Figure 4.2, shows all the types of images.

Table 4.4. The overall performance of the MobileNet SSD v2 [69] on both traditional and artefact attacks.

	Evaluation Metrics - Bounding Boxes detection (Precision) (higher is better)		
Type	**mAP at IoU=.50:.05:.95**	**mAP at .50 IoU**	**mAP at .75 IoU**
Ground truth images	0.3346	0.5859	0.3645
All Attacked Images	0.29285	0.56845	0.22795
Denoised Images	0.3025 (3.30% ↑)	0.58295 (2.55% ↑)	0.2619 (14.89% ↑)

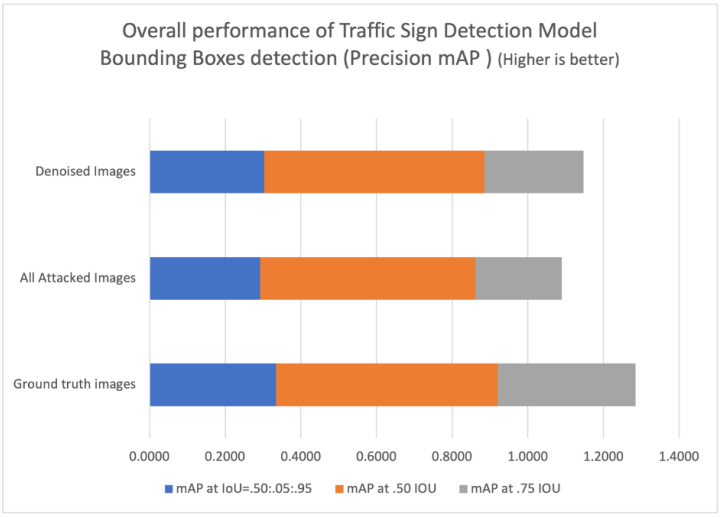


Figure 4.3. The overall performance of the MobileNet SSD v2 [69] on both traditional and artefact attacks.

Table 4.4 and Figure 4.3 shows the overall performance of the traffic sign detection model i.e. MobileNet SSD v2 [5], on ground truth, attacked and denoised images. A total of 100 images used for validation where 25 x ground truth images, 25 x 2 x attacked types images and 25 x denoised images. The performance of the detector degraded when the camera frame is attacked however, due to the applied mitigation strategy (i.e. use of DriveGuard network to denoised the attacked images) the traffic sign detection model performance improves by 2.27%. Likewise, further analysis was performed to evaluate

the effect of individual types of attacks. The Table 4.5, shows the performance of the detector on traditional attack and

Table 4.6, shows the performance on artefacts-based attack. The traditional attack degraded the performance of the detector more than the one with the artefacts.

Table 4.5. The performance of the MobileNet SSD v2 [69] on traditional attacks.

Type	mAP at IoU=.50:.05:.95	mAP at .50 IoU	mAP at .75 IoU
Groundtruth images	0.3346	0.5859	0.3645
Traditional Attack	0.2812	0.5662	0.2167
Denoised Images	0.2960 (5.56% ↑)	0.5843 (3.20% ↑)	0.2277 (5.08% ↑)

Table 4.6. The performance of the MobileNet SSD v2 [69] on artefacts attacks.

Type	mAP at IoU=.50:.05:.95	mAP at .50 IoU	mAP at .75 IoU
Groundtruth images	0.3346	0.5859	0.3645
Artifact attack	0.3045	0.5707	0.2392
Denoised Images	0.3090 (1.48% ↑)	0.5816 (1.91% ↑)	0.2961 (23.79% ↑)

IV. Traffic sign anomaly detection

The traffic sign anomaly detection model was developed for the CARAMEL project and detailed analysis of the model have been discussed in D4.2 report. The anomaly detection model is able to detect the external attack on the traffic sign such as graffiti, noise etc. To evaluate the performance of the anomaly detection on a denoised images after a camera sensor attack, a total of 82 traffic signs were used where 26 x 2 were attacked traffic signs and 26 were normal ones. Figure 4.4 shows the sample of traffic signs used for anomaly detection. The first column is a ground truth traffic image without camera attacks, second column is a denoise image of traditional attack and third is denoise artefact attack. Likewise, the first row of traffic signs does not have external attack whereas others have external such as graffiti and noise attack.

