D 6.5 – FINAL EVALUATION AND ASSESSMENT OF CPSOSAWARE PLATFORM

Authors CRF, PASSEU, Robotec, I2CAT, ATOS,ISI, UPAT, 8Bells,CTL

Work Package WP6 – Industry Driven Trial and Evaluation

Abstract

This report contains the output of Task 6.5 which lays the ground for the evaluation trials. It consists of designing and planning the way the pilot studies will be organized, supported, and managed throughout the duration of the project. The deliverable is organized around two use case groups. First one Human-Robot Interaction in manufacturing environment includes following use cases: a design operation continuum evaluation and resilience and safety. Second group of connected and autonomous L3-L4 vehicles use cases consists of: human in the loop control, cybersecurity issues and cooperative awareness.

Funded by the Horizon 2020 Framework Programme
of the European Union

Deliverable Information

Work Package	WP6 Industry Driven Trial and Evaluation
Task	T6.3 Deployment of Small Scale Trials
Deliverable title	Final Evaluation and assessment of CPSoSaware Platform
Dissemination Level	Public
Status	Final
Version Number	0.1
Due date	31/12/2022

Project Information

Project start and duration	1.01.2020-31.12.2022
Project Coordinator	Industrial Systems Institute, ATHENA Research and Innovation Center 26504, Rio-Patras, Greece
Partners	1. ATHINA-EREVNITIKO KENTRO KAINOTOMIAS STIS TECHNOLOGIES TIS PLIROFORIAS, TON EPIKOINONION KAI TIS GNOSIS (ISI) - Coordinator 2. FUNDACIO PRIVADA I2CAT, INTERNET I INNOVACIO DIGITAL A CATALUNYA (I2CAT), 3. IBM ISRAEL - SCIENCE AND TECHNOLOGY LTD (IBM ISRAEL 4. ATOS SPAIN SA (ATOS), 5. PANASONIC AUTOMOTIVE SYSTEMS EUROPE GMBH (PASEU) 6. EIGHT BELLS LTD (8BELLS) 7. UNIVERSITA DELLA SVIZZERA ITALIANA (USI), 8. TAMPEREEN KORKEAKOULUSAATIO SR (TAU) 9. UNIVERSITY OF PELOPONNESE (UoP) 10. CATALINK LIMITED (CATALINK) 11. ROBOTEC.AI SPOLKA Z OGRANICZONA ODPOWIEDZIALNOSCIA (RTC) 12. CENTRO RICERCHE FIAT SCPA (CRF) 13. PANEPISTIMIO PATRON (UPAT)
Website	www.cpsosaware.eu

Control Sheet

VERSION	DATE	SUMMARY OF CHANGES	AUTHOR
0.1	09.11.2022	Table of Contents Defined and Shared	*PASEU, All consortium members*
1.0	30.11.2022	Fifty Percent of the Material in the document	*All Consortium Members*
1.1	07.12.2022	Seventy Percent of the Material in the document	*All Consortium Members*
2.0	18.12.2022	One Hundred Percent of the Material in the document	*All Consortium Members*
2.1	29.12.2022	Harmonization of the Material	*PASEU*
2.2	02.01.2023	Deliverable Enters Review Phase	*8Bells, CRF*
2.3	09.01.2023	Review Comments Embodied in the deliverable	*PASEU*
3.0	17.01.2023	Final Version	*ISI, PASEU*

	NAME
Prepared by	PASEU
Reviewed by	CRF. 8BELLS
Authorised by	ISI

DATE	RECIPIENT
-	Project Consortium
18/01/2023	European Commission

Table of contents

List of figures

List of tables

1. Introduction

Scope and objectives of industry driven evaluation trials

The objective of this deliverable is to evaluate the effectiveness of solutions developed in CPSOSAWARE's , using different use cases scenarios as "actions" to represent possible attacks.

The current deliverable contains the results of the evaluation trial tests that were performed against the CPSoSAWARE solution in the context of Task 6.4 '*Small Trials Evaluation and testing*'. This deliverable was delayed due to the on-going at the time integration of the different components of the CPSoSAWARE solution that were planned to be completed in M36 (December/2022), where the final evaluation for the automotive testing was performed on the 13th and 14th of December 2022 in Frankfurt, Hessen, Germany. The corresponding testing for the Industrial Pilot was performed on the 21st and 22nd of November in Orbassano, Italy.

This document outlines the results of the penetration tests that were conducted in every aspect of the CPSOSAWARE solution, trying to identify possible vulnerabilities that would disrupt the functional system that uses the solutions developed throughout the duration of this project. The results from processes carried out in the technical work packages (WP3, WP4, and WP5), resulted in the integration of the CPSoSAWARE solution (as described in D5.4), that was used to perform the evaluation tests.

The document details the results of the penetration testing from the various components of CPSoSAWARE Pillars (Human-Robot Interaction in Manufacturing Environment – Chapter 3, and Connected and Automated Vehicles Chapter 4). The two pillars have followed different methodologies for the deployment and integration of their respective corresponding solutions. The results and observations extracted from the two trials ((a) Orbassano and (b) Frankfurt), were used for the overall assessment of CPSoSAWARE development presented in this document

1.1 Architecture and Hardware Computing Platforms, Developed in Trials

CPSoSAware project has developed an architecture which can be adapted to different scenarios, leveraging on the flexibility that a layered architecture provides. The picture below (Figure 1) depicts a possible configuration for the architecture, to build a solution for the cybersecurity use case, but it could be slightly reconfigured to adapt to other use cases. The picture also showcases the contributions from different partners involved in the cybersecurity use case and the location of the different components at partners

premises.

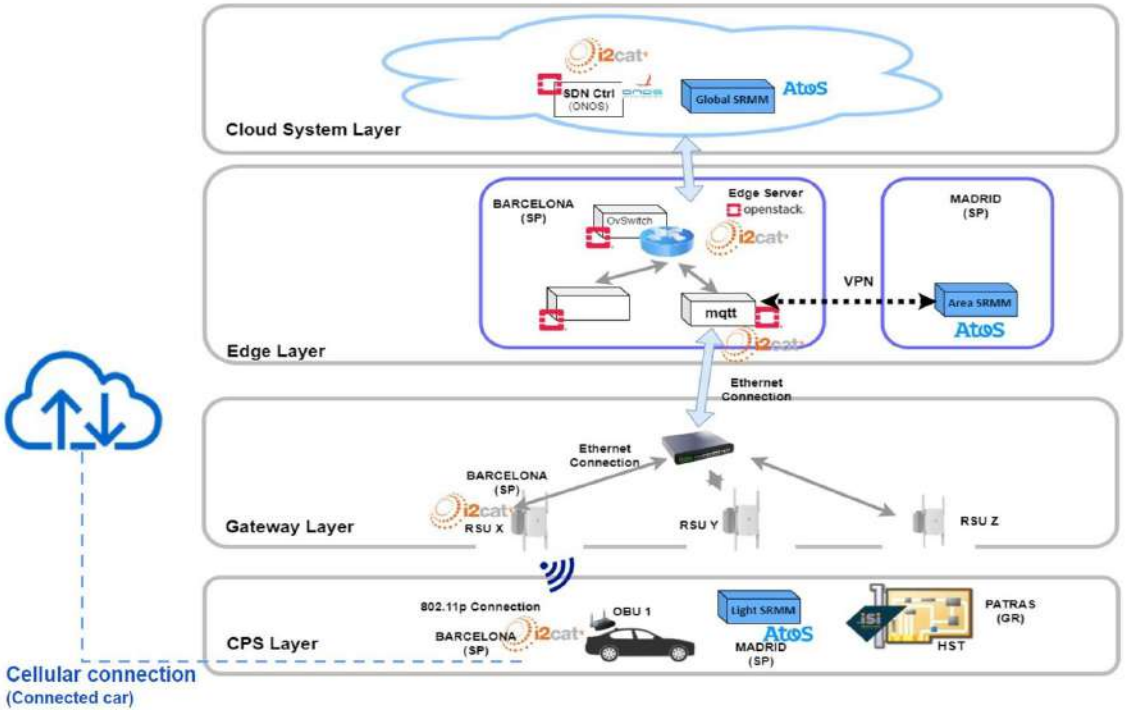


Figure 1: Architecture of the CPSoSAware project

From bottom to top, the schema shows the different architecture layers:

The bottom layer is the CPS Layer, where the cars (containing multiple CPS each) can be found. In our scenario, we assume that we have connected cars, so that's why a cellular connection is represented. But for the major part, the communications between cars, or cars and RSUs are done using V2X communications (ETSI-G5 + IEEE 802.11p). The cellular connection will be only used in the last step, when cars need to connect individually to a server, to download new versions of the firmware.

Next layer up is the Gateway Layer, in which the RSUs (Roadside Unit) are located. These are interconnected via a physical switch and via ethernet. The Gateway Layer connects with the Edge Server, where more computing power is available.

The Edge Layer contains the Edge Node. The project has chosen to manage the resources in the Edge Node as a virtual infrastructure managed by OpenStack, which will guarantee multi-tenancy and tenant isolation. In the Edge Layer, there are several hosts (Virtual Machines) implementing different functionalities as for example an MQTT broker used by the Area SRMM which is also in a different host. In this layer, there is also a host implementing a virtual switch based on, an open-source referent for virtual switches, Openvswitch, which can be controlled by an SDN controller residing in the upper layer, the cloud layer.

Cloud System Layer is the top layer in the architecture, and it is located in the cloud. In the diagram depicting the demo, the Cloud System Layer is a private cloud connected to the internet and reachable from the CPS Layer. In our use case, in the Cloud System Layer we have the SDN controller, and then some other services as the Global SRMM.

2 Evaluation trials

CPSoSaware developed a framework prototype, which was evaluated in T6.1 and T6.2, using the functional components defined and implemented through the technical work-packages of the project. The trial approaches also involved a small sample of human users supporting control and test task, that interact with the machines of the HRC work-cell environment. There were use case scenarios that validated the resilience, reliability and cybersecurity of the functions hosted in CPSoSAWARE ecosystem while we also assess how the HRC design operation continuum is supported. The Human factor was part of the evaluation the scenarios. The goal of this activity was to collect enough data to structure an operational model with enough information for the quantification of the CPSoSaware functions as well as for the risk prediction framework.

3 Human-Robot Interaction in Manufacturing Environment

As described in Deliverable D6.4, the manufacturing scenarios are using different CPSoS technologies mainly distributed between two main use cases:

- Design operation continuum evaluation: In this specific use case, the Human-Robot Collaboration system is considered, together with all the information which are necessary for its execution and that arrive from all the software systems above the field level in the use case (e.g. MES manufacturing execution system or line controllers). In the Design Operation Continuum Evaluation, the system will emulate the need of a change of part numbers. After this event occurs, the system will need to define which modification in the workplace or in the operator's activities will become necessary. The operator will be then invited to work wearing a set of Augmented Reality goggles to be guided in a session of training on the job.
- Resilience and Safety: will consider devices defined in the CPS system which are monitoring and defining set of postures and behaviors that might generate dangerous or non-ergonomic situations to the operator.

All the scenarios converged in a single use-case demonstrator where all the technologies will be integrated. The use-case demonstrator is based on a single application on a pilot work cell where an assembly Human Robot Collaborative cycle is performed.

For both use cases, the implementation will take place in the pilot in CRF premises. The use-case have been set up and run with different participants from different partners that have been testing the technology implemented.

The Use-case from CRF is derived from an in-line work cell in the Mirafiori assembly line (detailed in D1.2 in paragraph "2.3.4 Human Robot Collaboration Pilot").

3.1 Safety Considerations

Being the use-case related to Human Robot Collaboration, stringent safety requirements take place in its set-up (D1.2 in paragraph "2.3.2 Human Robot Collaboration rules according to ISO Standards"). As stated in the previously cited paragraph of tD1.2, the most relevant standards to be considered in the definition of a collaborative work cell are:

- EN ISO 10218-2:2011 [1] that sets the allowed behaviour of the COBOT in Human Robot Collaboration applications, through the definition of the collaborative modes and the rules for the integration in the collaborative workspace (targeted to integrators);
- EN ISO 10218-1:2011 [1] that sets the hardware and functional safety characteristics that a collaborative robot has to fulfil (targeted to robot's constructors);
- ISO/TS 15066:2016 [2] sets the numerical limits for the physical KPIs (velocity, force, power…), and the methodologies for the workplace safety in Human Robot Collaboration (HRC) applications.

While ISO 10218-2 and ISO/TS 15066, define the functional rules for the Safe applications to be certified and consistent, ISO 10218-1 and all its cited standards define the criteria for the hardware to be certified as incorporable in the final certified machine according to the "*DIRECTIVE* 2006/42/EC OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 17 May 2006 on *machinery*, and amending *Directive* 95/16/EC".

Considering that the technologies developed in CPSoSaware can give a support to the safety of the application, but don't have the nature in terms of type of functionalities, nor the certification to be considered Safe machines, (Cat 3/PLd according to ISO 13849-1), the manufacturing use-case have been designed to implement a unique and homogenous system of tools (systems) supporting safety, while not being "safe" themselves.

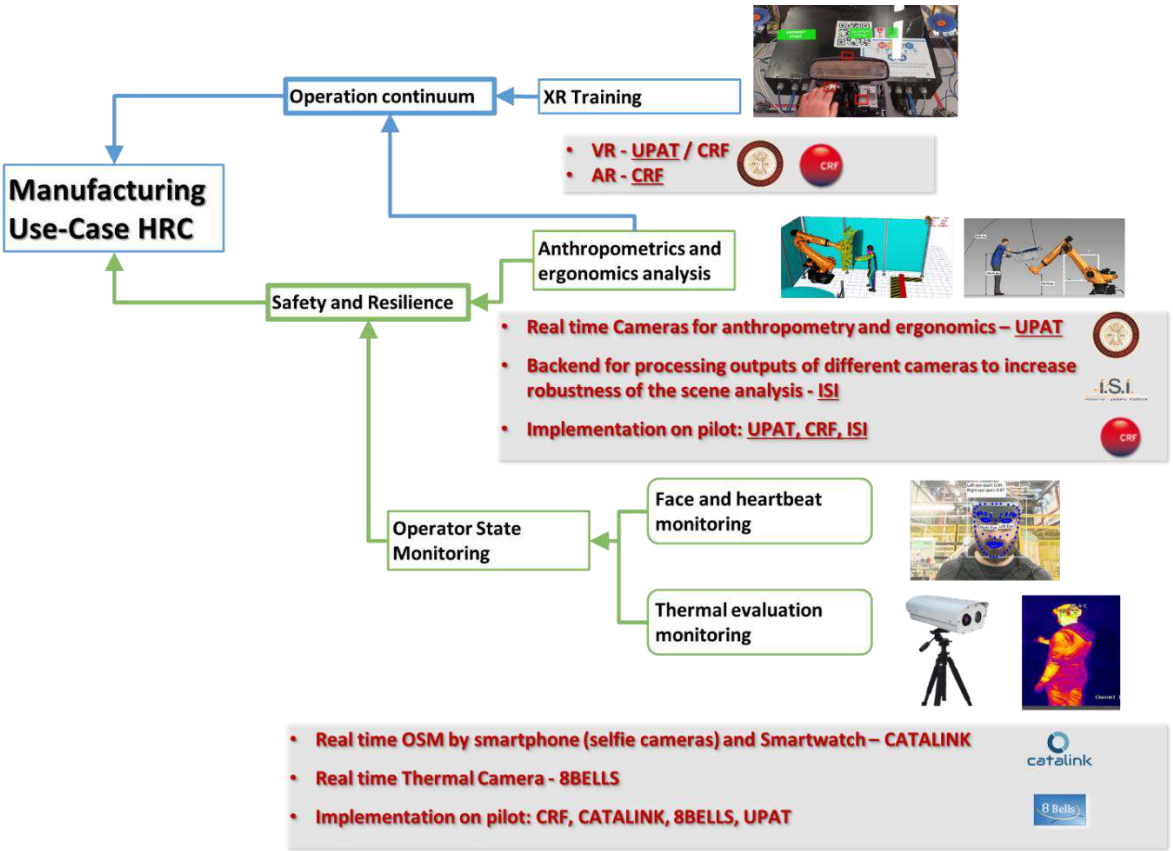


Figure 2: Structure of the Manufacturing demonstrator with the technologies concurring to the definition of its scenarios and use cases. Main involved partners are also indicated.

Figure 2 describes the CPSoSaware technologies concurring in the definition of the manufacturing use-cases (defined in previous deliverable). In the contest of the Pilot, deployed in CRF's site in Orbassano, several technologies are implemented in order to support the achievement of the two use-cases and to increase the *"operation continuum"* and the *"Safety and Resilience"* of the system.

The work cell is indeed designed to be inherently Safe and certified for Safety and uses the information arriving from the CPSoSaware system only to parametrize the height of the gripper (the end-effector acting as manipulator of the windshield and adaptive table for the assembly tasks performed by the operator). In order to achieve the safety certification of the workstation, the CPSoSaware technologies operate only at informational level with respect to the actions performed by the robot without influencing or directly controlling the motion the robot. The work cell structure is made in a way that the application itself is inherently safe also without the information from the CPSoSaware modules or in any case of failure from the CPSoSaware system.

In the standard application implemented on the work cell (described in detail in paragraph 0) a Safe Robot (KUKA KR150), equipped with the KUKA.SafeOperations SW module, picks up the windshield and goes to a collaborative position while all the zones of potential interaction between the robot and the human are monitored form an external 3D Safety monitoring system (The PILZ Safety Eye) which is ensuring that the operator can never enter in contact with the robot while this is moving. The Safety Eye control is obtained by stopping with a Safe Monitored Stop the robot in case of any violation or undesired interaction.

The presence of the Safety Eye that monitors the accesses from the operator, coordinated with the Safe zones definition in the KUKA.SafeOperation system and the proper programming of the access control cabinet makes the application safe for standalone use. As an additional Safety Feature the robot is programmed at a low speed never exceeding 250mm/s.

Indeed, according to the relevant ISO standard the work cell in CRF can be certified for Safety if:

- The Robot is:
  - Safe (PLd – Cat3)
  - Equipped with soft Safety axis
- The operator
  - enters in contact only with the stopped robot (suggested) OR…
  - … the gripper is equipped with a Hand Guiding control *(this feature is not installed nor considered in the CPSoSaware system)*
- The Workplace
  - is monitored against operator's intrusions in the zones where the robot moves

Furthermore, additional safety can be obtained by supporting the operator with visual warnings/HMIs which is the HoloLens AR application developed in CPSoSaware.

As visible in Figure 6, the robot subsystem (a representation of the extract of the subsystem is anticipated in the following Figure 3 for clarity), is completely bases on "standard" safe technologies and it is an isolated system. Its programming is based on standard robotic language and has only one point of communication to the external world (the black arrow below). The interaction with the CPSoSaware systems takes place only through the channel represented by the Black arrow, and, as stated, it is only at informative level. This

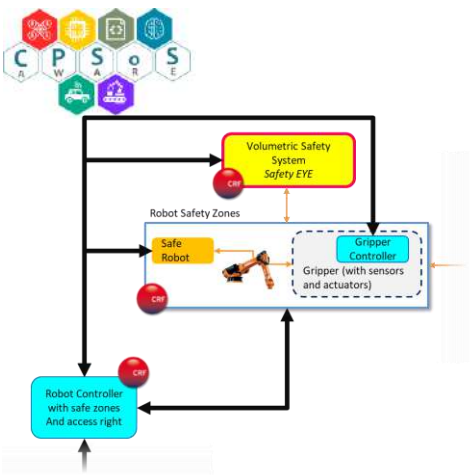


Figure 3: Robot subsystem in the CPSoSaware manufacturing use case (detail from Figure 6)

isolation is such to consider the "Robot" as a system in the overall System of Systems in the workcell. With this robot configuration the safety of the system is automatically achievable. The CPSoSaware subsystem don't directly influence the motion of the robot in any unpredictable way and thus they don't generate risks.

Among the CPSoSaware systems installed, only the Anthropometric analysis module by UPAT can influence the motion of the robot, but its influence is not affecting the Safety.

The Anthropometric analysis module (detailed in paragraph 3.7.1) performs a recognition and classification of the operator's height and publishes in the ROS system a code reporting the to which range of heights the operator belongs (5 classes are identified). According to the specific height class, the robot positions the gripper at a different height from the ground to improve the ergonomics of the operator. During the tasks execution (*see paragraph 0 in Figure 5*), the robot picks up the windshield and makes a rotation inside in the Safe Area iii, which is protected against intrusions from the operator by the Safety Eye. In this moment (end of phase *B.a. in Figure 5*) the robot reads the code related to the height class of the operator, and in the following phases the robot elongates completely toward the collaborative working position (*phase C-D of Figure 5*). The path that the robot follows in its movement from the grouped position to the elongated one is a predetermined path (one path per height class) which is bringing the gripper to a position in space which changes only in the height from the ground. During all the movements, the operator is not allowed to enter in the robot moving zone. When the operator is enabled to access the robot is stationary in a Safe Stop position. In case a non-valid height class, or no height class are provided when the robot reads the required value, the average value (class 3) is used to ensure a statistically reasonable height from the ground without interruptions in the robot's movements. We need to consider that in case no external height detection system is provided the height of the gripper would be fixed at the average position (corresponding to 50 percentile of the operators).

Upon these considerations the overall safety from the robot's motion cannot be hindered by the CPSoS system. On the other end the Augmented reality system developed has the possibility to act as and additional visual warnings/HMI integrating several functions as described in the following paragraphs. The AR system also performs on-demand training sessions enabling a further level of safety to the operator.

3.2 Actors overview

Figure 4.A represent the basic, standard ground layout of the robot with the designed Safety zones. Aleso the Safety Eye (which is placed at 7m from the ground is designed being a part of the standard robotic workcell; Figure 4.B represents the main subsystems of the CPSoSaware system and their indicative location in the CRF's pilot workcell. As visible the operator can be equipped with a smartwatch and the HoloLens AR glasses. The robot is place in the fenced area of the laboratory, but the Safety is such that the operations can be performed with open fences bacause of the Safety Eye monitoring. In the drawing the rough position of the windshield carrier (passive element) is represented.