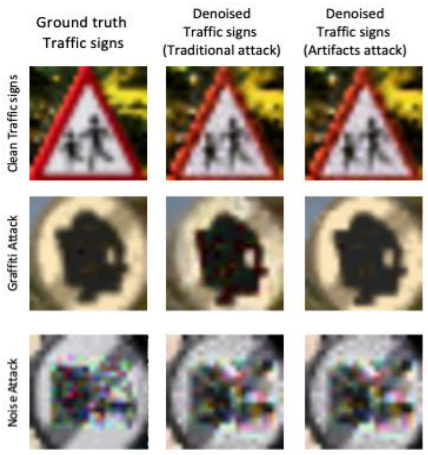


Figure 4.4.Sample of traffic signs used for testing purposes. First column - Ground truth images without any camera sensor attack, second column - Denoised traffic signs from traditional attack and Third column - Denoised signs from artefacts attacks. First row - Traffic signs with no external attack, second row - traffic sign with graffiti attack and third row - traffic sign with noise attack.

Figure 4.5. The performance of the traffic sign anomaly detection on denoised images.

Table 4.7. The performance of the anomaly detection model on clean and denoised images.

Image Type	Ground truth Image (No Camera sensor attack)	Denoise Traffic signs (Traditional Attack)	Denoised Traffic signs (Artefacts attack)
F1	0.8772	0.8761	0.8766
Precision	0.988	0.7978	0.7988
Recall	0.7998	0.9854	0.9867

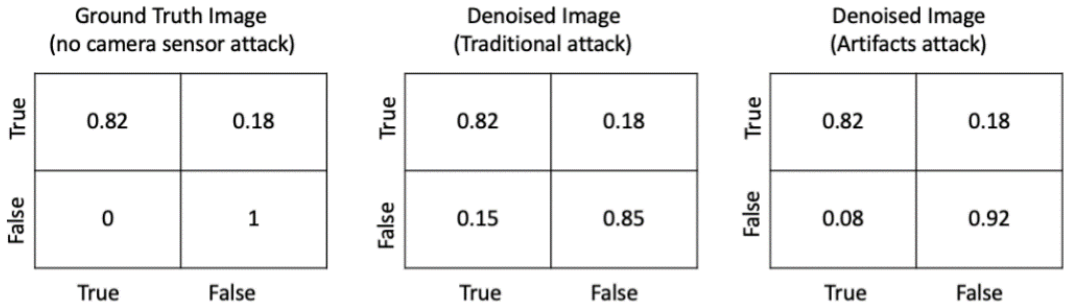


Figure 4.6. Confusion matrix for traffic sign anomaly detection on ground truth image, denoised traditional and denoised artefacts attacks.

Table 4.7 and Figure 4.5 shows the performance of the traffic anomaly detection model. The anomaly detection performed well on both the type of denoised images, however it performed better on denoised images containing artefacts-based attacks than the traditional attacks. From Figure 4.6, it can be seen that the model predicted accuracy predicted true positive (0.82) and true negative (0.92) values in artefacts attacked whereas the model has higher false positive (0.15) in denoise traditional attack.

4.2 Combination of GPS Location Spoofing Attack Detection Solutions

4.2.1 Relevance to Attack Scenarios

This section will provide a GPS location spoofing attack detection solution, combing the existing standalone solutions mentioned previously in Sections 3.2 and 3.3. Specifically, a fusion scheme based on optimal weighting is proposed, for combining the individual location estimates coming from the two proposed solutions, minimizing the overall uncertainty of the final location estimation as well as, providing robustness on the decision level of the attack detection.

4.2.2 Methodology Description

The main aim of this method, is to estimate the current location of the vehicle with minimum uncertainty. As shown in Figure 4.7, the two methods of estimating the current location can be combined in order to obtain an improved estimated state $\hat{X}$. The assumption made here, is that both standalone methods for estimating the current location of the vehicle, can provide the corresponding covariance matrices of the estimation.

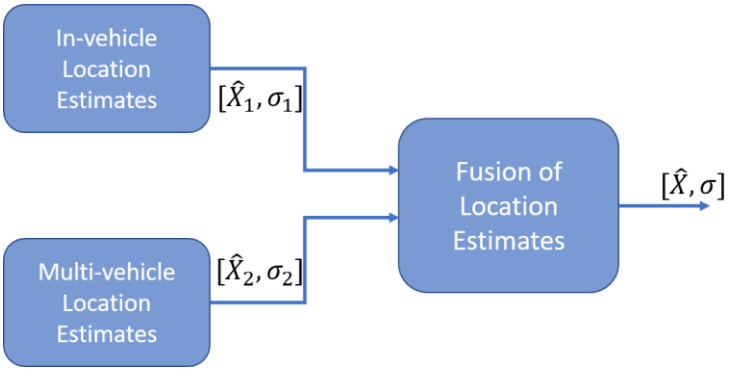


Figure 4.7. Block diagram of the location estimates fusion.

The idea is to use an optimal weighting scheme that minimizes the resulting variance. Let $\widehat{X_a}$, to be the a_{th} estimate of the mean current position of the vehicle and σ_a the covariance matrix associated with that estimate. The index a can take two values $(1\ or\ 2)$ i.e., $a = 1$ corresponds to the In-vehicle location estimates and $a = 2$, for the Multi-vehicle location estimates.

An improved estimate $\hat{X}$ can be obtained by a weight sum of the individual estimates $\widehat{X_a}$ as:

$$\hat{X} = \sum_{a=1;2} W_a \widehat{X_a}$$

The W_a term is a set of $N\ by\ N$ matrices whose matrix elements are to be obtained in a way that minimize the covariance of the improved estimate $\hat{X}$. The equation of the improved estimate $\hat{X}$ can be evaluated as:

$$\hat{X} = \sigma \sum_{a=1;2} \sigma_a^{-1} \widehat{X_a}$$

where,

$$\sigma = \left[\sum_{a=1;2} \sigma_a^{-1} \right]^{-1}$$

The In-vehicle estimates $[\widehat{X_1}, \sigma_1]$, are obtained from the procedure described in Section 3.2 and Figure 3.4 via the EKF where the vehicle's position is computed, following the motion model, and finally the predicted position is updated with the SoO-based estimated position for every time-step.

On the other hand, the collaborating GPS spoofing defense mechanism of Section 3.3.2 produces two estimates (x and y coordinates) for each vehicle of the VANET, removing the impact of spoofing. It is straightforward to employ a simple Kalman Filter for each vehicle, to further improve the location estimation. Thus, at the first step the vehicle is informed about its position from the defense mechanism, and then it utilizes the Kalman Filter. The latter performs the two steps of prediction (according to a kinematic model) and correction of estimation $\widehat{X_2}$ and its covariance σ_2. The correction step exploits in fact the Kalman gain matrix and the measurement vector (obeying a linear measurement model), which contains the two estimations produced by the defense. Therefore, the new pair of estimates $[\widehat{X_2}, \sigma_2]$ obtained by the collaborating defense mechanism and the Kalman Filter, can be combined with those generated by the in-vehicle procedure, to further robustify location estimation and spoofing detection.

5 Conclusion

This deliverable presents the details of two sensor fusion solutions for detecting camera attacks as well as respective mitigation mechanisms; the one is based on LiDAR as an auxiliary data source, while the other one is ML-based and uses imagery and GPS data. In addition, two sensor fusion solutions for detecting GPS location spoofing attacks are described that leverage either the rich multi-source in-vehicle sensor data or the exchange of information among neighboring vehicles. All these solutions are validated through extensive experiments using realistic data collected with the CARLA simulator. Furthermore, two methodologies are described for combining standalone fusion solutions, i.e., one that combines the standalone solutions for detecting camera attacks and another one that combines the standalone solutions for detecting GPS location spoofing attacks. These methodologies pave the way towards enhanced situational awareness and protection against these attacks.

Next steps include the implementation of a subset of the solutions described in this deliverable and their integration into the anti-hacking device that will be documented in deliverable D5.3. In addition, the direction of combining complementary solutions to achieve higher detection accuracy and increased robustness to measurement noise is promising and the CARAMEL partners plan to explore it further.