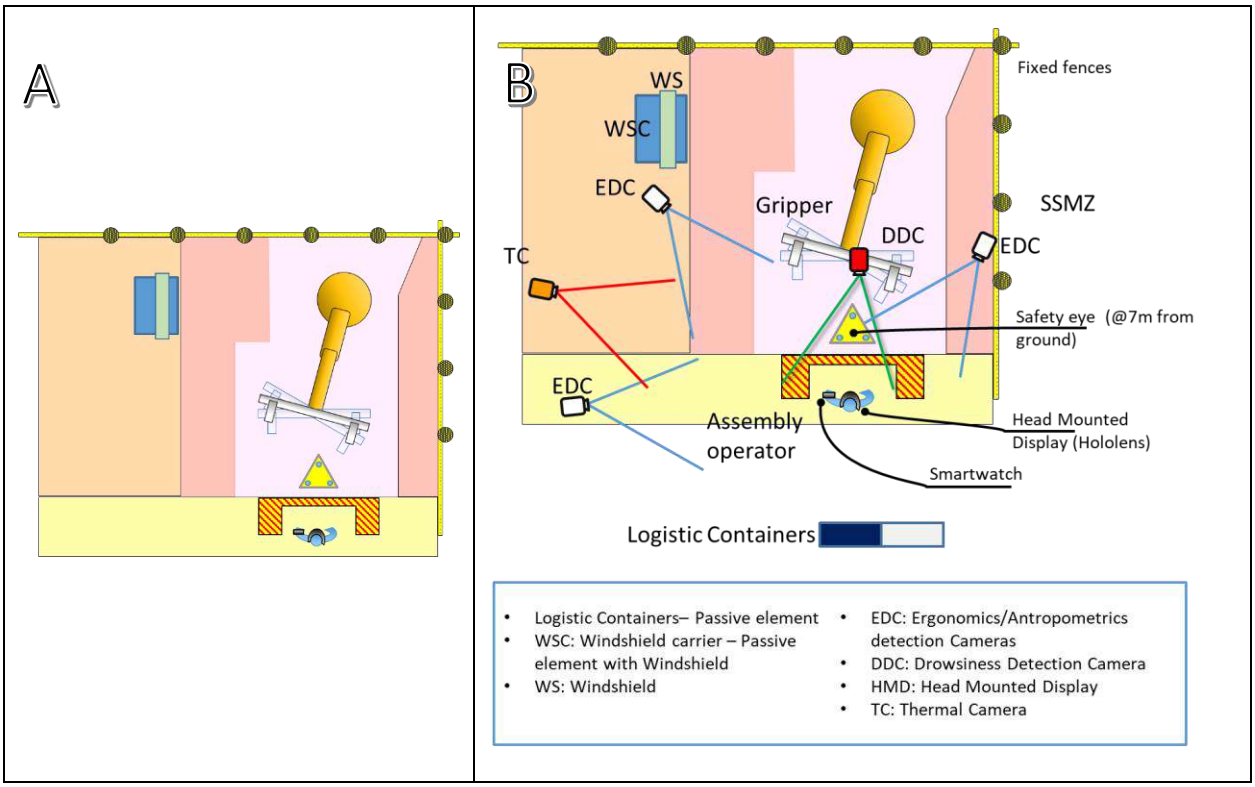


Figure 4: A: ground Layout; B: Main actors present in the manufacturing use-case of CPSoSaware.

Several cameras are indicated:

1. EDC: Ergonomics/Antropometrics detection Cameras
2. DDC: Drowsiness Detection Camera
3. TC: Thermal Camera

The set of EDC is used to make a robust classification of the operator's height and of the ergonomics of the operator during assembly operations. Three cameras are indicated to have a full and accurate covering of the working areas of the operator. The identified cameras are 3D stereo camara, but they can anyway produce inaccurate estimations due to the movements of the operator and to issues like environmental lighting. The CPSoSaware "cooperative localization" system enables a dynamic and adaptive identification of the most robust and reliable camera point of view, providing continuously a single valued of the evaluated parameters corresponding to the most reliable acquisition or to a fusion of data from several cameras. The system can work from one to multiple cameras with increasing reliability.

The drowsiness detection camera is used to detect the Operator's State. Being based on an Android OS, in this phase it was decided to implement the solution on a smartphone to evaluate its performances. The use of the Android system enables a simple and fast detection of the heartbeat by using a standard smartwatch worn by the operator; the heartbeat rate is an additional indicator of the drowsiness and stress level of the operator. Following improvements of the system the system could be adapted for further developments to different types of cameras (see paragraph 3.8.10).

The TC is a single thermal camera providing dynamically an evaluation of tiredness indicators of the operator based on it temperature. The camera performs the recognition of the operator's silhouetted and detects the group of pixels on the body with the highest temperature. To the overall system only a value of class of temperature is provided (see paragraph 3.7.2).

In the figures, the areas drowned on the ground represent the safety levels implemented on the access monitoring system; in Table 1, the correspondence between the area's colour and the action programmed on the robot in case of intrusion is described. All figures of scenarios, actors and similar refer to this table.

Table 1: Correspondence between the area's color and the action on the robot in case of intrusion used in all scenario's pictures.

Areas	Action
Yellow	Slow down the robot
Orange	Stop the robot
Red	Safe monitoring stationary robot
Purple	Safe monitoring stationary robot

3.3 Standard Scenario

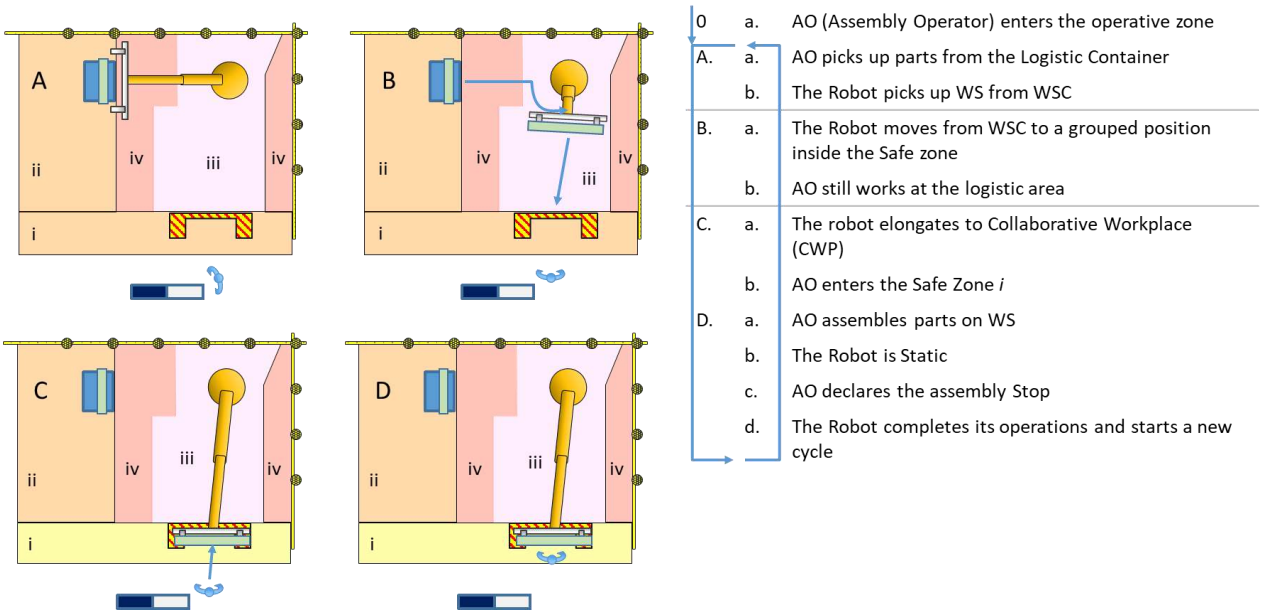


Figure 5: Standard operating cycle

The Figure 5: Standard operating cycle, describes the standard sequence of tasks from the robot and the operator. In the figure it is also evident how some specific safety zone (number *i* in particular) changes safety level during the execution of the program. This change is allowed as a function of the Safety Eye and is applied here to let the operator enter in zone *i*, only when assembly operations are performed. In the other phases (A-B) the operator cannot violate the area.

Table 2: Correspondence between the area's colour and the action on the robot in case of intrusion used in all scenario's pictures.

Task n°	Robot	AO (Assembly Operator)
0		Enters the operative zone
1 (A)	picks up the windshield from the carrier (WSC)	picks up parts from the Logistic Container
2 (B)	moves from WSC to a grouped position inside the Safe zone	continuous check tasks in proximity of the Logistic Container
3	rotates toward the collaborative zone while remaining in a grouped position inside the Safe zone	-
4 (C)	elongates its arm toward the collaborative zone linearly with a low and safe speed	ready to enter in the collaborative zone
5	reaches the collaborative zone and stops	enters the Safe Zone *i*
6 (D)	Static in Safe Stop	assembles parts on WS
7	completes its operations	declares the assembly Stop
8	*Other operations/New Cycle*	*Other operations*

Table 2 shows the sequence of tasks in parallel Human-Robot execution. Phase 0 is performed only at the beginning of the shift, or at the entrance of the operator in the work cell after a break. In the above table, phases from 1 to 7 are cyclically repeated. Letters in brackets refer to Figure 5.

3.3.1 A design operation continuum evaluation/Resilience and Safety

In the Use-Case the CPSoSaware subsystems and technologies concur to the definition of the initially listed two main use-cases: *Design operation continuum evaluation* and *Resilience and Safety*

In Figure 2 all the main CPSoSaware subsystems in the manufacturing use-case are represented together with their main developers and the relations to the original use-cases.

During the rest of the chapter, the scenarios are clustered as:

- Scenario-1: Operator State Monitoring
    - Android OSM
        - Face expression recognition
        - Smartwatch
    - Thermal camera
- Scenario-2: Anthropometrics and ergonomics analysis
    - Anthropometrics analysis, imagining and classification
    - Ergonomics, real time RULA evaluation
    - Cooperative localization for landmarks selection in multicamera systems
- Scenario-3: XR Training on the Job

With reference to the previously listed use cases, technologies concur to the execution of one or the other; some technologies are indeed applied to both use-cases. The Anthropometric analysis' functionality enables both functions of Safety (improved ergonomics) and supports and ensures an automatic operational continuum in case of change of operators by adapting accordingly the height of the gripper from the ground to the detected height of the operator.

Similarly, the AR application is made to ensure the Operational Continuum, enabling a dynamic adaptation to different products/ change of operators (also temporary) or new operators demanding for training on the job, while, on the other hand, the of safety and ergonomics related advice improve the level of safety for the operator and the application in general.

The functionalities related to Operator's State Monitoring directly apply to the Safety and Resilience Use-case. All technologies apply to the system functional resilience as well.

Based on the above considerations, the description of the trials and implemented functionalities in the following text will be based on the scenarios and not on the use-cases.

3.3.2 Other scenarios Overview and Update

With reference to previous deliverables, the standard scenario is substantially the same. Few simplifications have been applied in coherence with the decision to zero the potential impact on Safety. It has to be considered that all the systems defined in the CPSoSaware project have a networked dialogue (wired or wireless) and besides of the physical HW devices, the safety concerns should also be applied to the communication protocols, and this is complex for most of the technologies used and would involve the use of dedicated networks like ProfiSafe, typical of robotics and automation but not for standard CPS systems.

Considering thus a mixed network made of multiple devices, ensuring the Safety regulation alignment (the lowest grade of safety compliance is the level of the whole system) would be effort demanding and not focused in the project. Upon these considerations and in coherence to what has been explained in paragraph 3.1, the standard cycle was simplified in a way to have a Safety Certifiable structure. The final standard application is described in the previous paragraph.

In previous deliverables D6.3, besides of the main standard scenario, other scenarios were depicted to ensure the safety of the system (see Table 3). Of these scenarios, the logistic centered scenarios resulted not to be of immediate interest in the development of the CPSoSaware project (lines 1-2); the scenario n°3 is not applicable since the Rationale is not implemented for safety considerations, the scenarios 4,5 and 6 are implemented in the access management software in the robot work cell controller. They are a base part of the safety, implemented and certified in the work cell since they are necessary, but they are not related to CPSoSaware developments.

In D6.4 the outlined additional scenarios were aligned with the current and final developments that are described below. It is important to highlight and note that part of the developments and scenarios described in previous deliverables are effectively developed and implemented since core part of the robotic application. In the current deliverable, the attention is on collaborative development of the CPSoSaware project according to the summarizing Figure 2.

Table 3: additional scenarios besides of the standard scenario as planned in D6.3

	Name	Type of scenario	Rationale	Implemented
1	Gravity Shelf Refill Scenario	Scenarios for ergonomics and Safety	It is necessary to fill the Gravity Shelf	No – not of interest for CPSoSaware
2	Windshield Container Refill Scenario	Scenarios for ergonomics and Safety	It is necessary to fill the Windshield Container (only simulated)	No – not of interest for CPSoSaware
3	Robot Singularity Scenario	Scenarios for ergonomics and Safety	The operator brings the robot close to a singularity point	No – not applicable without Hang Guiding
4	Slow down zone entrance - SSM	Resilience from Safety zones violation	The operator violates the red zone	Yes – Access management system
5	Safety Zone Violation Scenario 1 – SMS	Resilience from Safety zones violation	The operator violates the red zone	Yes – Access management system
6	Safety Zone Violation Scenario 2 - SMS	Resilience from Safety zones violation	The operator violates the red zone	Yes – Access management system

The additional scenarios are described in detail in the following paragraphs.

3.4 Technologies implemented and architectural solution and implemented setup

Next Figure 6 represents the main subsystems in the system that together realize the manufacturing use-case. The system in the figure looks to be organized on a tree model where a central orchestrator is collecting the information from the other modules and coordinating the information exchange. As stated, the robot subsystem is a standalone system that only adapts the height of the gripper from the ground according to a value published on the network. As in D5.4 (paragraph 4.2.1 ROS communication) the CPSoSaware system in manufacturing is based on ROS (Robotic Operating System) as middleware. In ROS, programs run on isolated nodes that can communicate using a publish-subscribe model. Based on this model each subsystem publishes the generated messages and subsystems that need the information directly subscribe to the specific Topic and read the subscribed message. The logic used is mainly asynchronous and was used because of the nature of the information exchanged. In the architecture represented in Figure 6, each subsystem published values in appropriate Topics and subscribes to topics of interest, establishing a direct one to one communication between each module.

The robot used is not natively equipped with ROS interface modules and software and the communication can be established mainly by using an adaptation of the “KUKA.RobotSensorInterface” module. This module is designed to connect and control the robot through UDP connections to sensors publishing their data on a TCP/IP connection. Through this SW and with the addition of a specific additional module on a ROS industrial system, it is possible to convert the flux of data from a ROS system to a TCP system and communicate with the robot. In the final application we decided to include an intermediate module based on a computer programmed in LabVIEW that communicated toward the CPSoSaware system in ROS and toward the Robot in a digital mask of bit connected to digital IOs of the robot.

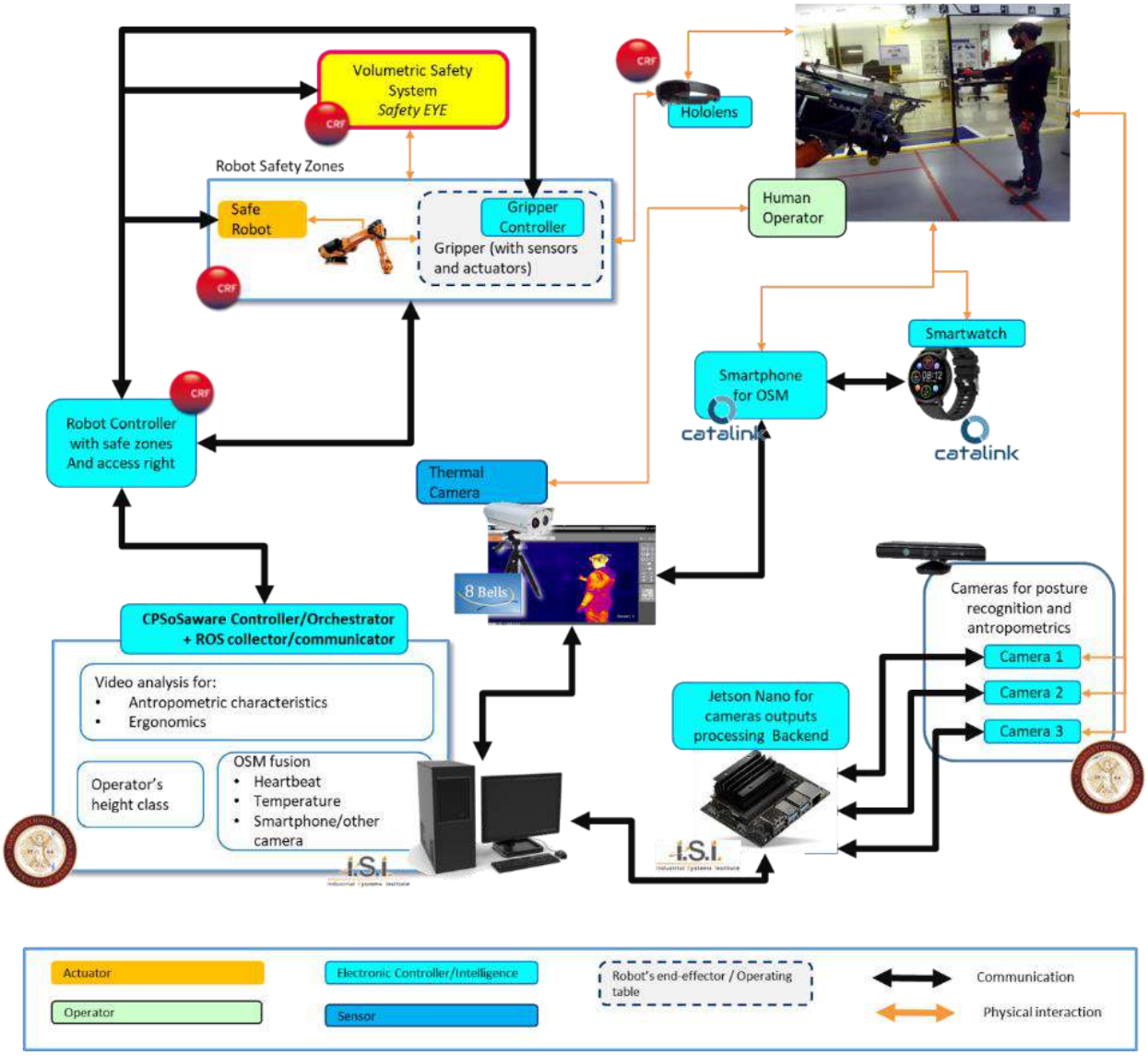


Figure 6: System solution architecture of the manufacturing Use-Case

This approach is ensuring a simple, coherent, and safe interaction with any type of robot and enables a richer and more universal use of the ROS interface (in paragraph 3.5 the LabVIEW application is detailed). In Figure 6 the CPSoSaware orchestrator is a computer that performs several actions based on intermediate ROS Topics from other systems; furthermore, it is connected directly (wireless) to the Android Smartphone. In its role the so called "orchestrator" is elaborating part of the data fusion from other subsystems, publishing on ROS the finally elaborated information. Because of this role it is indicated as an information collector and orchestrator, but the architecture used is indeed based on a direct *one to one* logic.

3.4.1 ROS network configuration/Topics

To integrate the components and algorithms mentioned, we decided to use the Robotic Operating System (ROS) as our middleware. ROS is an open-source framework that includes a collection of libraries and tools for building distributed applications as it enables developers to create complex robotics systems in a modular way, using a variety of pre-existing libraries and tools, and to easily connect these components together. One of the main advantages of ROS is that it provides a standardized way for different components of a robotics system to communicate with each other. Components can communicate using a publish-subscribe model, in which one component (the publisher) sends data and or more nodes can subscribe to this topic and consume the data. Thus, applications run on separate nodes that can communicate with each other using a publish-subscribe model.

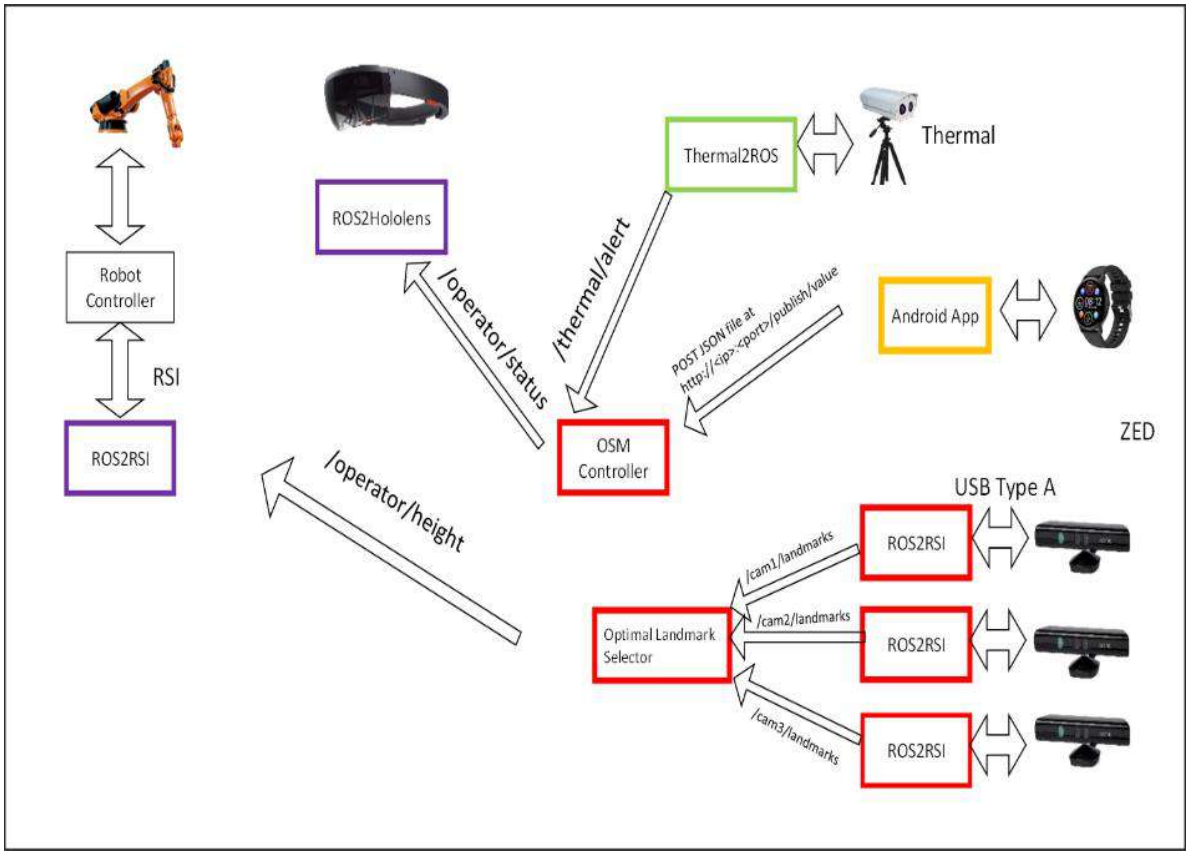


Figure 7 View of the ROS environment

For integrating the components of this demo, we created one ROS node for each camera used in the experiment (as shown in Figure 7). Each node captures a frame from the camera at a rate of 10Hz, processes and extracts the landmarks, and then publishes them to a relevant ROS topic. The decentralized structure of ROS allows us to connect the cameras to different physical machines, such as the Jetson TX2 embedded device. Another node subscribes to the topics published by the cameras and selects the optimal set of landmarks, which will be used later to calculate the operator's posture.

The OSM Controller node can be described as the orchestrator of this architecture. It is responsible for collecting data from both the thermal camera and the android applications. The former uses a ROS node for translating the readings from the camera to ROS messages and the latter uses a REST interface to post the updates of the status of the operator according to the reading of the smartwatch. Both data flows are collected by the ROS OSM Controller which assembles the final ROS message and outputs a single message that describes the status of the operator based on the information provided by the smartwatch and the thermal camera. This information is published under the relevant topic, which is subsequently consumed by the interested actors (HoloLens).

Table 4: Publishable ROS Topics

Publisher	Subscriber	Type of interface	URL/ROS Topic	Data Type	Description

Android app	OSMController	REST	http://<ip>:<port>/publish/value	POST a JSON file	Example: {
					"operator_state" : "alerted/drowsy",
					"heart_rate_value" : 75.0,
					"timestamp": "2022-03-21T12:46:13.101Z"
					}

Thermal2ROS	OSMController	ROS	/thermal/alert	Int16	An Alert which indicates the operator's status. Possible values: 1-> Hypothermia, 2-> Normal, 3-> Signs of fatigue, 4->Hyperthermia
zed_ros	Optimal Landmark Selector	ROS	/cam#/Landmarks	PointCloud2	Landkmarks detected by the zed camera
Optimal Landmark Selector	ROS2RSI	ROS	/operator	Int16	The class of the height of the operator as calculated by the fusion of the three sets of landmarks
ROS2RSI	Robot Controller	RSI			Set the height of the operator
OSM Controller	ROS2Hololens	ROS	/osm/status	String	The status of the operator as resulted by the fusion of the inputs by the thermal camera and the smartwatch.

OSM Controller	ROS2Hololens	ROS	/osm	String	The status of the operator as resulted by the fusion of the inputs by the thermal camera and the smartwatch. Example: {
					"operator_state" : "alerted or drowsy",
					"heart_rate_value" : 75.0,
					"thermal_alert": 1
					}

OSM Controller	ROS2Hololens	ROS	/ergonomics	String	An alertr raised if the posture of the operator is not good.

Regarding the physical topology of the networks, all entities are connected to the same Local Area Network wirelessly or not. For the purposes of the demo, there was no need for providing access to a remote source but due to the flexibility and the modularity of our architecture, this could be done with minimal effort.

3.5 Robot system and connection

As described in paragraph 3.4, the implemented solution uses a simple and flexible LabVIEW interface converting the ROS messages form the CPSoSaware system to a digital mask of bit series of digital IO on the Robot's controller. The interface is based on the Ros for LabVIEW software module developed by the Tufts University Center for Engineering Education and Outreach [4].

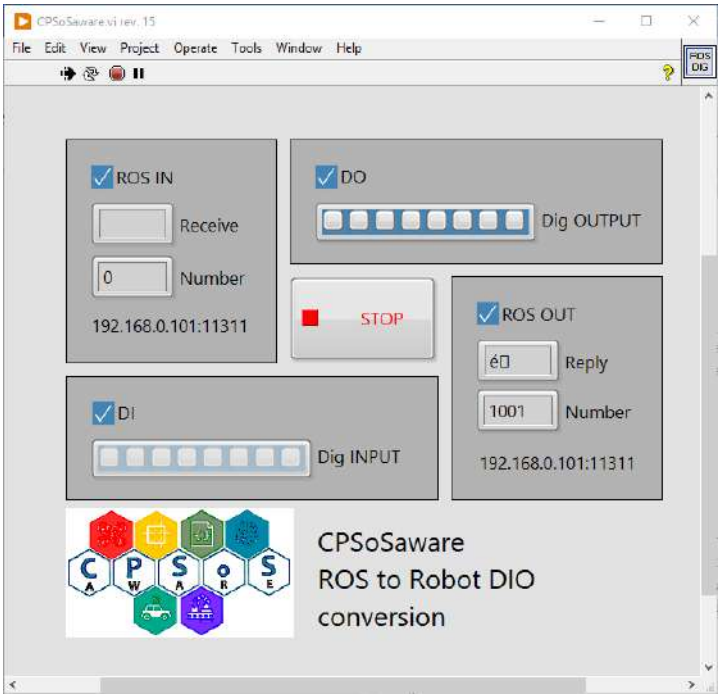


Figure 8: Labview panel of the ROS-DIO interface developed

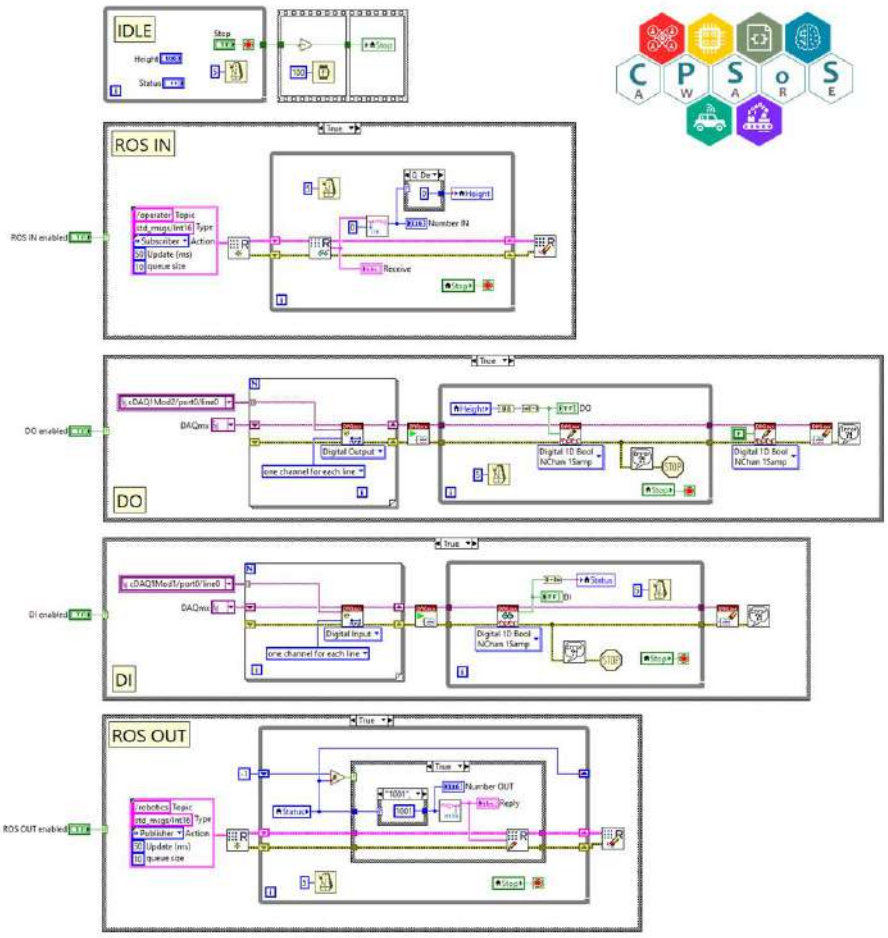


Figure 9: Main control panel structure of the interface program

There are five cycles:

IDLE: it is the main loop that always runs (the others can be deselected) and manages the stop

ROS IN: if enabled, it reads from the topic called "operator", using the library ROS for LabVIEW software. The message is of type Int16, the values that can arrive are: 1001, 1002, 1003, 1004, 1005, which correspond to the height table of the operator. The value is stored in the variable (Enum) "Height", the value in position 1 is 1001, the one in position 5 is 1005.

DO: If enabled, it manages the sending of the digitals to the robot controller. The numbers 1 to 5 (position of the Enum variable named "Height") are converted into an 8-bit Boolean array. Only the first 3 bits are connected to the digital card, the others are always equal to zero (the maximum number is 5, so 10100000)

DI: If enabled, it manages the reception of digital signals from the robot controller. The input Boolean array has 8 bits, but physically only the first channel is connected, so the robot sends the number 0 or 1. The number is stored in the "Status" variable, a variable of type Enum which in the position 0 contains the value 1001 and the position 1 contains the value 1002.

ROS OUT: if enabled, it writes to the topic called “robotics", using the library ROS for Labview software. The message is of type Int16, and the value allowed are 1001 and 1002 (the value of the variable “Status”)

3.6 Demonstration Setup

The pilot has been set up in CRF with the contribution from all the partners participating in the Manufacturing Use-Case. In Figure 25 there is represented the Process Simulate design of the workcell coherently with the affective setup. following Figure 10-A is a modified detail of Figure 25, while Figure 10-B is the image visualized by the Safety eye with the design of the active zones in one specific moment of the execution.

On Figure 10-A the safety zones are added accordingly to the color coding of Table 1. Some elements are added in the image:

- In dark blu the table used as deposit of the logistics (components to be assembled)
- In Light blue the effective Operator Work Place (the zone where the operator can act when not in the interaction phases)
- in green the Safe zones

Light blue and green zones are always accessible to the operator.

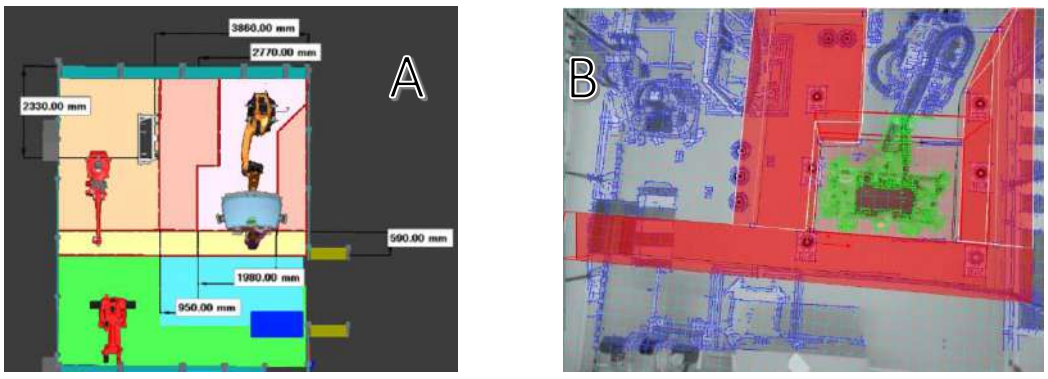


Figure 10: System solution architecture of the manufacturing Use-Case

In the Figures below same images of the real testing are visible. The cameras are positioned on tripod for simplification and to test several positions (

Figure 11: Pictures taken during the trial testing

3.6.1 Thermal camera setup

Thermal camera has been setup inside the working cell where the manufacturing demo trial was deployed. It has been placed approximately 2,5 meters from the spot where the operator stood to work with the robotic arm. The thermal camera was connected via ethernet to the local LAN computer network, which acted as a data channel between the camera feed and the rest of the components. Thermal images retrieved by the camera had been broadcasted in real time in 8BELLS AI Image processing unit, which then analyzed the feed through Computer Vision algorithms and determined the real-time temperature of the operator. Through the connection of the camera and the computer, over the local network of the manufacturing shell, data retrieved could be accessed, along with the thermal image and User Interface (UI) of the camera through the browser. As explained in D5.4 “Final version of CPSoSaware Integrated Platform”, since the same computer was connected to the internal network where the flow of ROS messages took place, we declared the IP of the master-subscriber in the ROS configuration script, and we started running the publisher in which the AI Algorithm is embedded. The messages were then sent to the ROS topic thermal\alert through which the subscriber was waiting to listen.

Once the publisher started running, every 3 seconds the algorithm pulled the thermal image data, isolated all temperatures, and kept the operator's highest value. Then calculated in which category the temperature fell and then sent the message to the subscriber in the form (Int16).

3.7 Scenario-1: Operator State Monitoring

The analysis and tracking of the operator's state supports safety and resilience in the workcell: distraction, tiredness, and other states of the operator causing discontinuity and lack of attention in the operator's tasks can generate risks for the same operator and his colleagues. After the COVID pandemic period also the thermal analysis of the operator took relevance. Its application not only enables the detection of temperature and fever situations, but it is also an indicator of stress and fatigue.

The subsystems implemented in the OSM use-case is made of the data fusion of information from the **"OSM Android Application"** including face and heartbeat monitoring and the **"thermal evaluation monitoring".** The overall system does not modify in any moment the standard cycle execution (Figure 5) and thus no updated representation of the cycle.

The execution of the OSM module is affecting the standard scenario as follows:

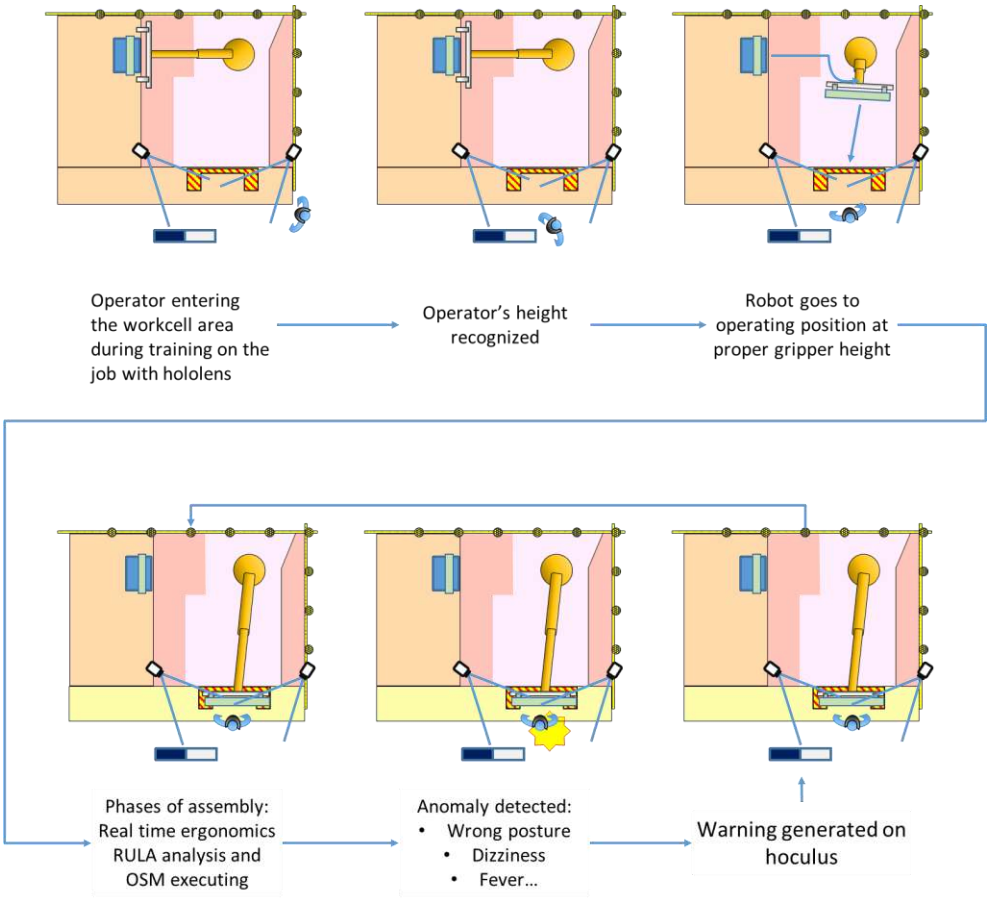


Figure 12: Scenario for all the warning generation

The execution of the OSM module, as well as the execution of the Ergonomics recognition scenario generates solely a warning on the HoloLens interface. The information can be anyway saved, monitored, or used for reporting.

3.7.1 OSM Android Application

As stated above, the Operator State Monitoring Android Application is a modified version of the DSM application which was designed and implemented for the automotive use case. The main goal of the application in the manufacturing use case was to provide a rough estimate of the state of the operator based mainly on the head pose and the yawning activity of the operator and on the heart rate monitoring which was retrieved from a smartwatch. That information combined and fused with the information coming from the thermal camera which continuously monitors the body temperature of the operator, provide useful input about the general status and attention level of the operator, and can potentially enhance the security while interacting with the robot.

For the purposes of the experiments one smartphone with the OSM application running, was installed on the gripper and a smartwatch with the HR monitoring application was provided to the operators. The specifications of the smartphone and the smartwatch can be found in Table 5.

Table 5: Specifications of the smartphone and the smartwatch

Device	Specifications
Smartphone	• Model: Xiaomi 11T • OS: Android 12 • Ram: 8GB • CPU: Octa-core 3.0GHz • Storage: 128GB
Smartwatch	• Model: Galaxy Watch4 • OS: Wear OS 3.5

Two operators conducted the experiments, one male and one female. The female operator conducted a 2-minute assembly and the male operator a longer session while wearing also the HoloLens while performing the necessary procedure.

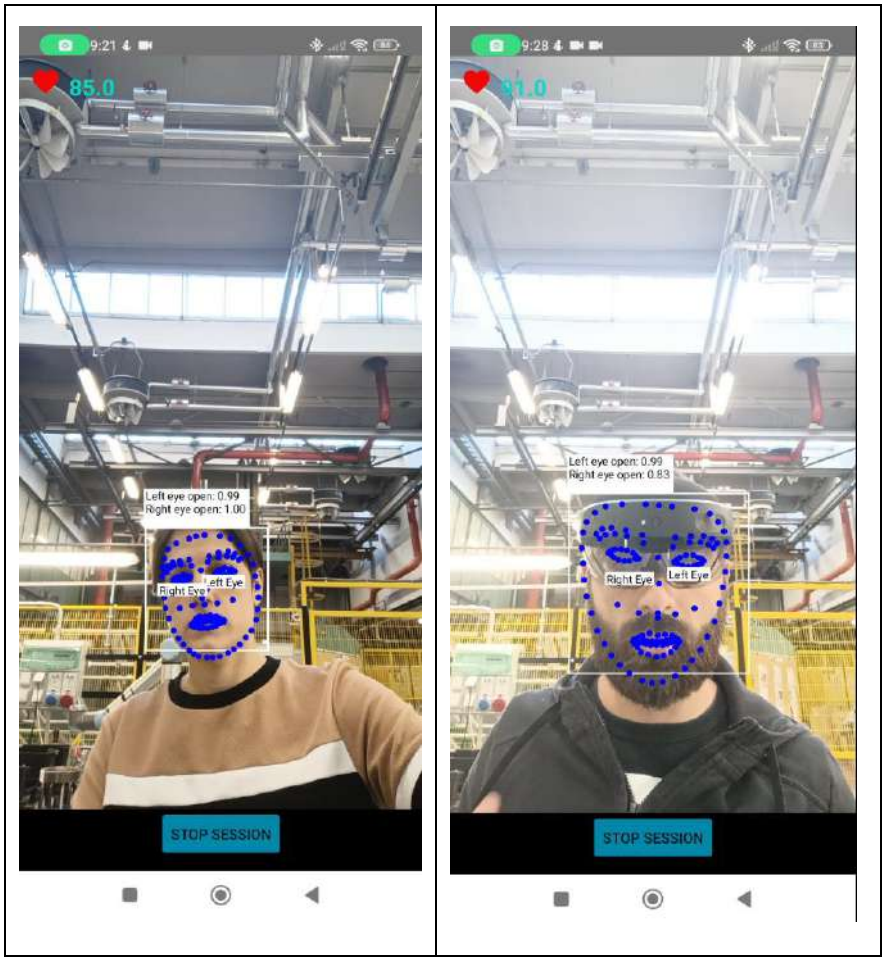


Figure 13: Different operators performing the assembly task while OSM application is running

The OSM application analyses each incoming frame from the front camera of the smartphone and stores the analysis in a JSON file which has the structure depicted in Figure 14.

```
[
    "SessionUUID": "6f575cbb-8dd3-456a-ad9b-0b0d897023b9",
    "Session Timestamp": "2022-11-15T12:46:12.598Z"
  ],
  [
    "Frame Number": "1",
    "FrameUUID": "769aae23-707d-43fe-b8d3-af3d56112e54",
    "FrameTimeStamp": "2022-11-15T12:46:12.986Z",
    "Number of Detected Faces": "1",
    "Eyes Closed": false,
    "Yawning": false,
    "Distracted": false,
    "Alert": false
  ],
  [
    "Frame Number": "2",
    "FrameUUID": "317c830b-afbf-4a1b-aff1-8ca737bc0485",
    "FrameTimeStamp": "2022-11-15T12:46:12.101Z",
    "Number of Detected Faces": "1",
    "Eyes Closed": false,
    "Yawning": false,
    "Distracted": false,
    "Alert": false
  ],
  [
    "Frame Number": "3",
    "FrameUUID": "51cd0a61-dc1b-407a-af94-750b28baa73b",
    "FrameTimeStamp": "2022-11-15T12:46:12.285Z",
    "Number of Detected Faces": "1",
    "Eyes Closed": false,
    "Yawning": false,
    "Distracted": false,
    "Alert": false
  ]
]
```


Figure 14: JSON file of the OSM application analysis

From the analysis of the JSON file we were able to extract the following conclusions about the functionality of the application and its potential use in such an environment.

- The OSM application captures the incoming frames of the front camera and forwards them to the machine learning algorithm running on the phone in order to firstly detect the face of the operator and subsequently analyse the facial expressions and make an estimation about his/her state. If the algorithm cannot spot a face within the given frame, then this frame is discarded. As stated above, the smartphone was placed on the gripper of the robot which although it is sufficient when the operator performs the assembly task, it cannot capture the operator's face while he/she is heading away from the robot in order to take the necessary parts. As depicted in Figure 15 and in Figure 16, the operator's face is not captured by the application for a significant number of frames, which indicates that we cannot perform a continuous monitoring of the facial expressions and the head pose. Thus, as a future work and enhancement of the solution additional cameras could be placed within the working area of the operator. Moreover, the difference among the analysed and the discarded frames was bigger for the male operator, since he was also wearing the HoloLens and it was harder for the application to conduct the face detection and the facial landmarks localization.

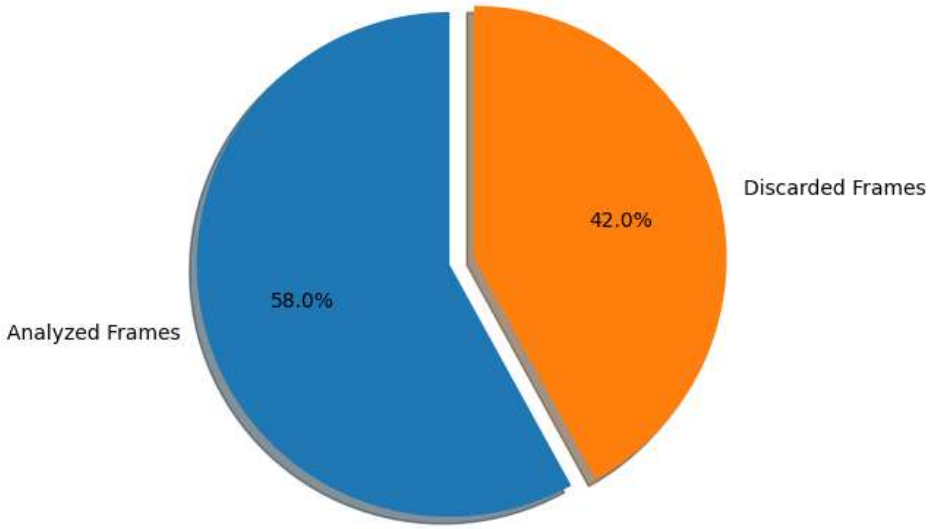


Figure 15: Number of analysed and discarded frames from the OSM application for the female operator.