6 References

[1] Joel Janai, Fatma Güney, Aseem Behl and Andreas Geiger (2020), "Computer Vision for Autonomous Vehicles: Problems, Datasets and State of the Art", Foundations and Trends", in Computer Graphics and Vision: Vol. 12: No. 1–3, pp 1-308.

[2] Rosique, Francisca; Navarro, Pedro J.; Fernández, Carlos; Padilla, Antonio. 2019. "A Systematic Review of Perception System and Simulators for Autonomous Vehicles Research" Sensors 19, no. 3: 648. https://doi.org/10.3390/s19030648

[3] J. Petit, B. Stottelaar, M. Feiri, and F. Kargl, "Remote Attacks on Automated Vehicles Sensors: Experiments on Camera and LiDAR," Blackhat.com, pp. 1–13, 2015.

[4] M. H. Eiza, Q. Ni, Driving with sharks: Rethinking connected vehicles with vehicle cybersecurity, IEEE Vehicular Technology Magazine 12 (2) (2017) 45–51. doi:10.1109/MVT.2017. 2669348.

[5] Szegedy, C.; Zaremba, W.; Sutskever, I.; Bruna, J.; Erhan, D.; Goodfellow, I.J.; Fergus, R. Intriguing properties of neural networks.CoRR2014,abs/1312.6199

[6] Goodfellow], I.J.; Shlens, J.; Szegedy, C. Explaining and Harnessing Adversarial Examples.CoRR2015,abs/1412.6572

[7] Dong, Y.; Liao, F.; Pang, T.; Hu, X.; Zhu, J. Discovering Adversarial Examples with Momentum. CoRR 2017,abs/1710.06081.

[8] Wei, X.; Liang, S.; Cao, X.; Zhu, J. Transferable Adversarial Attacks for Image and Video Object Detection. CoRR2018,abs/1811.12641.

[9] Xie, C.; Wang, J.; Zhang, Z.; Zhou, Y.; Xie, L.; Yuille, A.L. Adversarial Examples for Semantic Segmentation and Object Detection. CoRR2017,abs/1703.08603.

[10] Yan, B.; Wang, D.; Lu, H.; Yang, X. Cooling-Shrinking Attack: Blinding the Tracker with Imperceptible Noises, 2020.

[11] Jason Ku, Melissa Mozifian, Jungwook Lee, Ali Harakeh, and Steven Lake Waslander. Joint 3d proposal genera- tion and object detection from view aggregation. CoRR, abs/1712.02294, 2017.

[12] Xiaozhi Chen, Huimin Ma, Ji Wan, Bo Li, and Tian Xia. Multi-view 3d object detection network for autonomous driving. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 1907–1915, 2017.

[13] Bin Yang, Wenjie Luo, and Raquel Urtasun. Pixor: Real-time 3D object detection from point clouds. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 7652–7660, 2018.

[14] Shuran Song and Jianxiong Xiao. Deep sliding shapes for amodal 3D object detection in rgb-d images. In Proceed- ings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 808–816, 2016.

[15] Yin Zhou and Oncel Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. CoRR, abs/1711.06396, 2017.

[16] Charles Rui Zhong Tai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In Advances in Neural Information Processing Systems, pages 5099–5108, 2017.

[17] Charles R Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3D classification and segmentation. Proc. Computer Vision and Pattern Recognition (CVPR), IEEE, 1(2):4, 2017.

[18] S. Shi, X. Wang and H. Li, "PointRCNN: 3D Object Proposal Generation and Detection From Point Cloud," 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Long Beach, CA, USA, 2019, pp. 770-779, doi: 10.1109/CVPR.2019.00086.

[19] Tsung-Yi Lin, Priyal Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. IEEE transactions on pattern analysis and machine intelligence, 2018.

[20] Charles Rui Zhong Tai Qi, Wei Liu, Chenxia Wu, Hao Su, and Leonidas J. Guibas. Frustum point nets for 3d object detec- tion from RGB-D data. CoRR, abs/1711.08488, 2017

[21] Geiger, A., Lenz, P., Stiller, C., & Urtasun, R. (2013). Vision meets robotics: The KITTI dataset. The International Journal of Robotics Research. The International Journal of Robotics Research, October, 1–6.