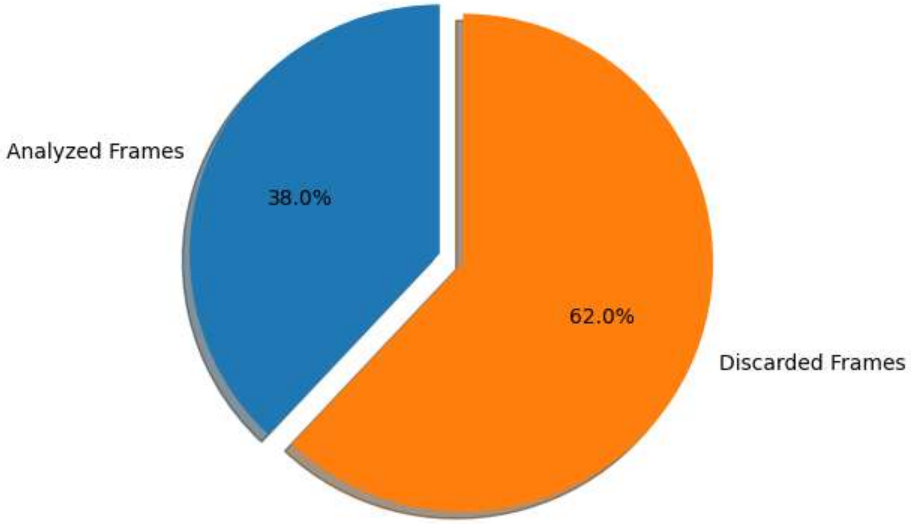


Figure 16: Number of analysed and discarded frames from the OSM application for the male operator.

- Regarding the heart rate monitoring, the OSM application was able to continuously retrieve the heart rate values from the smartwatch. As depicted in the diagrams, some disconnections have been also noted and in that case the heart rate value was equal to 0, although the frequency of those was very low and did not affect the overall procedure.

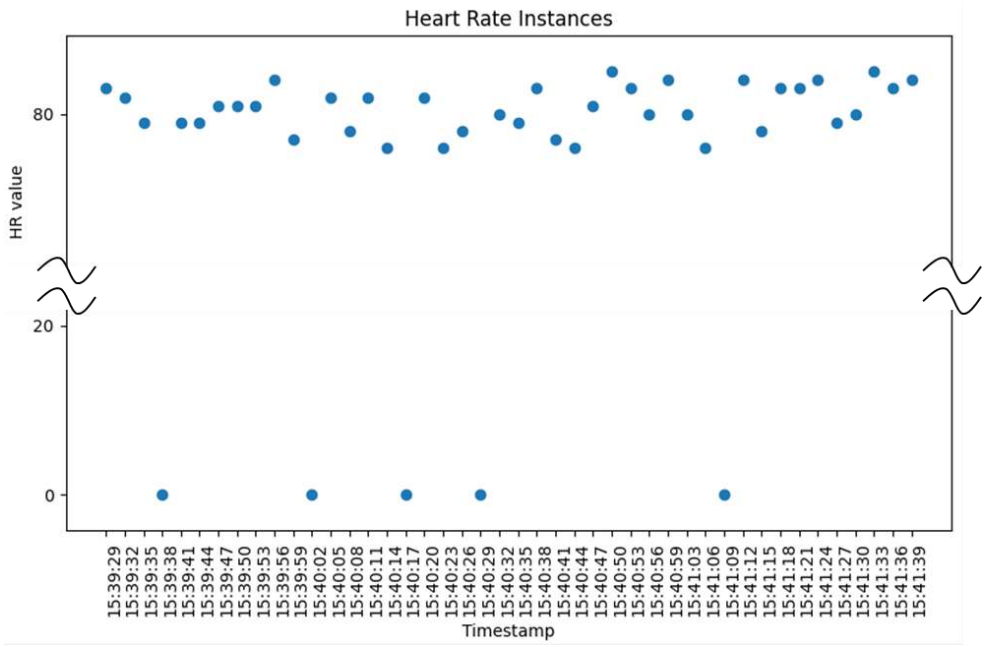


Figure 17: Heart rate of the female operator

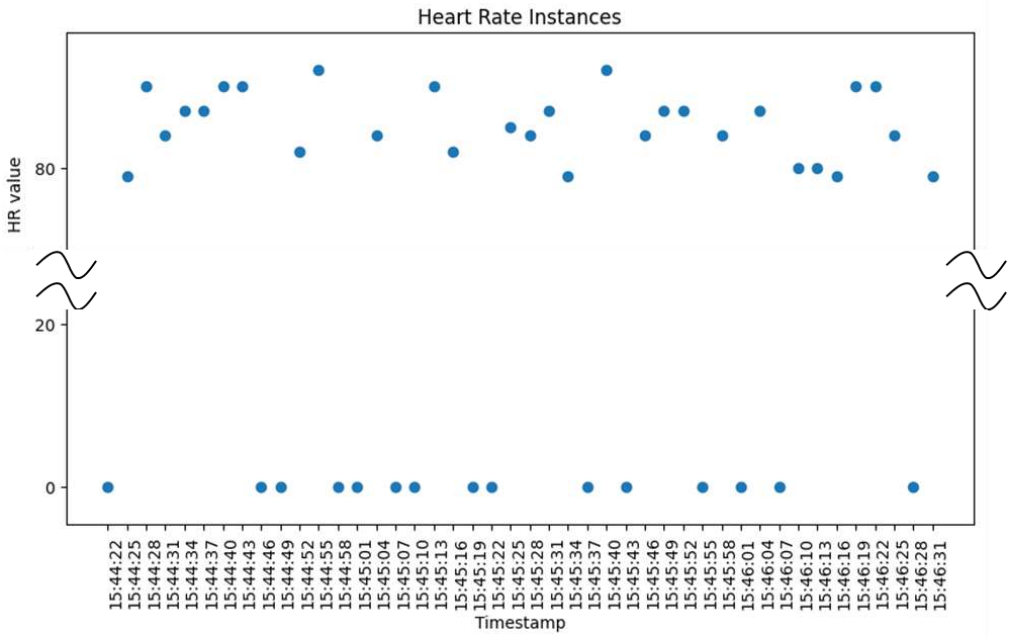


Figure 18: Heart rate of the male operator

- Due to the constant movement of the operators for making the requested actions and the periods where they were not captured from the camera's smartphone, the detections of yawning and distraction were minimum and thus there is not present a relevant graph in this report.

As a general remark, we could state that even though changes and alterations should be performed in the OSM application in order to meet in a better way the requirements of the manufacturing environment, the solution could be adopted and provide an extra source of information which can help to safeguard the people within the working cell.

3.7.2 Thermal camera

The integration of anthropometric thermal monitoring in the Operator State Monitoring scenario was successful as it has facilitated the monitoring of the operator's body temperature in a non-invasive manner, in real time. Data and information retrieved from the setup of the thermal camera inside the manufacturing shell have been successfully used to identify if the operator experienced hypothermia, signs of fatigue/ drowsiness, and fever.

One of the main benefits of anthropometric thermal monitoring is that it allows for continuous, real-time monitoring of an individual's body temperature. This proved to be very useful in the industrial setting where the manufacturing demo trial took place, ensuring that the operator collaborating with the robotic arm was not experiencing any adverse effects on his health. Table below presents a partial extract of the dataset collected from the integration of the thermal camera during the manufacturing trial demonstration.

Table 6: Partial extract of thermal camera dataset

#	Date time	Temperature (^{0}C)	Alert	Operator Status
1	19:20:30	36,6	2	Normal
2	19:20:33	35,9	2	Normal
3	19:20:36	35,6	2	Normal
4	19:20:37	36,3	2	Normal
5	19:20:40	36,3	2	Normal
6	19:20:43	35,2	1	Hypothermia
7	19:20:46	35,3	1	Hypothermia
8	19:20:49	36,2	2	Normal
9	19:20:52	36,4	2	Normal
10	19:20:55	36,3	2	Normal
11	19:20:58	36,6	2	Normal
12	19:21:01	36,2	2	Normal
13	19:21:04	36,3	2	Normal
14	19:21:07	36,6	2	Normal
15	19:21:10	36,7	2	Normal
16	19:21:13	36,7	2	Normal

17	19:21:16	36,5	2	Normal
18	19:21:19	36,5	2	Normal
19	19:21:22	36,4	2	Normal
20	19:21:25	36,5	2	Normal
21	19:21:28	36,5	2	Normal
22	19:21:31	37,5	3	Signs of fatigue
23	19:21:34	37,7	3	Signs of fatigue
24	19:21:37	36,6	2	Normal
25	19:21:40	35,3	2	Normal

Each of the alerts produced and transmitted through ROS messages has been visible in the HoloLens that the operator was wearing as part of the overall demonstration scenarios. Sudden decreases (i.e., lines 6 and 7 of the table above) or increases (i.e., lines 22 and 23 of the table above) in the operator's thermal measurement can be explained as interference from external factors, like for example, thermal measurement of the HoloLens instead of the operator.

Overall, anthropometric thermal monitoring proved to be a useful technique that allows for continuous, non-invasive monitoring of an individual's body temperature. This information can be used to identify operators who may be at risk of developing hyperthermia or other health problems and can help ensure that workers in an industrial environment are able to maintain a healthy body temperature in a variety of settings.

From the overall view of Table 6, a simplified threshold table has been set. The Table 7 is showing a simplified version of the table, describing ranges, physical meaning and alert Code.

Table 7: Thermal Camera ROS published values

Alert	Temperature	Operator Status
1	T<35.5	Hypothermia
2	35,6 > T >37,4	Normal
3	37,5 > T >38,4	Signs of fatigue
4	T >38,5	Hyperthermia (Fever)

3.8 Scenario-2: Anthropometrics and ergonomics analysis

Robot behaviour can potentially affect the operator's safety, health and comfort. Enhancing operator physical monitoring reduces the risk of accidents, strain and musculoskeletal disorders. Within CPSoSAware, we propose a multi-stereo camera system and novel distributed fusion approaches that are executed on accelerated AI platforms, continuously monitoring the posture of the operators, while they collaborate with the robot, and assisting them to avoid uncomfortable and unsafe postures based on their anthropometric characteristics and a real-time risk assessment standard methodology based on ergonomics.

When operators are physically interacting with a robot, trying to accomplish a collaborative task, their posture is inevitably influenced by the robot's movement and trajectory. To overcome this challenging situation, an adapted system is required which would be able to be adjusted according to human anthropometrics and the special characteristics of each operator, improving in this way their physical ergonomics.

In manufacturing, the tasks of the robots that collaborate with humans are usually performed as a sequence of known and pre-defined actions, since it is not safe to allow robots to improvise. A parameter that can affect the robot's movement is the characteristics of the operator's body (i.e., height). This parameter does not change the main action of the robot but can be used to select the best adjustment of the robot's position and configuration in order to optimize the ergonomics assessment of the operator. The adjustment happens only at the beginning of the robot's operation since, for security reasons, it is not appropriate to have a dynamic adjustment.

The main objectives of this Pilot scenario are:

- To increase the operators' situational awareness, regarding their well-being related to their anthropometric characteristics, during human-robot collaboration in real-time manufacturing conditions.
- To provide an ergonomic analysis of the operators under different actions during their collaboration with a robot.
- To adapt the robot's trajectory to facilitate the operator's ergonomics.

The insertion of the Anthropometric analysis module from modifies the robot's cycle and scenario as in Figure 19

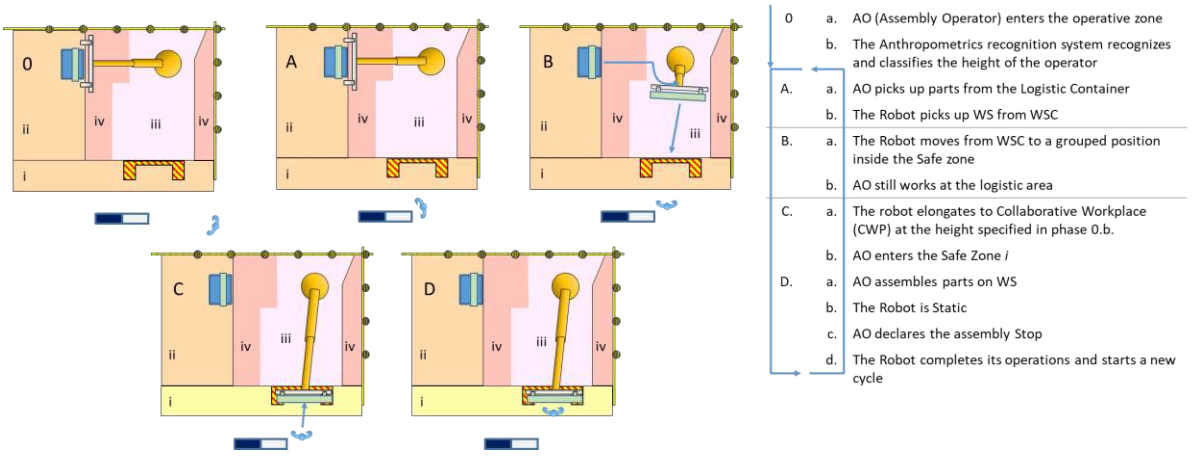


Figure 19: Height adaptation of the gripper

The scenario used in case on anomalies detected by the ergonomics RULA analysis module is represented in Figure 12.

3.8.1 Golden Zone and Strike Point

Approaching the Lean Manufacturing model for the workstation under study, ergonomics analyses of the working area and the activities performed by operators were carried out. For this study, an analysis model was used to assess the exposure of workers to risk factors that can cause trauma in the upper limbs.

In the specific use case, activities that operator have to carry on are to assembly components from logistic kit on the windscreen; in the image below, we can see the operative area from the Operator's Point Of View (POV)

Figure 20: Assembly process

In order to avoid incorrect postures, both in terms of position and duration, which can cause physical and cognitive problems for the operators and thus also a decline in productivity, the focus was on making all assembly operations fall within the so-called Strike Point.

First of all, we need to define the Strike Zone. The Strike zone is an ideal working area that ensures reduction of Not Added Value Activities (NVAA) and difficult or unnatural operations, keeping materials, tools and operative plan in specific positions, reducing fatigue and increasing productivity. This ideal volume is positioning in front of the operator and is directly correlate to the operator's height; on an ideal Z-axis, this volume occupied from eyes line to knees line.

A specific zone was interested in ergonomics and anthropometrics analysis on our automotive assembly process use case: the Strike Point. This is a specific area of strike zone included between shoulders' line and elbows' line (considering arms along the body); working on this area allow operators to be more comfortable when operating on light and precision activities. According to this, we can identify three sub-areas:

- Strike Point: area between shoulders and elbows. It could be possible to move arms up and down between elbows and shoulders and left and right without extending elbows.

- Strike Zone: area between eyes and knees. It could be possible to move arms up and down between knees and eyes level, and right and left extending elbows.
- Red Zone: area above the eyes level and below the knees level.

Reflecting on an ideal XY plan, we can identify a specific working area in front of operator: the Golden Zone. This area is defined as the preferred working area for the operator so that everything is within reach: more precisely, the necessary tools and equipment must be within 60deg in front of the operator and no further than the semi-extension of the arm (in zone AA). In fact, the area is divided into sub-areas (details in image below) in which we can find an:

- AA: all components must be provided at the workstation between the field of view and the working height.
- A: components are placed in an area three times larger than the working area. Components may be captured by extending the arms and using both hands.
- B: components can be captured by extending the arms above the shoulders. Components are placed in an area 6 times larger than the working area.
- C: components can be captured by turning.
- D: components can be captured by walking.

Combining Strike Zone and Golden Zone concepts, we have a preferred working area in terms of height and reach. According working area designed, it could be

- avoiding any curved or unnatural body postures
- avoiding forward or sideward arms extension (reducing fatigue and increasing precision)
- having a working area into operator's the maximum visual distance and hands' operative distance at around 300mm from eyes
- having all the tools and working materials in position so that all frequent movements are made close to the operator's body and with elbows bended

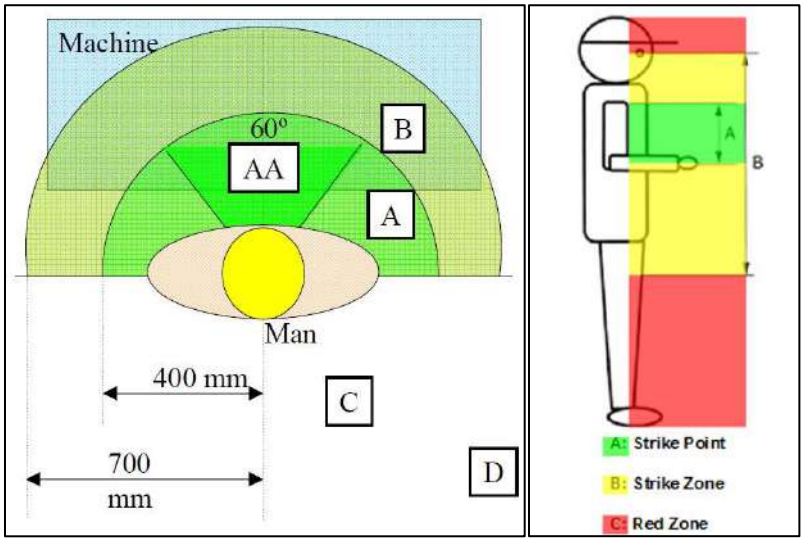
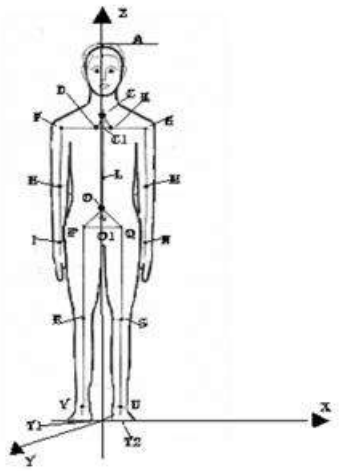


Figure 21: Golden Zone and Strike Zone

On this scenario, in optics to optimize height of working plan into an assembly process, we identified in Strike Point the best working area (in terms of height) and Golden Zone AA area (in terms of reach) for operator's tasks.

In order to be able to more reliably identify the optimal positions that the workstation, and thus the robot, had to assume, the most important databases on the anthropometric characteristics of the worker population were analysed. This is because anthropometric characteristics vary according to so-called group factors such as age, gender, regional and geographical differences. Analysing the literature, we chose the ANSUR anthropometric database. Then, analysing data for gender and for percentile (that represents people in percent who have the value less than or equal to the said value, for that anthropometric size) we carried on analysis on ergonomics positioning of working plan.

Starting from these tabulated data, all information regarding percentile *1, 5, 10, 15, 20, 25, 30, 35, 40, 45, 50, 55, 60, 65, 70, 75, 80, 85, 90, 99* were analyzed for both the population of female workers and the population of male workers, and an average group height was identified that respected the concept of Strike Point.



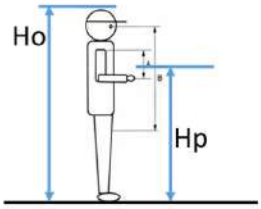
	15%-ile	Uomo
A-T	**Stature**	1681,23
C-C1	**Cervical - clavicular link**	52,11
F-G	**Clavicular link**	341,01
F-H	**Shoulder - elbow**	264,25
H-I	**Elbow - wrist**	246,72
O-O1	**Pelvic link**	84,53
O-C	**Cervical - pelvic link**	410,59
P-R	**Hip - knee**	407,01
R-V	**Knee - ancle**	400,09
V-T	**Ankle height**	74,29
I-Z	**Hand grip line**	72,87
P-Q	**PQ**	143,63
AC	**AC**	304,72
C-E	**C - E**	59,21
D-E	**D - E**	55,98
L-O	**Lumbar link**	136,98

Figure 22: Example of Strike Point measure on 15-Percentile Man (ANSUR DB)

Analysing data from ANSUR DB, we calculated each Strike Zone for each percentile and for each gender; then, using all ranges as independent each other. As shown below, it was possible to divide the entire population of male and female workers into five macro-groups, each identified by an average reference height.

Figure 23: Strike Point areas vs Percentile: Clustering

Each cluster was represented by average value of included strike point areas; then, with this value, it was possible to classify operators on the basis of its height. As shown below, for each cluster it was used the value of average strike point areas as assembly plane height (Hp), and associated to each operators' maximum (or minimum for last cluster) height value a scalar value as ID Operator Group.



Expected scalar	Ho=height of the Operator	Hp= Height of assembly plane
1001	h < 1544	1065
1002	h < 1641	1135
1003	h < 1727	1198
1004	h < 1829	1265
1005	h > 1829	1349

Figure 24: Operators' ID group classification

Following the identification of the optimal heights for the workstation, a simulation model of the assembly process was defined within the SIEMENS Process Simulate SW in order to verify and analyze the obtained scenarios.

3.8.2 Process Simulate

To reduce costs of design phase and to anticipate all critical issues at the first design stage, it is necessary to adopt digital manufacturing solutions. Digital manufacturing consists of the use of an integrated computer system with tools to represent in a digital environment production process.

There are a lot of point of view to see manufacturing process through tools and solutions of digital manufacturing; from production-oriented design to computer-aided manufacturing, from flexible to lean manufacturing. Digital manufacturing is a key aspect of the integration between Product Lifecycle Management (PLM) and shop-floor applications, as it facilitates the flow of product-related information between design and manufacturing stakeholders. Benefits with this approach include:

- allows product, process, and resources information to be associated, visualized, and modified, consistent with product design
- allows optimization of production and the visualization of 2D/3D component data and machining and tooling instructions
- reduces commissioning costs by validating in a virtual environment
- accelerates creation of factory models by finding optimal modes (layout, logistics flows, throughput) at first steps of production start-up
- simplifies the sharing of high-quality data within the organization
- providing real-time access to life-cycle data

To pursue these objectives, we chose Siemens PLM Tecnomatix Process Simulate software as discrete event simulation environment. Process Simulate is a digital manufacturing solution for manufacturing process verification in a 3D environment, that facilitates virtual validation, optimization, and commissioning of complex manufacturing processes. The software, with any own simulation engine for robotics inverse kinematics computing, allows users to verify the feasibility of an assembly process introducing in a 3D realistic scenario all actors of real production process. Through this "digital twin" of assembly line, designers can determinate most efficient assembly sequences, detect collisions of components, equipment and point cloud of environment with a specific clearance, identify the best layout of cells e the position of equipment. Process simulate allows also users to verify and validate operators' workstations design, from ergonomics

or efficiency point of view, through possibility to insert and simulate human tasks (postures and movements).

According to this point, for our scope, we used Process Simulate as development environment to verify hypothesis expressed above. To do this, we developed the Orbassano robotic assembly Cell on virtual environment: from 2D layout to a complete 3D scene

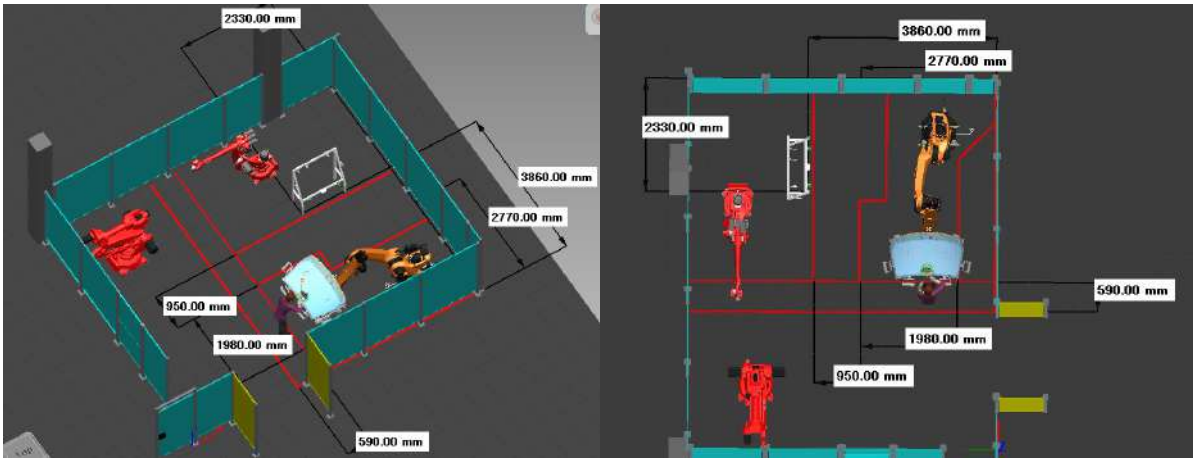


Figure 25: Process Simulate 3D scenario of working cell: ISO View (left), Top View (right)

Developing the assembly cell in virtual environment allows us to plan and verify the Kuka paths from picking the windshield from static support to placing the windshield on assembly position and return.

Using hypothesis of previous phase, we developed five scenarios with five Human mannequin from ANSUR anthropometric database, respecting five ID Group.

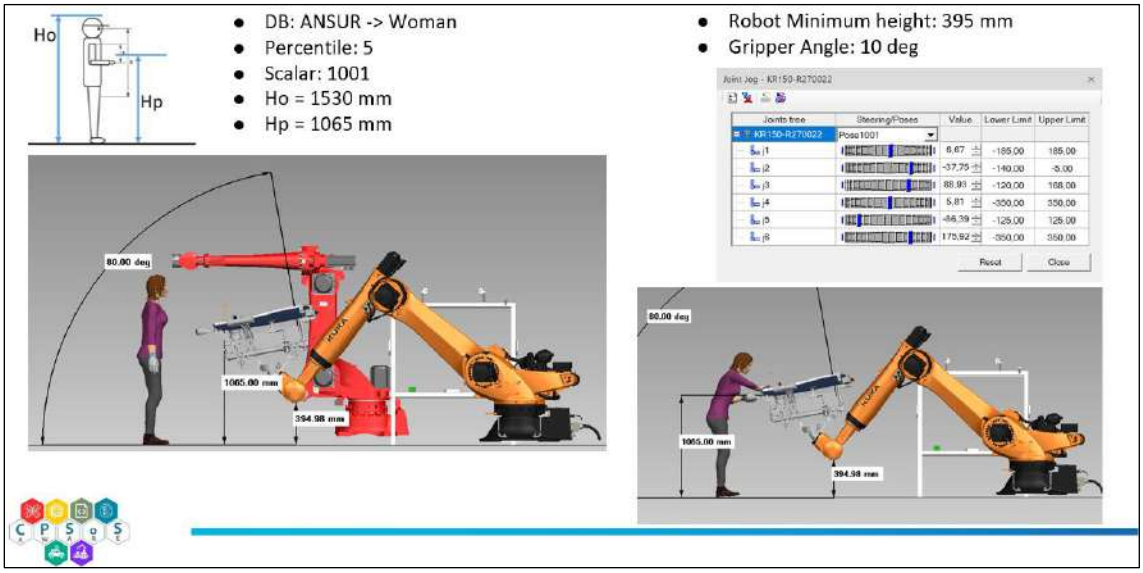


Figure 26: Simulation with Group 1001 Operator

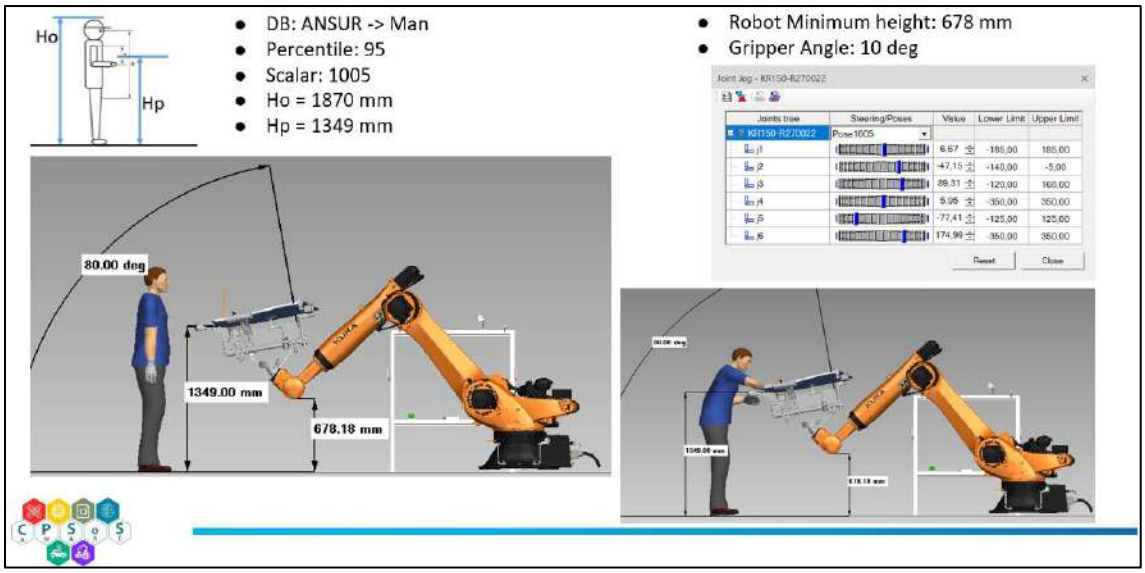


Figure 27: Simulation with Group ID 1005 Operator

As showed in previous images, using the digital environment allowed us to determinate and verify the assembly process postures of human digital representation working on its own strike point zone. Using the Siemens Process Simulate Ergonomics Tool, we also check operators' static posture with RULA index. According to this, using a full developed Kuka robot, with all kinematic blocks set with producer's data-sheet specifics, and using Siemens's simulation engine for robot inverse kinematics, allowed us to verify path and joints position of robot to reach assembly final pose.

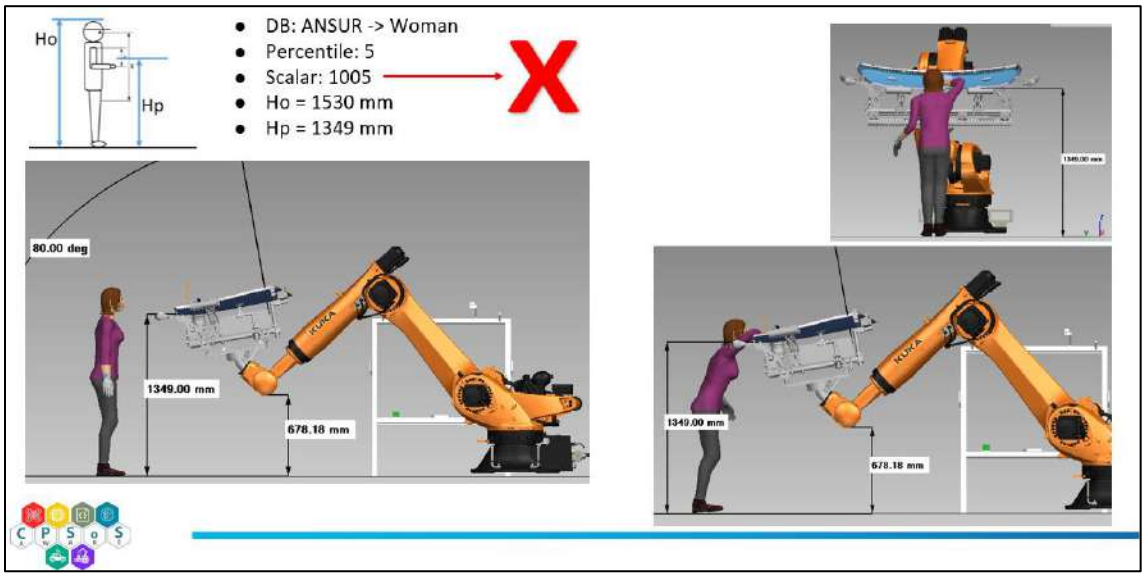


Figure 28: Simulation on an incorrect association between Group ID (group 1005) and Operator (group 1001)

Digital solution gives us the opportunity to verify also what could happen when we are in presence of an wrong design choice or of an incorrect anthropometric tool recognition of operator; in fact, as we can see in previous image, this is an extreme scenario in which we have an operator of belong to group ID 1001 and a wrong positioning of assembly working plan (robot in position useful for an operator belong to group ID 1005). As showed, human mannequin is operating out of own Strike Point zone, at upper edge of Strike Zone, with elbows over the shoulders' height; consequently, RULA index was very high.

3.8.3 Architecture

For the needs of the Pilot's scenarios, 3 stereo cameras are utilized with the purpose to cover the most possible visible area of the working space of the operators. This architecture is used so as to increase the possibility to have, at any time, a good capture of the operator poses (i.e., view of the camera in front of the operator), at least from one of the cameras, while he/she is moving in different directions and positions. Each stereo camera is used to extract the 3D pose landmarks of the operator that will be used then for the height estimation and to calculate the current anthropometric state (based on the RULA score (https://ergo-plus.com/rula-assessment-tool-guide/)), in real-time.

Each RGB camera of each stereo set is used to monitor the human's actions. A pose estimation algorithm is running (locally to the device that the stereo camera is connected) to extract the posture 2D landmarks that are used then for the 3D landmark estimation based on a Direct Linear Transform (DTL) triangulation approach (https://en.wikipedia.org/wiki/Direct_linear_transformation).

Two stereo cameras are directly connected to the computer server which runs all the necessary algorithms (i.e., 2D landmark estimation, 3D landmark calculation, anthropometrics, and ergonomics estimation) and the third one is to a Jetson device. Jetson runs only the 2D landmark estimation algorithm and then sends all the appropriate captured information (i.e., 2D landmarks per frame) via ROS messages to the server (wireless connection). All of them (i.e., the server and the external devices) have to be connected to the same local network.

Other connection topologies could also be used, without the need for further configurations, such as the 3 stereo cameras to be connected in 3 different Jetson devices and then all of them to send the corresponding information to a central server for additional processing.

3.8.4 Equipment

1) ZED camera

Figure 29 presents a representative image of the ZED camera.

Figure 29: ZED stereo camera.

The ZED depth sensor [5] is composed of stereo cameras with a video resolution of 2560 × 1440 pixels (2K) with dual 4 mega-pixel RGB sensors. It consists of two RGB cameras that are at a fixed base distance of 12 cm. This base distance allows the ZED camera to generate a depth image up to 20 m (40 m is the maximum distance, according to the Stereolabs datasheet) (Specifications, 1, & Stereolabs: Orsay). The camera contains a USB video device class-supported USB 3.0 port backwards compatible with USB 2.0. standard.

The ZED sensor uses wide-angle optical lenses with a FOV of 110°, and it can stream an uncompressed video signal at a rate of up to 100 fps in Wide Video Graphics Array (WVGA) format. Table 8 presents more details about ZED camera specifications. Both, right and left video frames are synchronized and streamed as a single uncompressed video frame in the side-by-side format. Furthermore, ZED devices support several software packages, called "wrappers," such as ROS, MATLAB, Python, etc.

Table 8: ZED camera specifications.

Features	ZED
Output Resolution	Side by Side 2x (2208x1242) @15fps
	2x (1920x1080) @30fps
	2x (1280x720) @60fps
	2x (672x376) @100fps
Output Format YUV 4:2:2	Output Format YUV 4:2:2
Field of View	Max. 90° (H) x 60° (V) x 100° (D)
RGB Sensor Type	1/3" 4MP CMOS
Active Array Size	2688x1520 pixels per sensor (4MP)
Focal Length	2.8mm (0.11'') - f/2.0
Shutter	Electronic synchronized rolling shutter
Interface	USB 3.0 - Integrated 1.5m cable

ZED factory calibration parameters are presented in the following Table 9 and Table 10.

Table 9: Intrinsic parameters for each sensor of the ZED camera.

	Left sensor	Right sensor
fx	1399.4500	1400.6700
fy	1399.4500	1400.6700
cx	979.3500	960.6500
cy	535.7540	549.0920
k1	-0.1712	-0.1711
k2	0.0244	0.0239
p1	0.0000	0.0000
p2	0.0000	0.0000
k3	0.0000	0.0000

Table 10: Stereo calibration parameters.

Stereo	
Tx	120.0000
Ty	0.0000
Tz	0.0000
Rx	-0.0035
Ry	0.0131
Rz	-0.0002

Finally, it should be noted that the ZED camera comes with a unique factory calibration file, which can be downloaded automatically. The official recommendation is to use the Stereolabs factory settings (Specifications, 1, & Stereolabs: Orsay), however, the users can also calibrate the ZED sensor with the ZED SDK software package.

2) Operating system of the Server and requirements

The operating system that is used for the PC server is UBUNTU 20.04 which is compatible with ROS 1. All the algorithms are written in Python 3.10.

There are no special requirements regarding the computational power of the computers that will run the algorithms since the implementation can easily run also to low-cost devices (e.g., Jetson).

3) Jetson device

Jetson TX2 Developer Kit is used (Figure 30) including:

- mini-ITX Reference carrier board

- Jetson TX2 Module
- fan and heatsink (pre-assembled)
- 5MP CSI camera module (with Omnivision OV5693)
- WiFi/BT antennas
- USB OTG adapter
- 19VDC Power brick
- AC Power cable

Technical Specifications of the Jetson TX2 module are:

- GPU 256-core NVIDIA Pascal™ GPU architecture with 256 NVIDIA CUDA cores
- CPU Dual-Core NVIDIA Denver 2 64-Bit CPU
- Quad-Core ARM® Cortex®-A57 MPCore
- Memory 8GB 128-bit LPDDR4 Memory
- 1866 MHx - 59.7 GB/s
- Storage 32GB eMMC 5.1
- Power 7.5W / 15W

Figure 30: Jetson TX2 in use.

3.8.5 2D landmarks estimation

The landmarks of the operator pose could be identified by any ordinary RGB camera. The recognition algorithm was designed according to the landmarks provided by MediaPipe. MediaPipe Pose is a machine-learning pipeline that predicts high-fidelity body pose tracking from an RGB video that utilized BlazePose and MobileNetv2 CNN architecture. BlazePose is a machine-learning model that consists of a detector and an estimator developed by Google that can compute the coordinates of 33 skeletal points, as presented in Figure 31.

The MediaPipe algorithm is a machine-learning pipeline using a two-step detector. The pipeline first locates the human within the frame, and then a tracker uses the pose of interest cropped frame as input to predict the pose landmarks and segmentation masks.

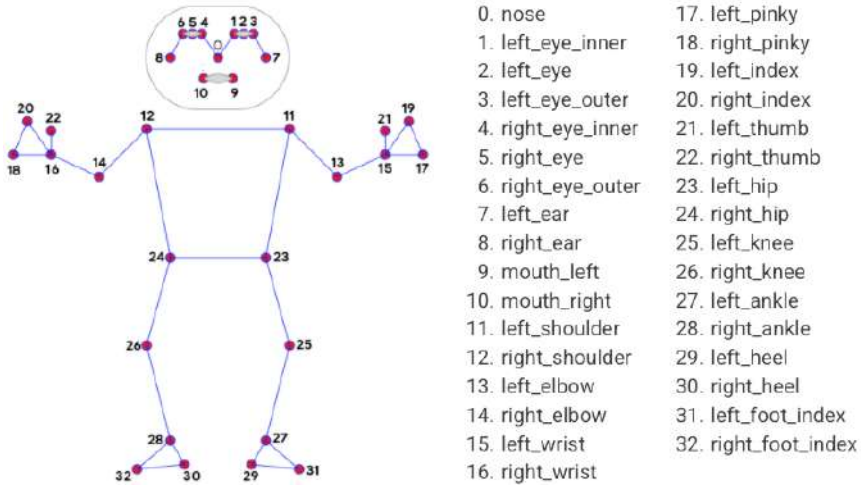


Figure 31: Pose landmarks with 33 joints (Valentin Bazarevsky).

From the above pose model, we use only 19 landmarks. More specifically, the used points are: 0 nose, 11 left shoulder, 12 right shoulder, 13 left elbow, 14 right elbow, 15 left wrist, 16 right wrist, 17 left pinky, 18 right pinky, 19 left index, 20 right index, 21 left thumb, 22 right thumb, 23 left hip, 24 right hip, 25 left knee, 26 right knee, 27 left ankle, 28 right ankle. The used simplified model consisting of the 19 landmarks is presented in Figure 32.

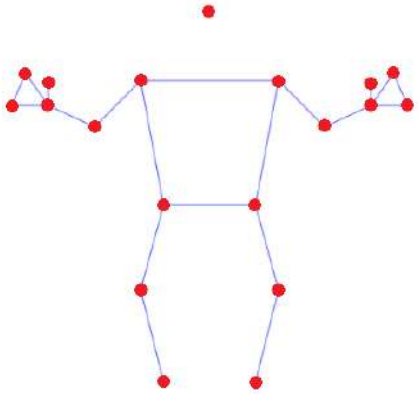


Figure 32: Simplified pose model.

3.8.6 3D landmarks calculation

1) Single Camera Calibration:

For the calibration of each camera separately, we start using the same checkerboard pattern that however has to be visible to all cameras at the same time to perform the stereo calibration later. We have also to make sure that all of the used cameras have the same resolution analysis and that they capture images at the same fps so that all captured frames are synchronized.

To mention here that each camera is considered as a simple pinhole model having a coordinate system at the centre of the camera lens. Light rays from a distinct point in the camera's view travel through the camera lens and converge into a single point in the camera (Figure 33). In the central projection camera

model, this vector intersects an image plane at a distance f, which represents the focal length, along the Z axis (Vaishnav Chunduru).

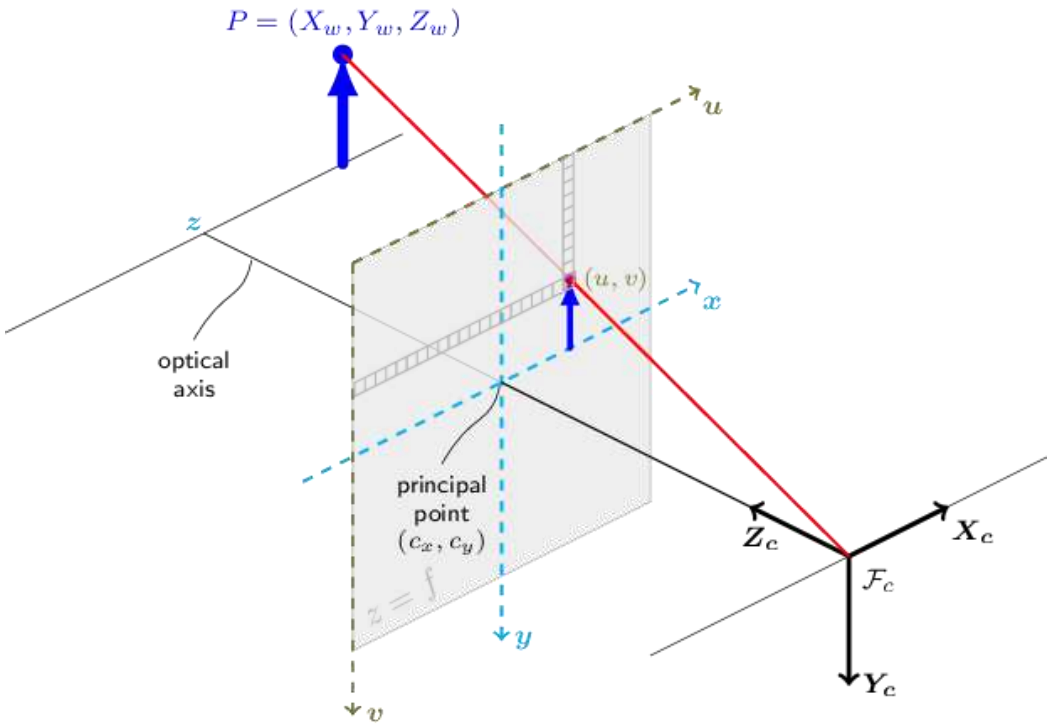


Figure 33: Pinhole camera model example (https://github.com/TemugeB/python_stereo_camera_calibrate).

In Figure 34, we present the form and the specification (i.e., number of rows, columns, and dimensions of the squares) of the checkerboard that is used for the calibration.

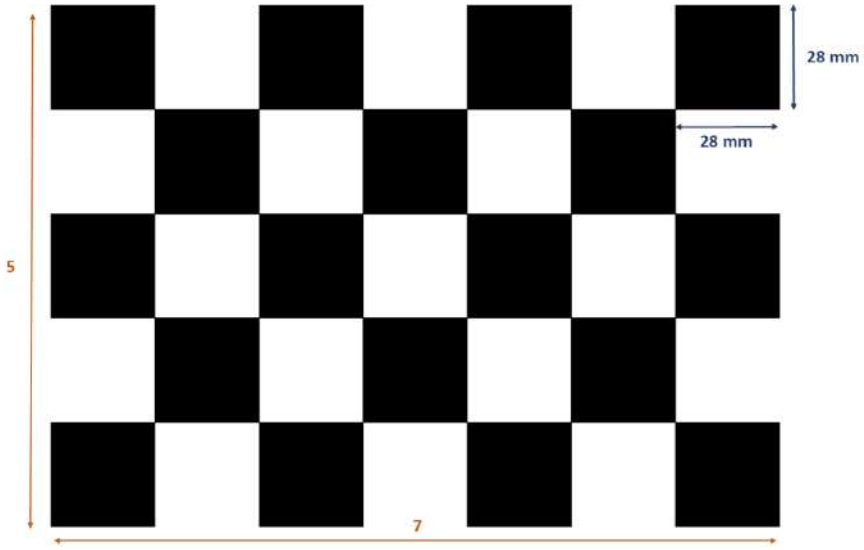


Figure 34: Specification of the checkerboard that is used for the calibration.