[22] A. Geiger, F. Moosmann, O. Car, and B. Schuster, "A toolbox for automatic calibration of range and camera sensors using a single shot," in ICRA, 2012.

[23] Zhou Wang, A. C. Bovik, H. R. Sheikh and E. P. Simoncelli, "Image quality assessment: from error visibility to structural similarity," in IEEE Transactions on Image Processing, vol. 13, no. 4, pp. 600-612, April 2004, doi: 10.1109/TIP.2003.819861.

[24] J. Kocić, N. Jovičić, and V. Drndarević, ‘Sensors and Sensor Fusion in Autonomous Vehicles’, in 2018 26th Telecommunications Forum (TELFOR), Nov. 2018, pp. 420–425, doi: 10.1109/TELFOR.2018.8612054.
[25] H. Gao, B. Cheng, J. Wang, K. Li, J. Zhao, and D. Li, ‘Object Classification Using CNN-Based Fusion of Vision and LIDAR in Autonomous Vehicle Environment’, IEEE Transactions on Industrial Informatics, vol. 14, no. 9, pp. 4224–4231, Sep. 2018, doi: 10.1109/TII.2018.2822828.
[26] D. Nguyen, F. Nashashibi, T. Dao, and E. Castelli, ‘Improving poor GPS area localization for intelligent vehicles’, in 2017 IEEE International Conference on Multisensor Fusion and Integration for Intelligent Systems (MFI), Nov. 2017, pp. 417–421, doi: 10.1109/MFI.2017.8170356.
[27] L. Xu, C. Feng, V. R. Kamat, and C. C. Menassa, ‘An Occupancy Grid Mapping enhanced visual SLAM for real-time locating applications in indoor GPS-denied environments’, Automation in Construction, vol. 104, pp. 230–245, Aug. 2019, doi: 10.1016/j.autcon.2019.04.011.
[28] M. P. Muresan and S. Nedevschi, ‘Multimodal sparse LIDAR object tracking in clutter’, in 2018 IEEE 14th International Conference on Intelligent Computer Communication and Processing (ICCP), Sep. 2018, pp. 215–221, doi: 10.1109/ICCP.2018.8516646.
[29] M. P. Ananda, H. Bernstein, K. E. Cunningham, W. A. Feess, and E. G. Stroud, ‘Global Positioning System (GPS) autonomous navigation’, in IEEE Symposium on Position Location and Navigation. A Decade of Excellence in the Navigation Sciences, Mar. 1990, pp. 497–508, doi: 10.1109/PLANS.1990.66220.
[30] P. J. Navarro, C. Fernández, R. Borraz, and D. Alonso, ‘A Machine Learning Approach to Pedestrian Detection for Autonomous Vehicles Using High-Definition 3D Range Data’, Sensors, vol. 17, no. 1, Art. no. 1, Jan. 2017, doi: 10.3390/s17010018.
[31] J. Li, X. Mei, D. Prokhorov, and D. Tao, ‘Deep Neural Network for Structural Prediction and Lane Detection in Traffic Scene’, IEEE Transactions on Neural Networks and Learning Systems, vol. 28, no. 3, pp. 690–703, Mar. 2017, doi: 10.1109/TNNLS.2016.2522428.
[32] Z. Zhu, D. Liang, S. Zhang, X. Huang, B. Li, and S. Hu, ‘Traffic-Sign Detection and Classification in the Wild’, 2016, pp. 2110–2118, Accessed: Jan. 13, 2021. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2016/html/Zhu_Traffic-Sign_Detection_and_CVPR_2016_paper.html.
[33] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, ‘CARLA: An Open Urban Driving Simulator’, arXiv:1711.03938 [cs], Nov. 2017, Accessed: Nov. 16, 2020. [Online]. Available: http://arxiv.org/abs/1711.03938.
[34] S. Dabiri and K. Heaslip, ‘Inferring transportation modes from GPS trajectories using a convolutional neural network’, Transportation Research Part C: Emerging Technologies, vol. 86, pp. 360–371, Jan. 2018, doi: 10.1016/j.trc.2017.11.021.
[35] X. Ma, Z. Dai, Z. He, J. Ma, Y. Wang, and Y. Wang, ‘Learning Traffic as Images: A Deep Convolutional Neural Network for Large-Scale Transportation Network Speed Prediction’, Sensors, vol. 17, no. 4, Art. no. 4, Apr. 2017, doi: 10.3390/s17040818.
[36] C. Zhou and R. C. Paffenroth, ‘Anomaly detection with robust deep autoencoders’, in Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2017, pp. 665–674.
[37] B. Bilgic et al., ‘Fast image reconstruction with L2-regularization’, Journal of Magnetic Resonance Imaging, vol. 40, no. 1, pp. 181–191, 2014, doi: https://doi.org/10.1002/jmri.24365.
[38] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, ‘Dropout: a simple way to prevent neural networks from overfitting’, J. Mach. Learn. Res., vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
[39] K. Ren, Q. Wang, C. Wang, Z. Qin and X. Lin, "The Security of Autonomous Driving: Threats, Defenses, and Future Directions," in Proceedings of the IEEE, vol. 108, no. 2, pp. 357-372, Feb. 2020, doi: 10.1109/JPROC.2019.2948775.
[40] M. L. Psiaki and T. E. Humphreys, "GNSS Spoofing and Detection," in Proceedings of the IEEE, vol. 104, no. 6, pp. 1258-1270, June 2016.
[41] R. T. Ioannides, T. Pany and G. Gibbons, "Known Vulnerabilities of Global Navigation Satellite Systems, Status, and Potential Mitigation Techniques," in Proceedings of the IEEE, vol. 104, no. 6, pp. 1174-1194, June 2016.