Figure 35 shows an example of two synchronized frames, captured by the left and right RGB camera of the ZED that are used for the calibration (single and stereo).

Figure 35: Example of the left and right camera calibration.

2) Stereo Calibration:

Let's start defining each camera of the general integrated system as C_i ($\forall i = 1, \cdots, 6$). The single-camera calibration step of camera C_i returns the rotation matrix $\mathbf{R}_i$, the translation $\mathbf{T}_i$ matrix and the distortion coefficients for each one of the single cameras. However, the $\mathbf{R}$ and $\mathbf{T}$ matrices alone are not enough to triangulate the 3D points. So, in this case, we have to first define a world space origin point and orientation.

For the stereo-calibration procedure, we need two sets of cameras. For the sake of simplicity, let's define these two cameras as C_1 and C_2. Notice here that the same approach can be used for the stereo calibration of all other cameras.

The rotation matrix **R** of a stereo-calibrated system is the coordinate rotation matrix to go from the C_1 coordinate system to the C_2 coordinate system. The translation vector **T** is also the location of C_2 from C_1. So, the aforementioned approach does not contain any information about the world coordinate rotation matrix and translation vector. They only provide information related to the C_2 position and rotation concerning C_1. Nevertheless, we can overcome this problem, obtaining world coordinates to C_2 rotation and translation, by calculating the following equations:

$$\boldsymbol{R2} = \boldsymbol{R}\,\boldsymbol{R1}$$

$$\boldsymbol{T2} = \boldsymbol{R}\,\boldsymbol{T1} + \boldsymbol{T}$$

Next, the most effortless way to take into account the world coordinates is to simply choose C_1 position as world space origin (x=0, y=0, z=0). Therefore, the world origin to C_1 rotation is the identity matrix and translation is a zeros vector. Then, the **R** matrix and the **T** vector that have been estimated from the previous stereo-calibration step becomes the rotation and translation from world origin to C_2. Practically, this means that the 3D triangulated points will be in a space with respect to the coordinate system of C_1's lens. In other words, we simply overlap world coordinates with the coordinates of the C_1 camera. This means $\mathbf{R_1}$ = eye(3), $\mathbf{T_1}$ = zeros(3) and $\mathbf{R_2}$ = **R**, $\mathbf{T_2}$ = **T**. Therefore, all triangulated 3D points are measured from the C_1 camera position in the world.

3) Direct Linear Transform:

The 3D point triangulation from multiple cameras is performed via the Direct Linear Transform (DLT) method. DLT is a general approach designed to solve systems of equations as the following [6]:

$$\lambda x = \boldsymbol{A} y$$

This type of equation frequently appears in projective geometry, showing that the DLT can be used to estimate the matrix P.

$$\lambda \begin{pmatrix} x_1 \\ x_2 \\ 1 \end{pmatrix} = K\,[\boldsymbol{R}\,\boldsymbol{T}] \begin{pmatrix} X_1 \\ X_2 \\ X_3 \\ 1 \end{pmatrix} <=> \lambda x = \boldsymbol{P} X$$

We start by formulating a homogeneous linear system of equations and then we try to solve it by finding an approximate null space. To do that, we first have to express the matrix **P** in terms of row vectors:

$$\boldsymbol{P} = \begin{bmatrix} p_1^T \\ p_1^T \\ p_1^T \end{bmatrix}$$

Then, the camera equation can be written as:

$$X_i^T p_1 - \lambda_i x_i = 0$$
$$X_i^T p_2 - \lambda_i y_i = 0$$
$$X_i^T p_3 - \lambda_i = 0$$

which is equivalent to:

$$\begin{bmatrix} X_i^T & 0 & 0 & -x_i \\ 0 & X_i^T & 0 & -y_i \\ 0 & 0 & X_i^T & -1 \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_i \end{pmatrix} = \begin{pmatrix} \mathbf{0} \\ \mathbf{0} \\ \mathbf{0} \end{pmatrix}$$

where $\mathbf{0} \in \Re^{1x4}$.

Collecting all the projection equations of all the measured data in the same matrix, then we have:

$$\begin{bmatrix} X_1^T & 0 & 0 & -x_1 & 0 & 0 & \cdots \\ 0 & X_1^T & 0 & -y_1 & 0 & 0 & \cdots \\ 0 & 0 & X_1^T & -1 & 0 & 0 & \cdots \\ X_2^T & 0 & 0 & 0 & -x_2 & 0 & \cdots \\ 0 & X_2^T & 0 & 0 & -y_2 & 0 & \cdots \\ 0 & 0 & X_2^T & 0 & -1 & 0 & \cdots \\ X_3^T & 0 & 0 & 0 & 0 & -x_3 & \cdots \\ 0 & X_3^T & 0 & 0 & 0 & -y_3 & \cdots \\ 0 & 0 & X_3^T & 0 & 0 & -1 & \cdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots \end{bmatrix} \begin{pmatrix} p_1 \\ p_2 \\ p_3 \\ \lambda_1 \\ \lambda_2 \\ \lambda_3 \\ \vdots \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix}$$

As shown from the previous system, we want to obtain the non-trivial solution of an equation of the form A**x** = 0. However, in the real world, there is the possibility of the presence of noise, so we have to write the equation as A**x** = **w**, taking into consideration the noise, and we solve for **x** such that **w** is minimized. Assuming that the **SVD** is used to determine the **SVD** decomposition of A.

3.8.7 Operator's height estimation

Once the coordinates of the 3D landmarks have been estimated. Two different approaches can be used, for the estimation of the operator's height:

- Geodesic distance between nose to ankles of the operator + 15 cm.
- Geodesic distance between the right index finger to the left index finger.

The experiments show that the second approach is less prone to errors and this is the one that is finally used during the pilot.

Based on the anthropometrics information, the robot control receives the control information that the wanted set of actions is carried out. A total of 5 different classes are used based on the operators' height, leading to 5 adaptable robot responses (adaptation scenarios) to allow the human to work in an optimal ergonomics state. The algorithmic framework identifies the operators' class, based on their height, and the corresponding selected scenario configures the cooperating robot's movement parameters and relative trajectories to adapt to the environment in a way that is ergonomically most comfortable for the interacting user. More specifically:

- Class 1 - Operators with height < 1544
- Class 2 - Operators' height is between 1544 and 1641
- Class 3 - Operators' height is between 1641 and 1727
- Class 4 - Operators' height is between 1727 and 1829
- Class 5 - Operators are taller than 1829

The use cases of this Pilot take place by taking into account the use of different users with different characteristics (height, sex, experience). More specifically, we present the results of different operators (male and female) with a variety of heights (1.68 – 1.91) and levels of experience (none, less or more experienced users). The following figures (Figure 36, Figure 39, Figure 42 & Figure 43) present examples of the operator's class estimation based on the height of the user, for different users with different characteristics. Each one of these figures represents a screenshot of the system's output while it runs in real-time, capturing the operator's pose for the estimation of the 2D landmarks (i.e., bottom-left display), calculating the 3D landmarks (i.e., top-left display), as well as estimate and send via ROS messages the class of the operator's height (i.e., top-right display), the ergonomics state (i.e., top-middle display) and the OSM (i.e., bottom-right display).

3.8.8 ROS messages

The server sends ROS messages to communicate with the other external devices. The class of operator's height is sent to the robot every 6 seconds (60 frames with a rate of 10 frames per second (fps)) through the ROS topics \operator while the ergonomics state of the operator is sent to the HoloLens through the ROS topics \ergonomics, correspondingly.

Table 11 presents the form of the ROS message that is sent based on the operator's height class.

Table 11: ROS message based on the operator's height.

Operator's height (in mm)	ROS message
height < 1544	1001
1544 <= height < 1641	1002
1641 <= height < 1727	1003
1727 <= height < 1829	1004
height > 1829	1005

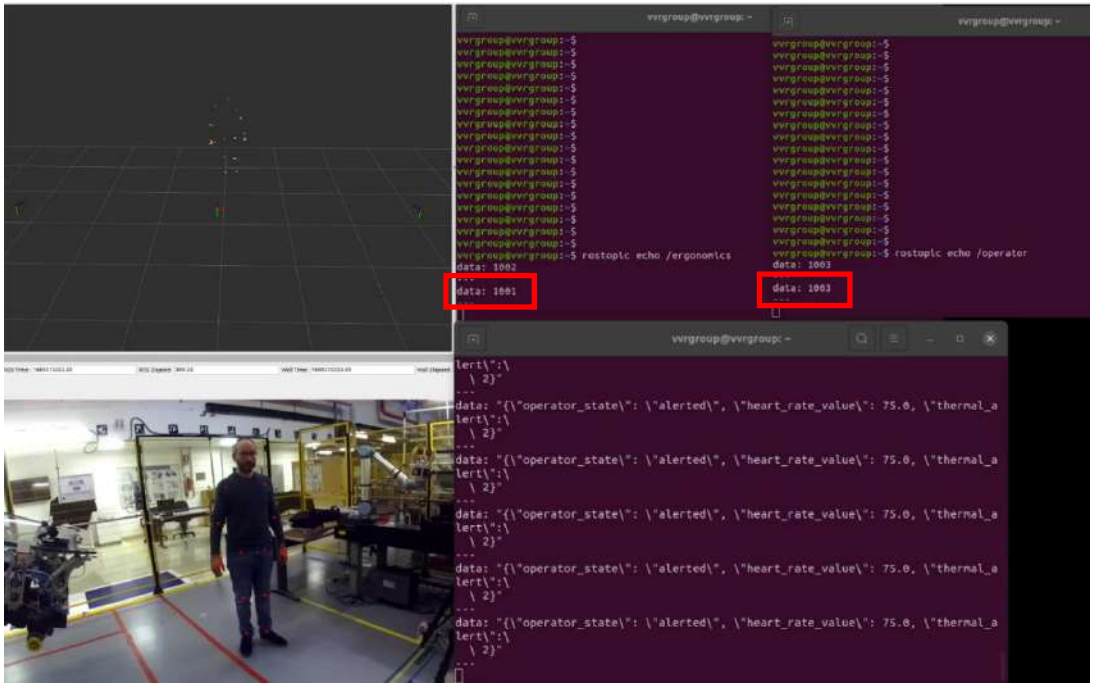


Figure 36: Operator's class estimation based on the height of the Operator 1 (Class 3 - 1641 <= height < 1727).

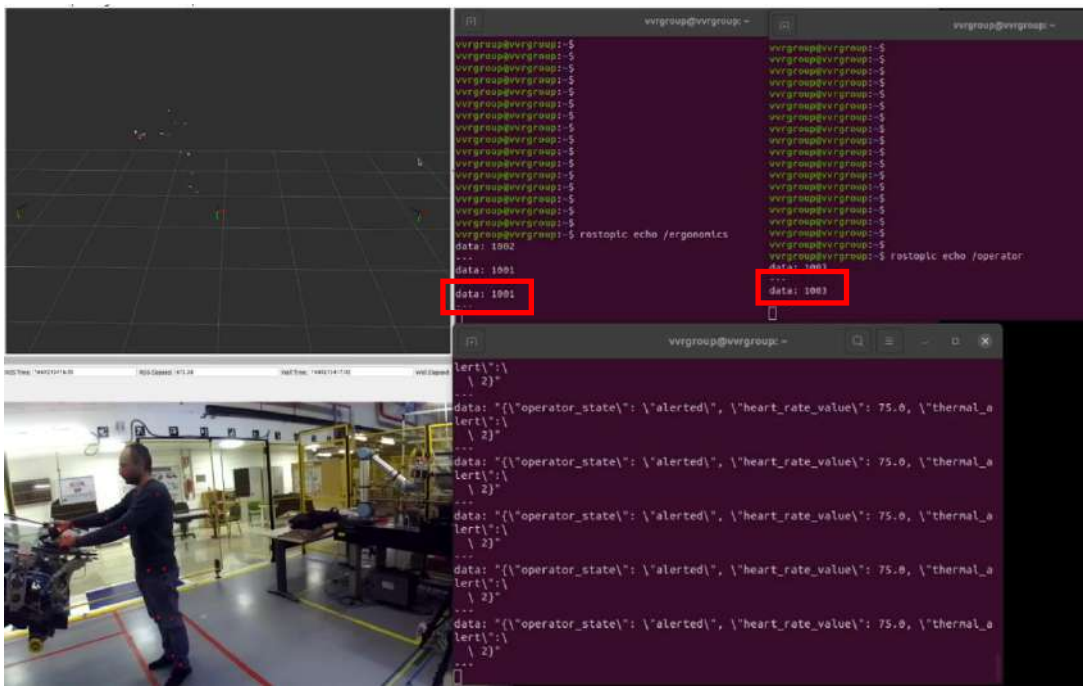


Figure 37: Ergonomic state of the Operator 1 (Safe pose).

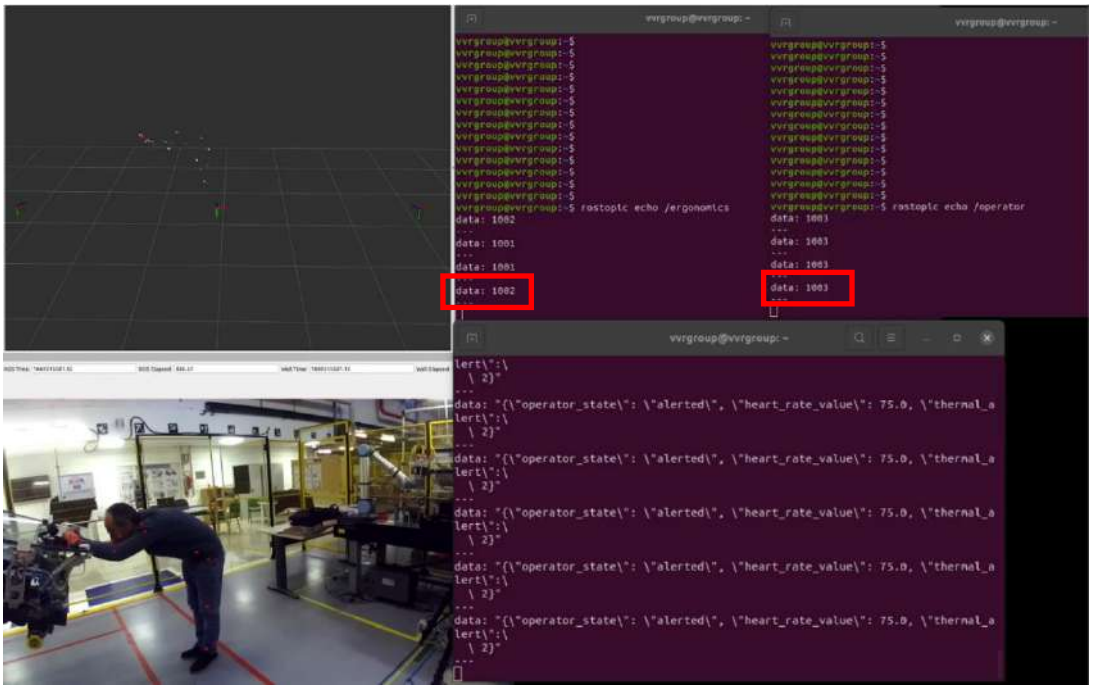


Figure 38: Ergonomic state of the Operator 1 (Not safe pose).

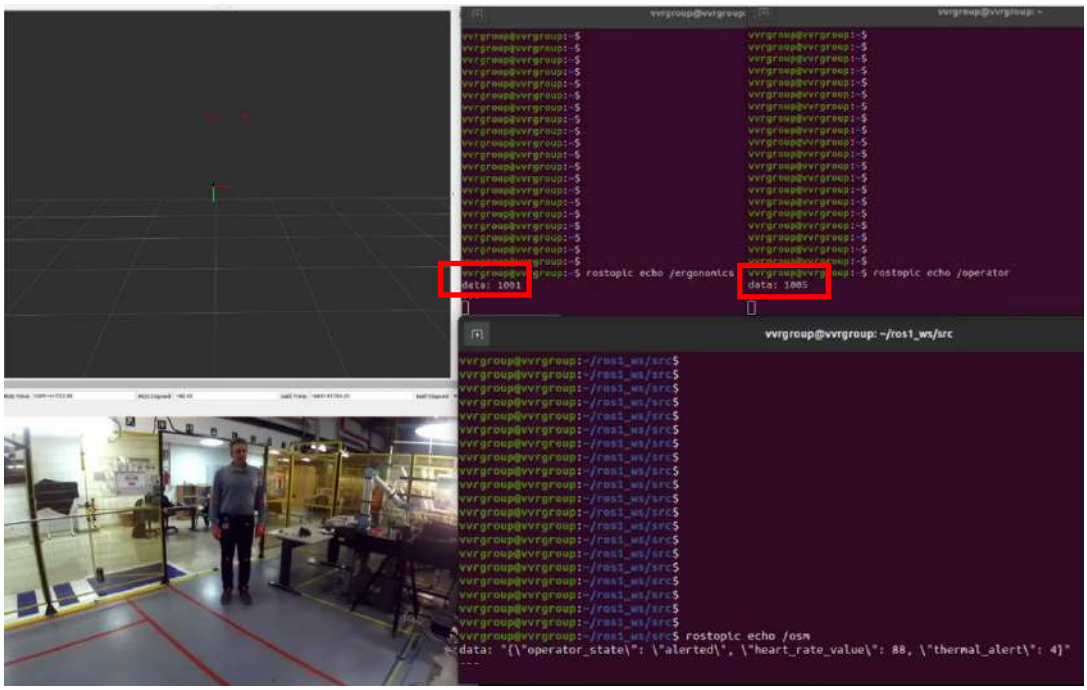


Figure 39: Operator's class estimation based on the height of the Operator 2 (Class 5 - height > 1829).

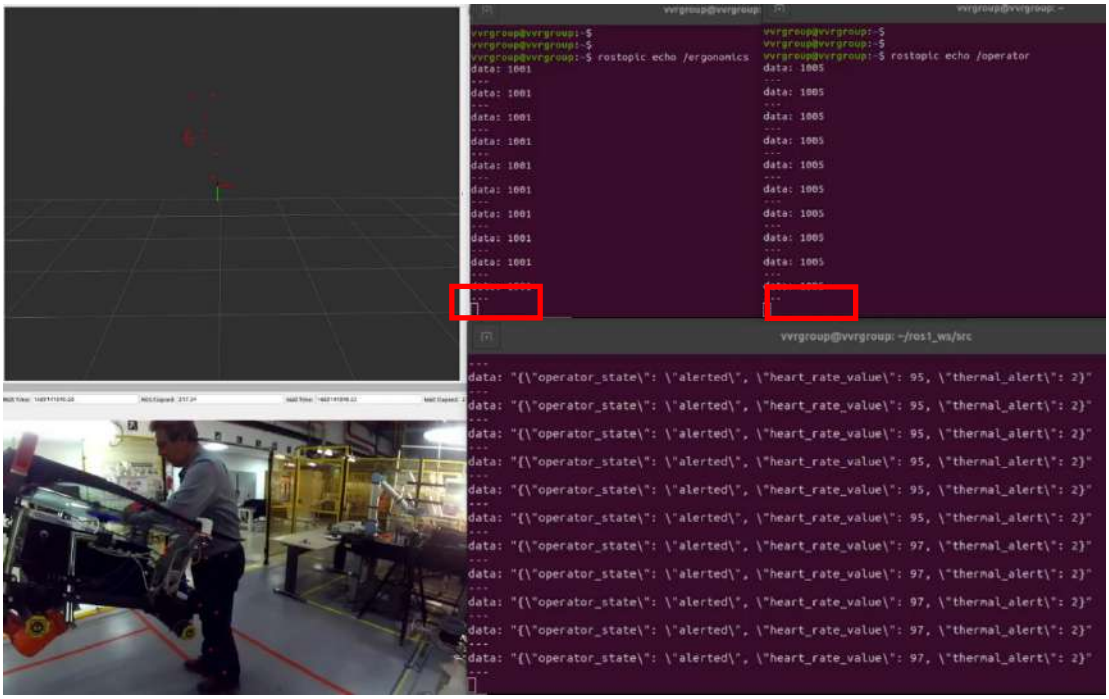


Figure 40: Ergonomic state of the Operator 2 (Safe pose).

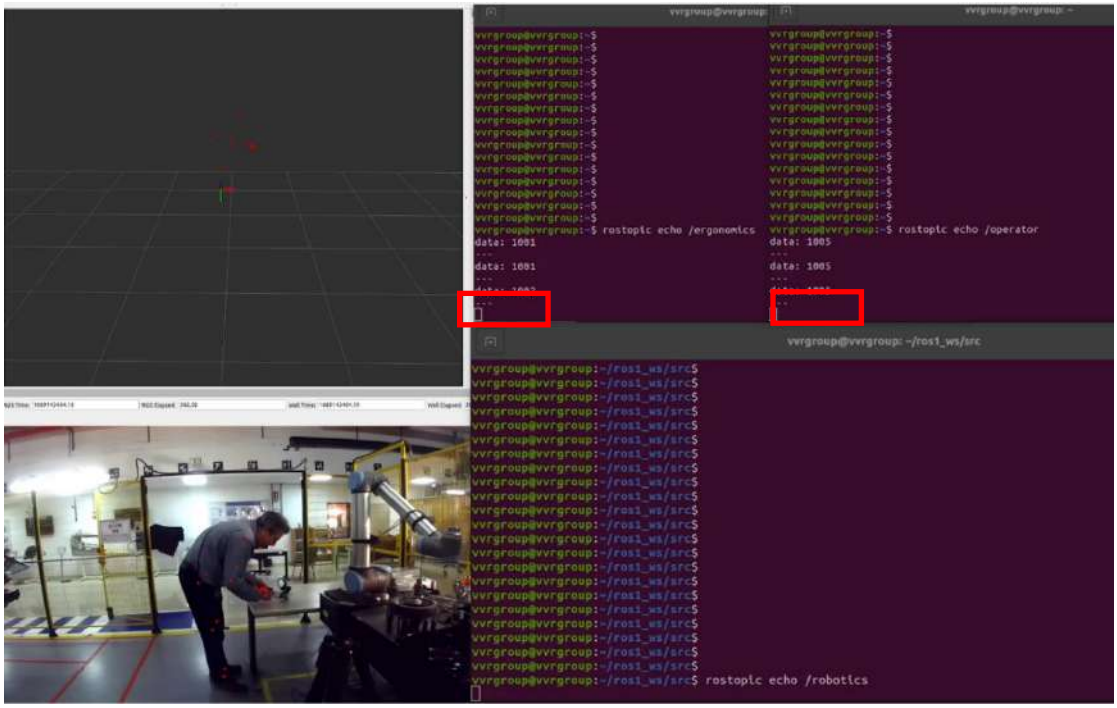


Figure 41: Ergonomic state of the Operator 2 (Not safe pose).