[42] Ali Jafarnia-Jahromi, Ali Broumandan, J. Nielsen, and Gérard Lachapelle. (2012). "GPS Vulnerability to Spoofing Threats and a Review of Antispoofing Techniques," International Journal of Navigation and Observation. 2012. 10.1155/2012/127072.
[43] A. Volpe, “Vulnerability assessment of the transportation infrastructure relying on the global positioning system,” National Transportation Systems Center, 2001.
[44] T. E. Humphreys, B. M. Ledvina, M. L. Psiaki, B. W. O’Hanlon, and P. M. Kintner, “Assessing the spoofing threat: Development of a portable GPS civilian spoofer,” in Proc. Radionavigat. Lab. Conf., 2008, pp. 2314–2325.
[45] N. O. Tippenhauer, C. Pöpper, K. B. Rasmussen, and S. Capkun, “On the requirements for successful GPS spoofing attacks,” in Proc. 18th ACM Conf. Comput. Commun. Secur. (CCS), 2011, pp. 75–86.
[46] T. Nighswander, B. Ledvina, J. Diamond, R. Brumley, and D. Brumley, “GPS software attacks,” in Proc. ACM Conf. Comput. Commun. Secur. (CCS), 2012, pp. 450–461.
[47] Gutierrez, P.: Galileo to transmit open service authentication. Inside GNSS (2020)
[48] K. Shamaei and Z.M. Kassas, “LTE receiver design and multipath analysis for navigation in urban environments,” Navigation, 65(4):655–675, 2018.
[49] Z.Z.M. Kassas, et al., “I hear therefore I know where I am: Compensating for GNSS limitations with cellular signals,” IEEE Signal Processing Magazine, 34(5):111–124, 2017.
[50] K. Shamaei, et al., “Exploiting LTE signals for navigation: Theory to implementation,” IEEE Transactions on Wireless Communications, 17(4):2173–2189, 2018.
[51] N. Souli, P. Kolios, and G. Ellinas, “Relative positioning of autonomous systems using signals of opportunity,” IEEE 91st Vehicular Techn. Conf. (VTC Spring), 2020. (under publication http://arxiv.org/abs/2004.07090)
[52] C. Laoudias, et al., “A survey of enabling technologies for network localization, tracking and navigation,” IEEE Communications Surveys & Tutorials, 20(4), pp 3607--3644, 2018.
[53] G. Mao, et al., “Design of an extended Kalman filter for UAV localization,” Information, Decision and Control Conference, pp 224-229, 2007.
[54] N. Souli, P. Kolios, and G. Ellinas, “Relative positioning of autonomous systems using signals of opportunity,” IEEE 91st Vehicular Techn. Conf. (VTC Spring), 2020. (under publication http://arxiv.org/abs/2004.07090)
[55] Dosovitskiy, A., Ros, G., Codevilla, F., Lopez, A., Koltun, V.: CARLA: An open urban driving simulator. In: Conference on Robot Learning, pp. 1{16 (2017)
[56] D. Schmidt, K. Radke, S. Camtepe, E. Foo, and M. Ren, “A survey and analysis of the GNSS spoofing threat and countermeasures,” ACM Comput. Surveys, vol. 48, no. 4, p. 64, May 2016. [Online]. Available: http://doi.acm.org/10.1145/2897166.
[57] H. Kim, S. H. Lee and S. Kim, ”Cooperative localization with distributed ADMM over 5G-based VANETs,” 2018 IEEE Wireless Communications and Networking Conference (WCNC), Barcelona, 2018, pp. 1-5.
[58] H. Wymeersch, J. Lien and M. Z. Win, ”Cooperative Localization in Wireless Networks,” in Proceedings of the IEEE, vol. 97, no. 2, pp. 427-450, Feb. 2009.
[59] V. Kekatos and G. B. Giannakis, "From Sparse Signals to Sparse Residuals for Robust Sensing," in IEEE Transactions on Signal Processing, vol. 59, no. 7, pp. 3355-3368, July 2011.
[60] N. Piperigkos, A. S. Lalos and K. Berberidis, "Graph based Cooperative Localization for Connected and Semi-Autonomous Vehicles," 2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD), Pisa, Italy, 14-16 September 2020, pp. 1-6.
[61] N. Piperigkos, A. S. Lalos, K. Berberidis, C. Laoudias and K. Moustakas, "5G Enabled Cooperative Localization of Connected and Semi-Autonomous Vehicles via Sparse Laplacian Processing," 2020 22nd International Conference on Transparent Optical Networks (ICTON), Bari, Italy, 19-23 July 2020, pp. 1-4.
[62] S. Thrun, W. Burgard, and D. Fox, Probabilistic Robotics (Intelligent Robotics and Autonomous Agents). The MIT Press, 2005.
[63] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen, ‘MobileNetV2: Inverted Residuals and Linear Bottlenecks’, 2018, pp. 4510–4520, Accessed: Nov. 17, 2020. [Online]. Available: https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.