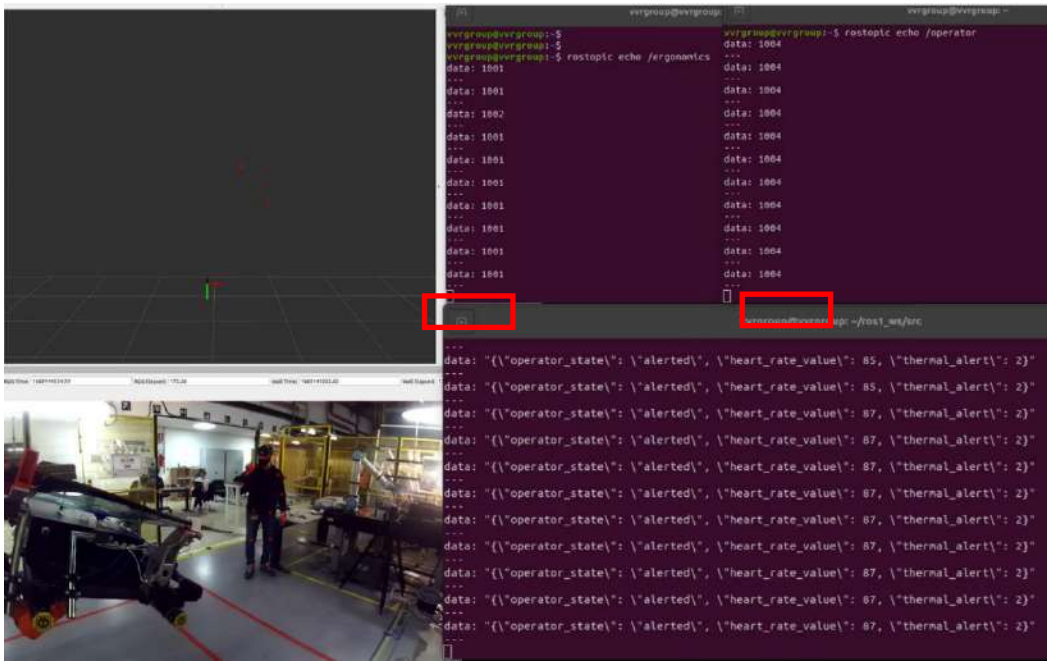


Figure 42: Operator's class estimation based on the height of the Operator 3 using Hololens (Class 4 - 1727 <= height < 1829).

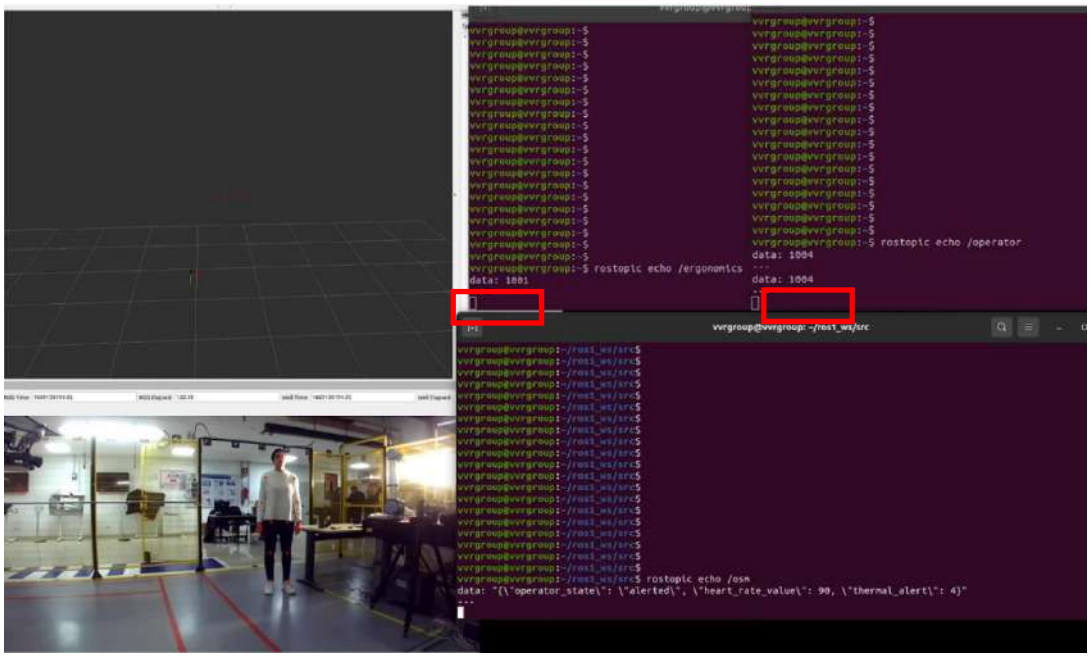


Figure 43: Operator's class estimation based on the height of the Operator 4 (Female) (Class 4 - 1727 <= height < 1829).

3.8.9 Ergonomics, real time RULA evaluation

This scenario executed as part of pillar 2 demonstrations involves real-time RULA score estimation and relevant warnings regarding the pose of the operator. Additionally, a real-time joint angle estimation is performed to estimate the RULA score using the general rules as presented in Figure 44.

Based on the ergonomic information (i.e., regarding the estimation of the RULA score value), the appropriate warning messages are sent to the operators, informing them if their working pose is ergonomically correct or not.

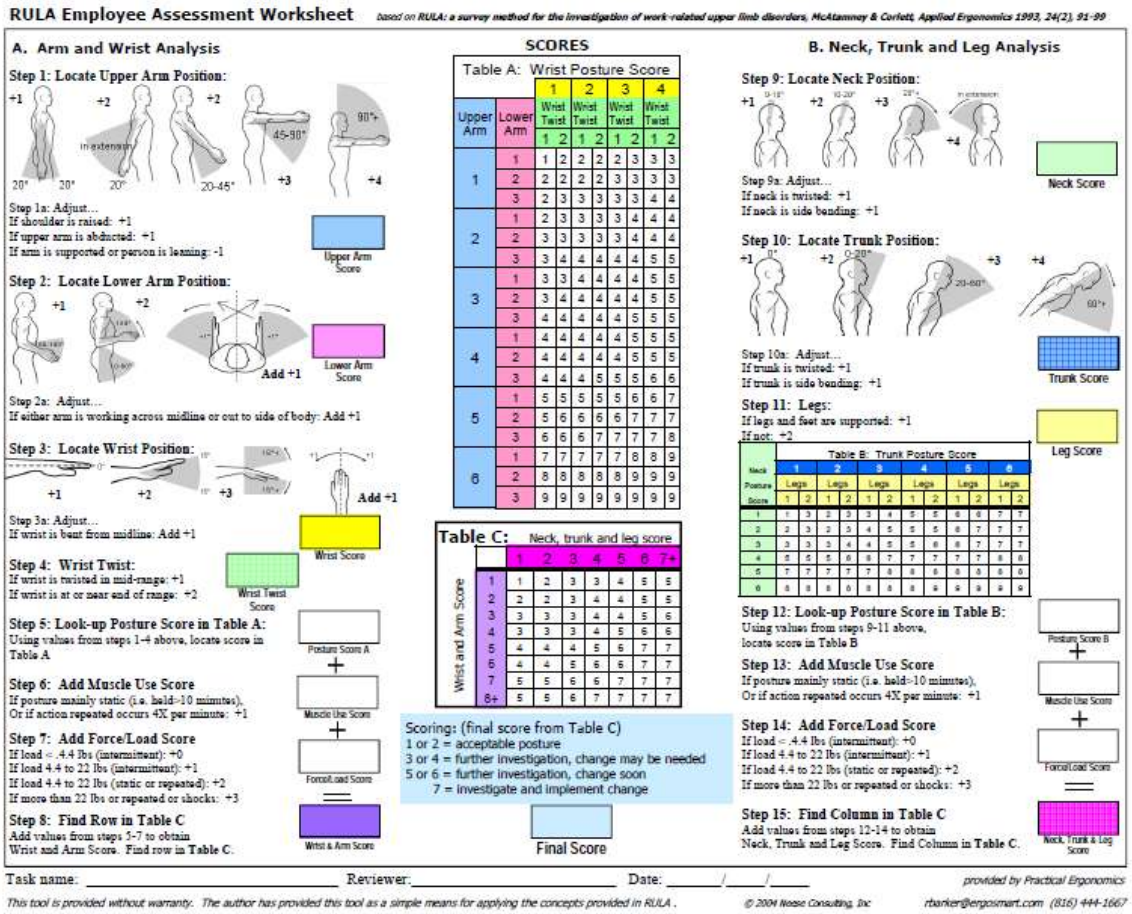


Figure 44: Rapid Upper Limp Assessment (RULA) estimation [7].

The following figures (Figure 45 - Figure 50) illustrate examples showing how the joint angles (estimated by the corresponding 3D landmarks that are calculated based on the output of the described stereo-camera system) are strained during different actions, using also a heatmap that visualizes the level of strength of the joint angles (deep blue colour represents non-stress at all while deep red colour represents very high stress).

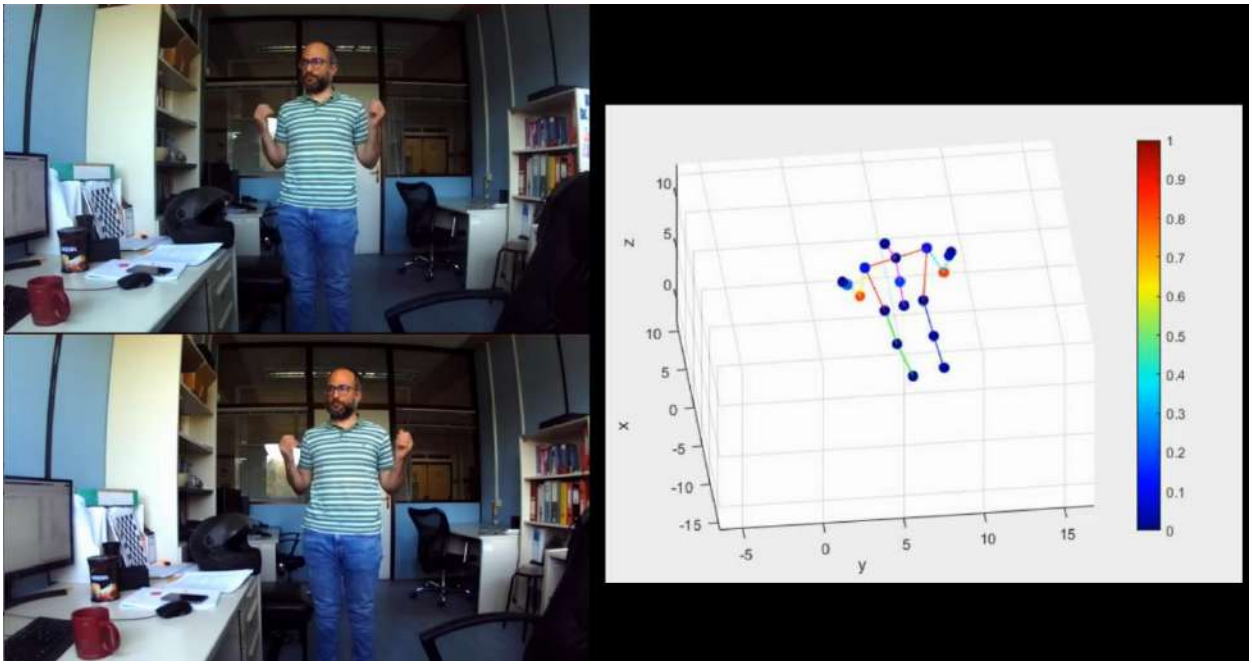


Figure 45 Visual example of the estimated joint angle stress of the user's elbows.

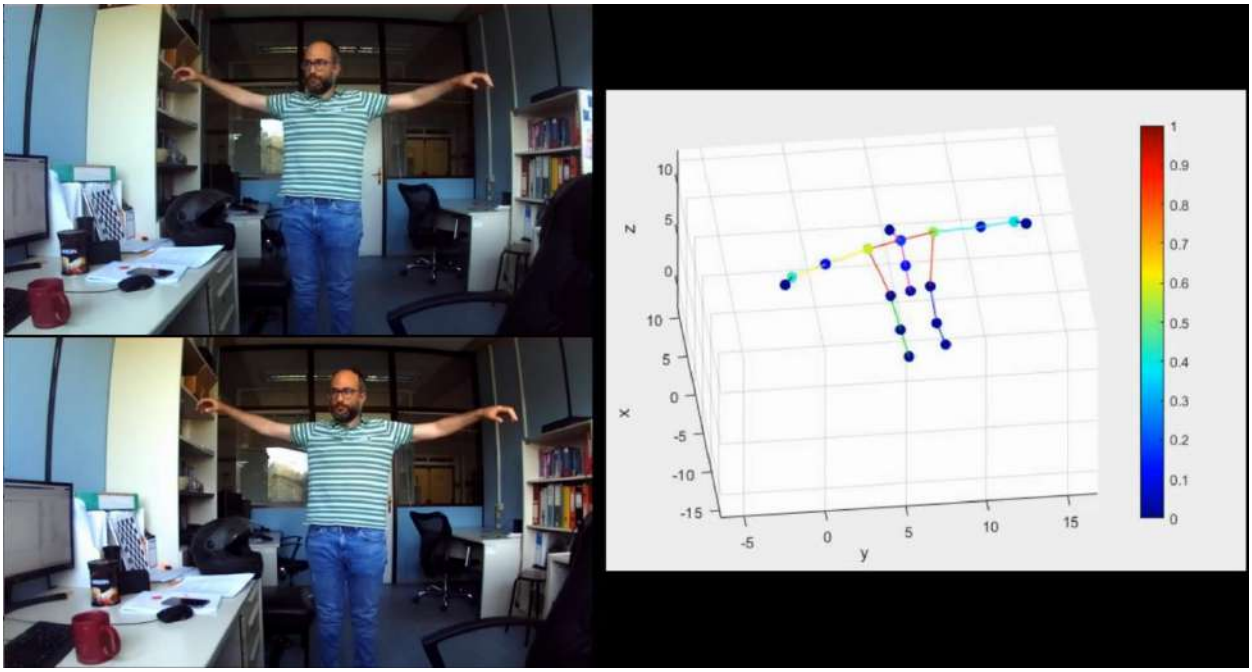


Figure 46: Visual example of the estimated joint angle stress of the user's shoulders.

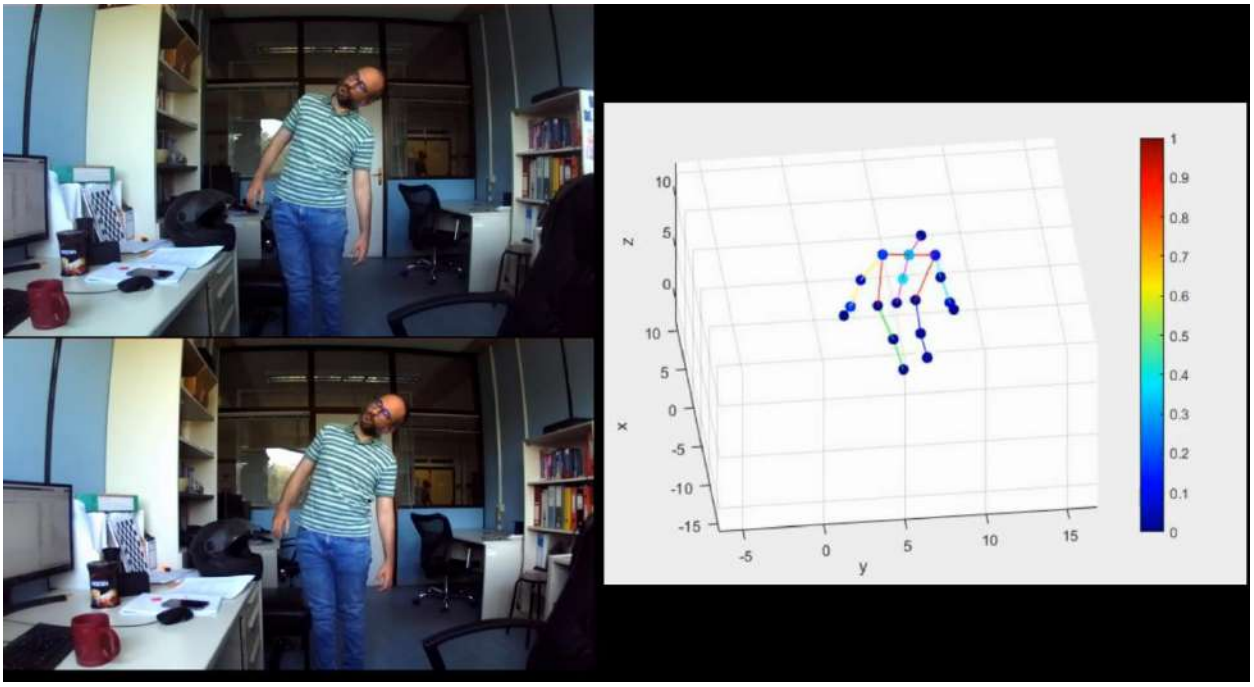


Figure 47: Visual example of the estimated joint angle stress of the vertebras of the user's back.

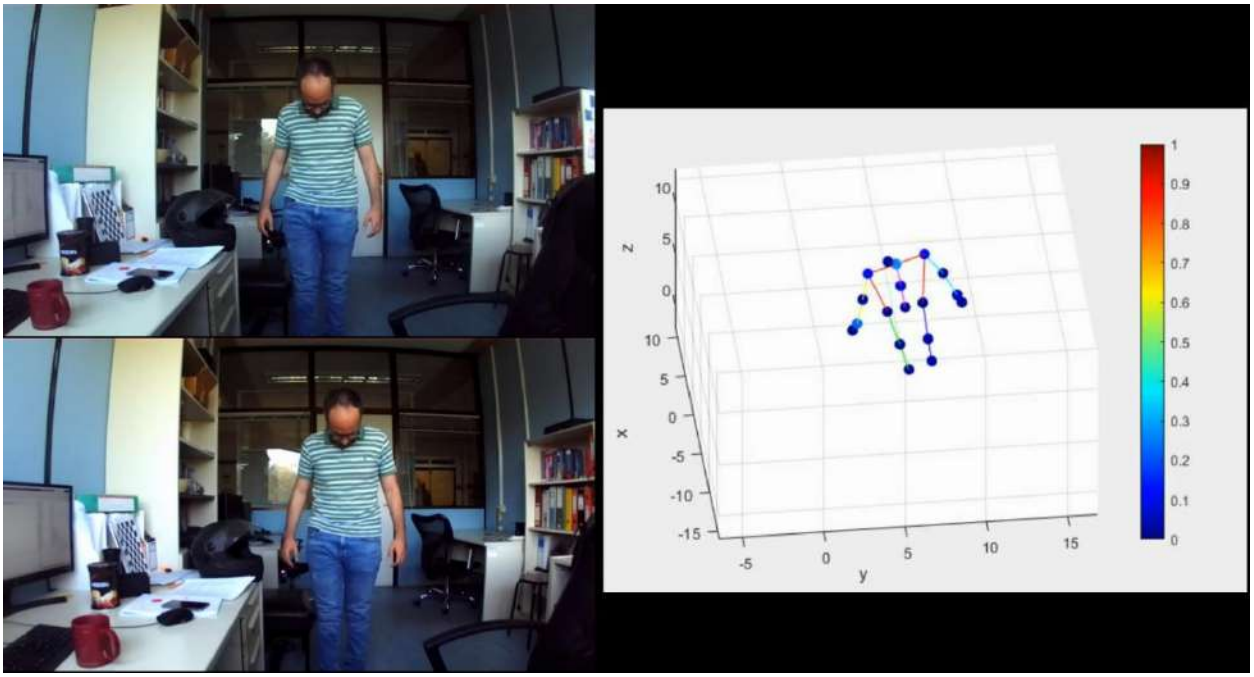


Figure 48: Visual example of the estimated joint angle stress of the user's neck.

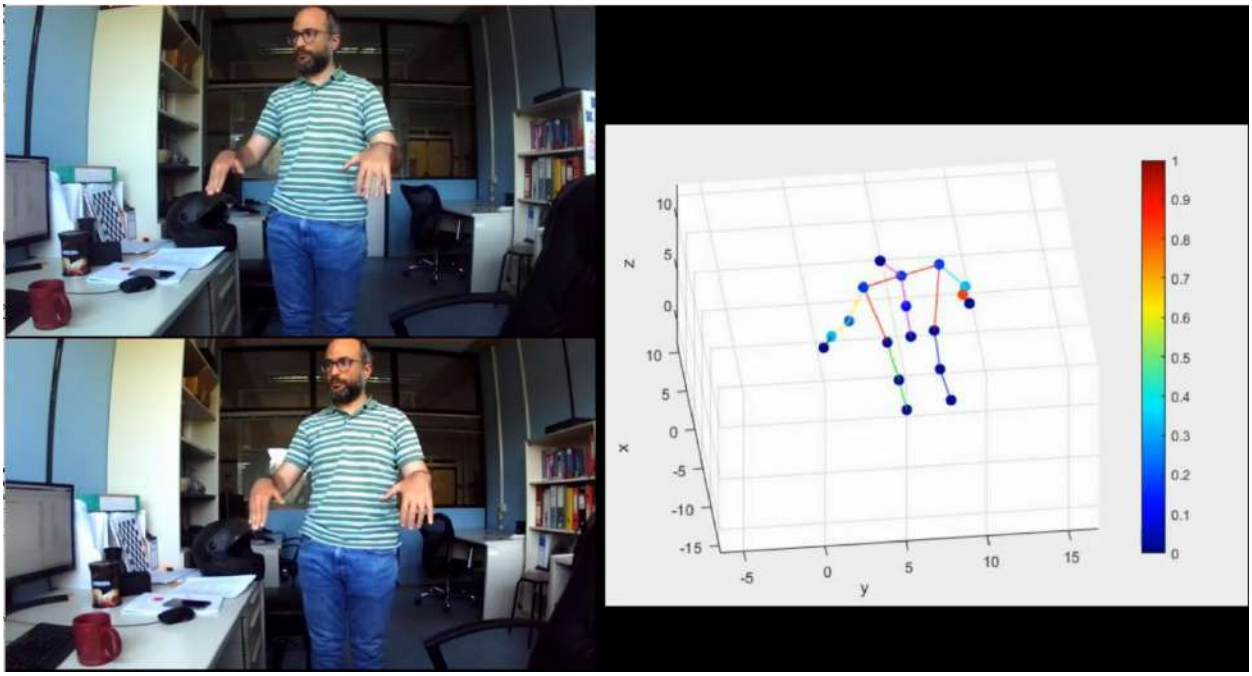


Figure 49: Visual example of the estimated joint angle stress of the user's wrists (in up and down direction) and arms.