[64] J. Kocić, N. Jovičić, and V. Drndarević, 'Sensors and Sensor Fusion in Autonomous Vehicles', in 2018 26th Telecommunications Forum (TELFOR), Nov. 2018, pp. 420–425, doi: 10.1109/TELFOR.2018.8612054.
[65] Seenouvong, N., U. Watchareeruetai, C. Nuthong, K. Khongsomboon, and N. Ohnishi. 'A Computer Vision Based Vehicle Detection and Counting System'. In *2016 8th International Conference on Knowledge and Smart Technology (KST)*, 224–27, 2016. https://doi.org/10.1109/KST.2016.7440510.
[66] Brunetti, Antonio, Domenico Buongiorno, Gianpaolo Francesco Trotta, and Vitoantonio Bevilacqua. 'Computer Vision and Deep Learning Techniques for Pedestrian Detection and Tracking: A Survey'. *Neurocomputing* 300 (26 July 2018): 17–33. https://doi.org/10.1016/j.neucom.2018.01.092.
[67] Zhu, Zhe, Dun Liang, Songhai Zhang, Xiaolei Huang, Baoli Li, and Shimin Hu. 'Traffic-Sign Detection and Classification in the Wild', 2110–18, 2016. https://openaccess.thecvf.com/content_cvpr_2016/html/Zhu_Traffic_Sign_Detection_and_CVPR_2016_paper.html.
[68] Dosovitskiy, Alexey, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. 'CARLA: An Open Urban Driving Simulator'. *ArXiv:1711.03938 [Cs]*, 10 November 2017. http://arxiv.org/abs/1711.03938.
[69] Sandler, Mark, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. 'MobileNetV2: Inverted Residuals and Linear Bottlenecks', 4510–20, 2018. https://openaccess.thecvf.com/content_cvpr_2018/html/Sandler_MobileNetV2_Inverted_Residuals_CVPR_2018_paper.html.