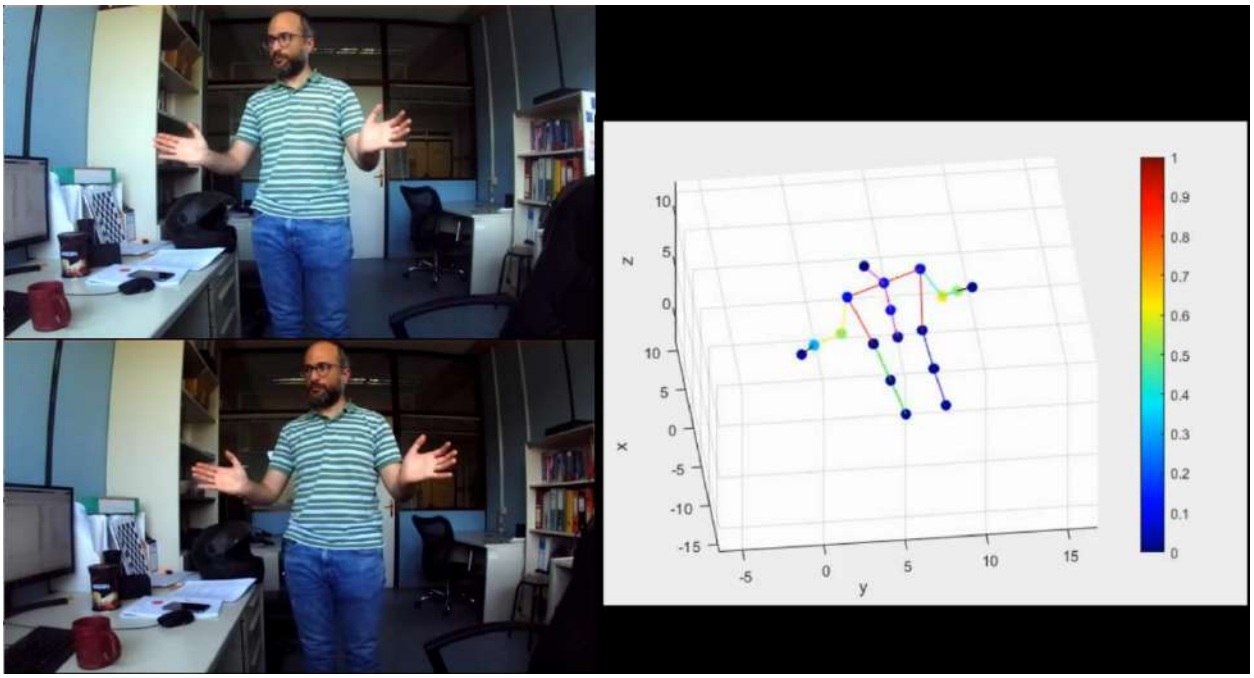


Figure 50: Visual example of the estimated joint angle stress of the user's wrists (in right and left direction) and arms.

Table 12 presents the ROS message, which is sent to Hololens, regarding the ergonomics state of the operator.

Table 12: ROS message based on the ergonomics state of the operator.

Ergonomics state	ROS message
Safe sequence of poses	1001
Harmful sequence of poses	1002

Figure 37, Figure 38, Figure 40 & Figure 41 present examples of the ergonomic state estimation of the operator (i.e., Safe and Not safe pose), for different users with different characteristics, in ROS message format for the communication with the HoloLens devise.

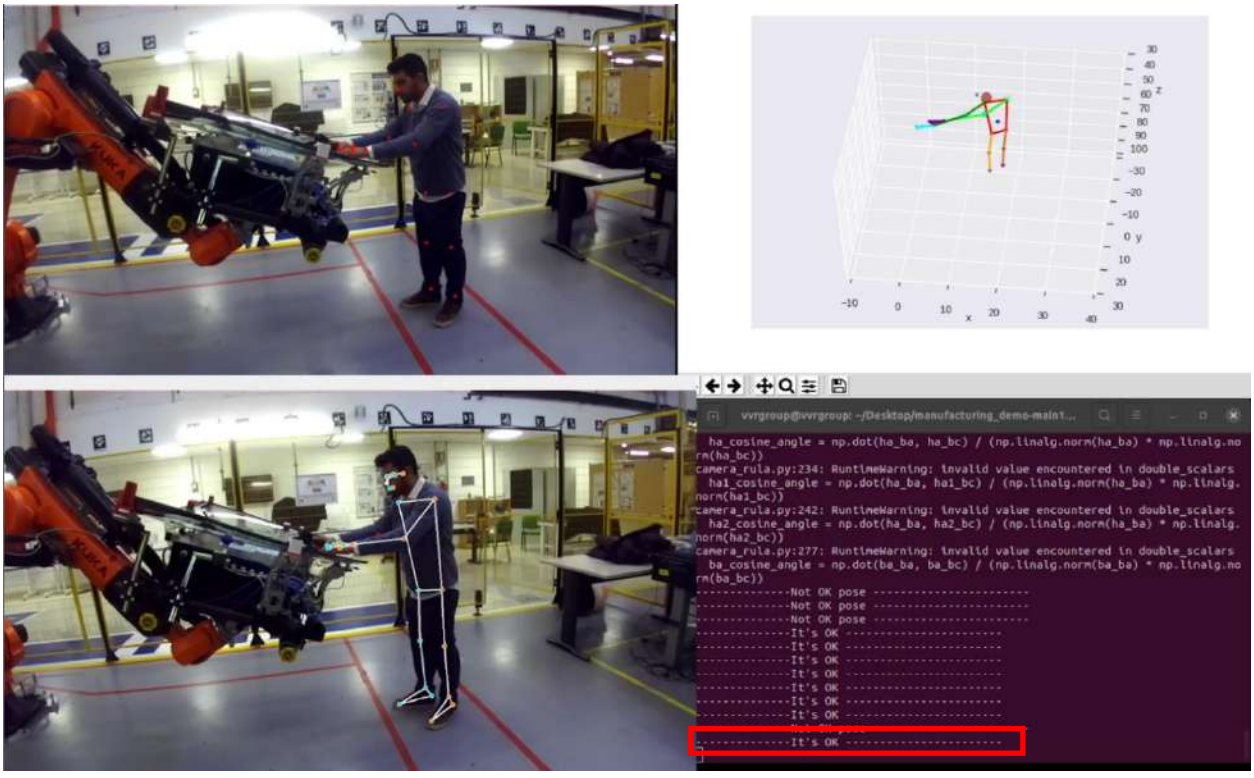


Figure 51: Safe pose identification based on the RULA score value.

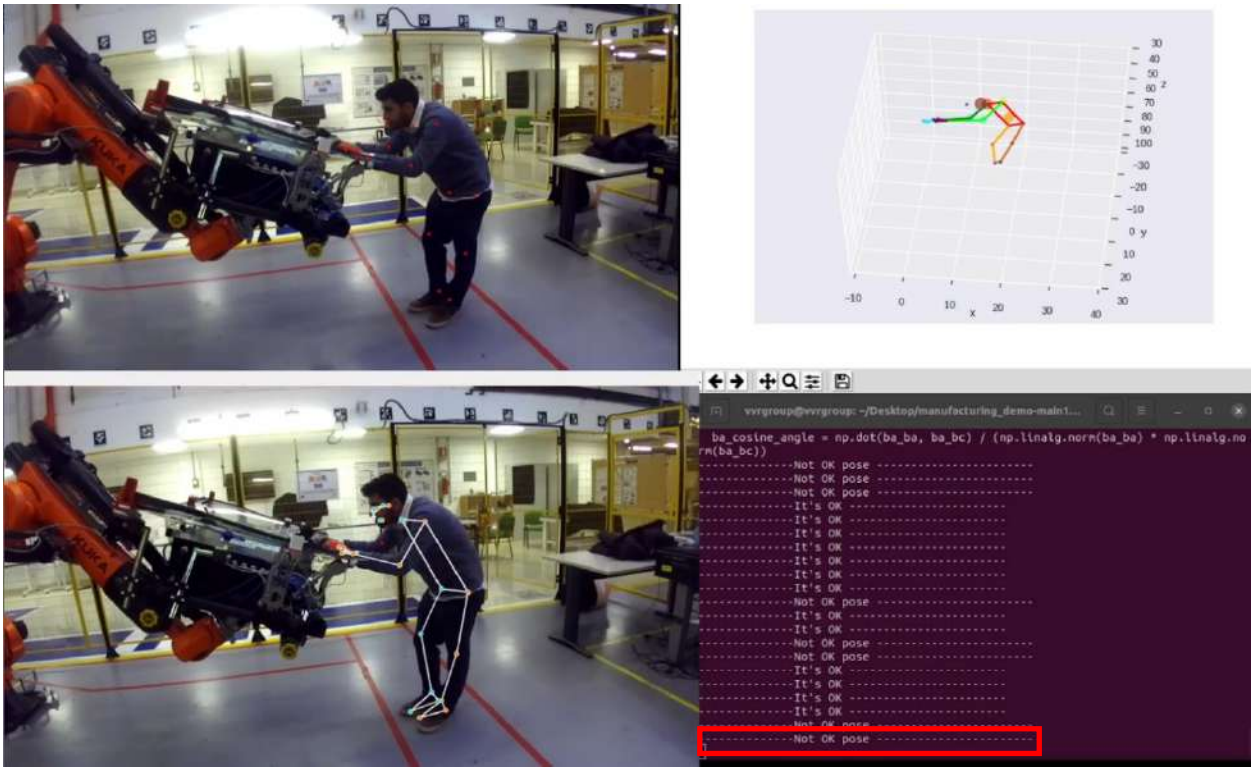


Figure 52: Example of warning for wrong pose of the operator.

Figure 51 and Figure 52 present screenshots of the integrated system that captures in real time the operator's pose using two stereo-calibrated RGB cameras, estimate the 2D landmarks of the operator, then calculates the corresponding 3D landmarks and uses these landmarks to estimate and visualize the joint angles and finally to find the RULA score, per frame or per a sequence of 60 frames (6 seconds of action). More specifically, Figure 51 illustrates an example of a safe pose identification while Figure 52 illustrates a case of detection of a wrong pose that can affect the operator's wellbeing.

3.8.10 Cooperative localization for landmarks selection in multicamera systems

Considering the previous scenarios of RULA estimation, it is highly probable that when the operator is moving then multicamera systems are required in order to measure the RULA score from different viewpoints. However, in those cases the challenge is to choose the best possible pair of camera and landmarks configuration so as to have a single RULA score. Motivated by the cooperative localization module TC3.3.1 designed and documented in deliverable D3.3 for automotive applications, we exploit a similar concept in order to perform efficient multimodal fusion and determine the "best" pair of camera and detected landmarks of operator. To be more specific, considering a graph topology (shown in Figure 53) of four cameras (C_i) which detect in total four landmarks (l_i) we perform the following steps: 1. Encode the spatial relationship of cameras and landmarks via Graph Laplacian Processing (see D3.3 for automotive pillar), 2. Construct Laplacian matrix of connectivity graph, indicating which landmark is related to which camera, 3. Compute relative distance and angle between cameras and landmarks, 4. As anchor positions,

use the known location of cameras and the average position of landmarks using the estimated positions by each one of the cameras, 5. Relative measurements and anchors are used to formulate differential coordinates, 6. Using Laplacian matrix and differential coordinates, re-estimate landmarks' positions via linear least-squares minimization. Following this methodology, we end up with a camera-and-landmarks configuration very close or even better than the optimal one.

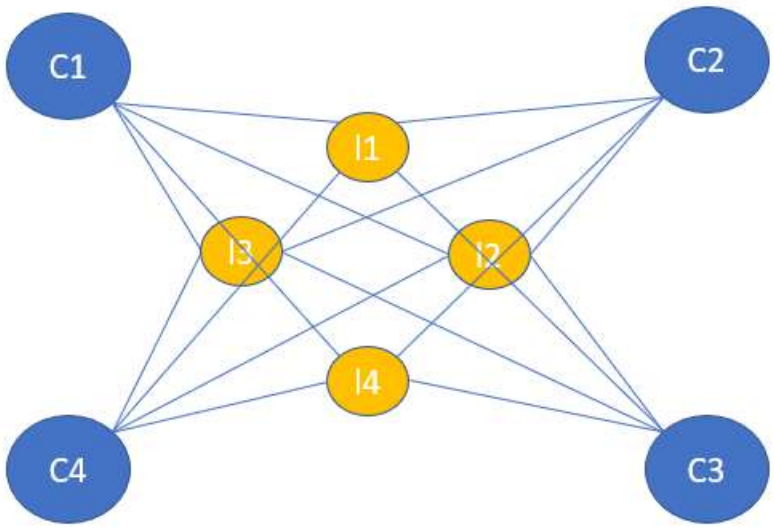


Figure 53: Indicative example of graph topology for cameras and landmarks

Table 13 summarizes the main outcomes from a testing scenario of 250 frames in Unity environment where three cameras detect 11 landmarks of the operator. More specifically, we measured the RMSE over simulation horizon for each one of the landmarks using the cameras and the proposed module. As we can see, in most of the cases Graph Laplacian Processing based Fusion achieved results very close to the optimal camera, while in two cases (landmarks 6 and 11), landmarks were detected slightly better than the optimal camera.

Table 13: RMSE of landmarks using multicamera system and Graph Laplacian Processing

	L1	**L2**	**L3**	**L4**	**L5**	**L6**	**L7**	**L8**	**L9**	**L10**	**L11**	**L12**
CAM1	0.34	0.38	0.19	0.24	0.15	0.18	0.24	0.27	0.27	0.27	0.22	0.35
CAM2	0.36	0.4	0.19	0.28	0.15	0.21	0.27	0.29	0.28	0.32	0.24	0.38
CAM3	0.56	0.57	0.34	0.35	0.32	0.3	0.41	0.42	0.43	0.4	0.31	0.43
Fusion	**0.38**	**0.39**	**0.2**	**0.24**	**0.16**	**0.16**	**0.27**	**0.27**	**0.3**	**0.29**	**0.21**	**0.35**

3.9 Scenario-3: XR Training on the Job

The training on the job was implemented on see through HoloLens devices in a mixed reality (MR) environment, which is a combination of real physical and virtual objects. In this step the user is guided by a virtual UI but the working tools and equipment are real, so the user can perform the activity with help if necessary/required. The use of HoloLens 2 [8], a state-of-the-art tool in the field of mixed reality, will help the operator in performing the tasks.

This technology employs holograms (3D images with correct prospective and POV) projected on glasses in order to see and interact with augmented elements within a real environment.

The system recognizes the controlling UI touch events trends, time spent on UI interaction, and when the UI is still needed. The evaluation procedure will maintain some common features with the VR evaluation procedure to assure continuum of evaluation.

The logical cycle of the training integrated and developed is described in the paragraphs below.

The application includes high level of interaction between real and virtual environments assuring:

- Picking objects and their virtually assembly to the engine.
- Spatial mapping.
- Synchronized moving of real parts and holograms.

3.9.1 Mixed Reality Toolkit (MRTK) Tool

To achieve our goal of creating a holographic UWP application to train operators learning a new assembly operation or to support operators through an interactive step-by-step GUI to replace or supplement the traditional assembly worksheet, we need to create 3D scenarios through the identified game engine (Unity) and create interactive elements using voice commands or gestures.

In the context of eXtended Reality and the development of holographic solutions in Unity editor, the Mixed Reality Toolkit (MRTK) represents a powerful open source development kit that can interface with the IDE of choice. Therefore, the MRTK (version 2) was chosen as the toolkit for the development of the wizard and user interface for receiving information from the CPSoSAware system.

The toolkit is designed for the development of mixed reality applications with specific tools to facilitate access to HoloLens 2 functionality.

Through a customizable profile system, the toolkit, once imported as an asset and added to the scene (the toolkit is added through two GameObjects, the MixedRealityPlayspace and the MixedReality Toolkit), allows the user to set the different aspects that affect the interaction with the game engine and, in particular, with the device being deployed in the application; specifically:

- Camera

- Input
- Boundary
- Teleport
- Spatial Awareness
- Diagnostics
- Scene system
- Extensions
- Editor

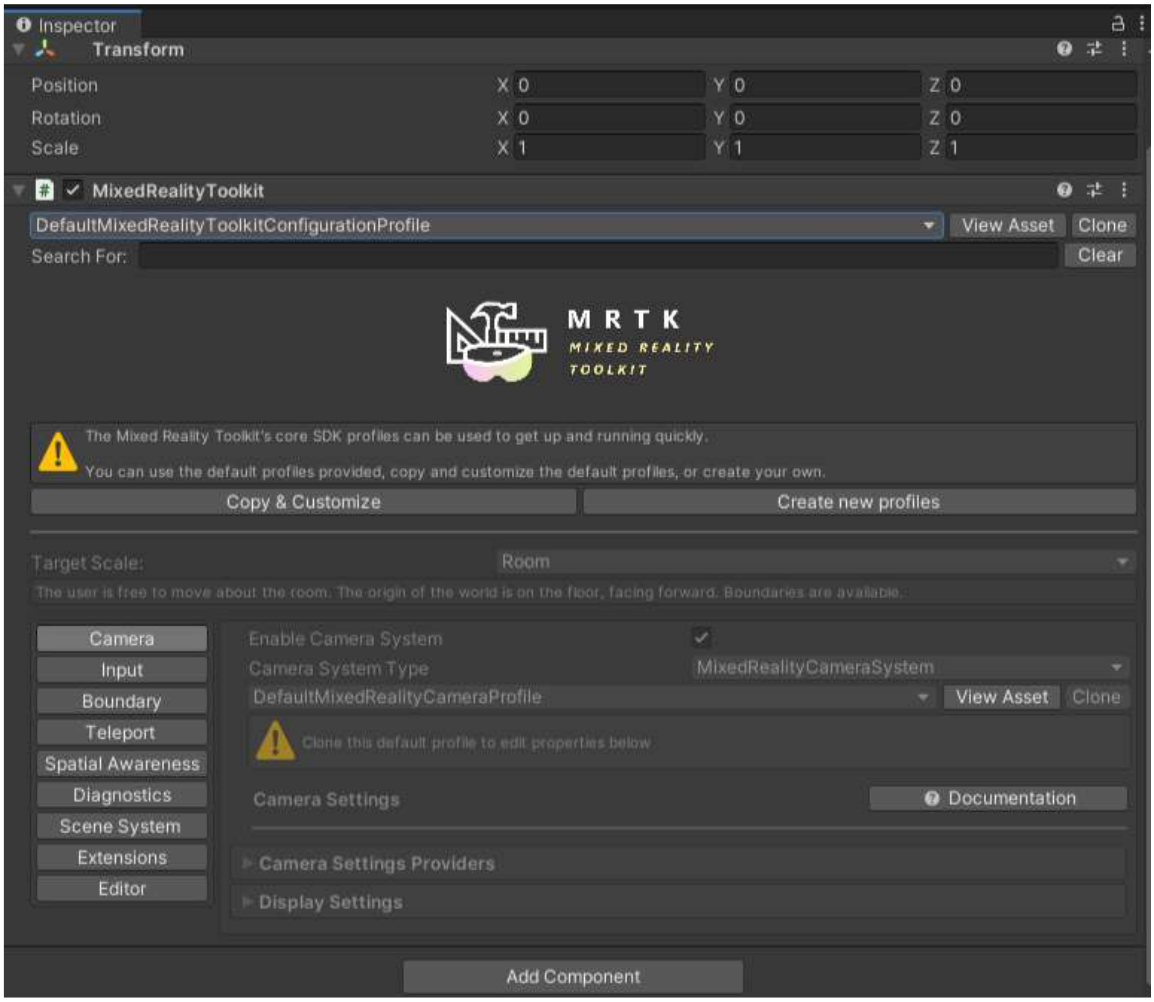


Figure 54: MRTK Scene GameObject - Inspector tab details with configuration tabs

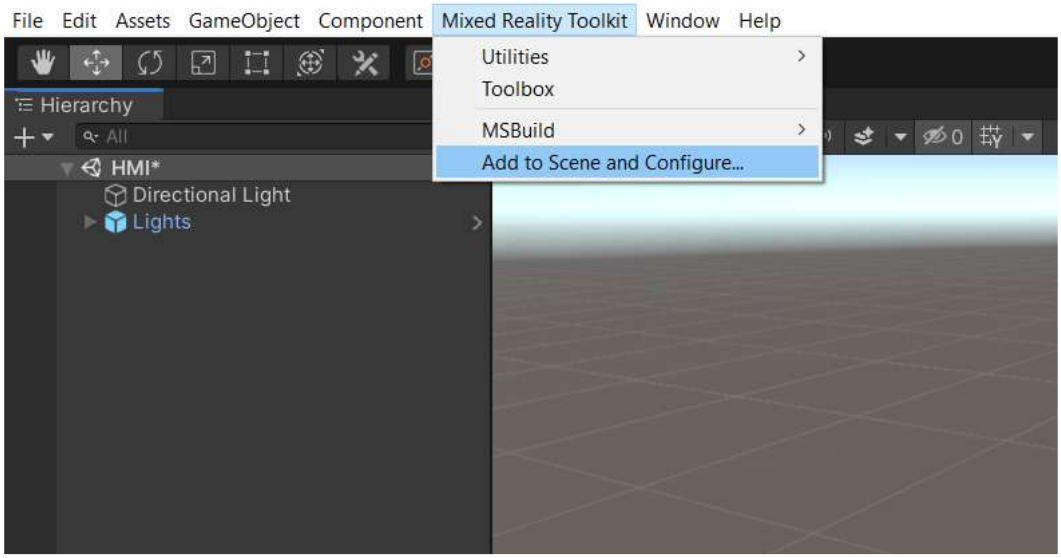


Figure 55: Mixed Reality Toolkit (MRTK) Asset - Add to Scene

The profile mechanism allows to increase the flexibility level of the system as it can be adapted to different areas of Reality Spectrum (MR, VR) and therefore to different devices (Hololens1, Hololens2, Oculus, etc...).

It's not the purpose of this document examine in depth the MRTK, but we can say that through the customization of modules, we can characterize all the aspects related to the perception of reality environment through the HoloLens' see-through display; for example, activating the spatial awareness feature allow the complete integration of the 3D virtual scenario into the real environment.

To carry on our analysis about the toolkit, we focused into the various library system contained in the asset which makes a lot of different interactive 3D objects (prefab) available for developers to create User Interface (UI). Among the main prefabs we can find the interaction buttons reachable at MRTK/SDK/Features/UX/Interactable/Prefab path, and the behaviour scripts for object manipulation; in particular:

- *Bounds Control*: a component script for managing objects by controlling the edges
- *ObjectManipulation*: a component script for manipulation (translation, rotation and scaling) of 3D objects; it is possible to discriminate type of manipulation through how user interact with object (with one or two hands) by personalizing the component parameters
- *Interactable*: a component script to collect all interaction type inputs, such as touch, ray cast or voice, and transforms them into visual output
- *Solver* a component script to manage, on the basis of a specific algorithm, the behaviour and relative real-time positioning of an object with respect to others. This is useful in case of we want to develop a following object with a specific range of distance in order to follow the movements of master to which it refers; the *RadialView* predefined block allows you to maintain the radial position as in a viewing cone with respect to the object to which it refers; the *HandConstraint* predefined block allows you to associate an object to the user's hand and therefore to its movements.

- *VoiceControl:* a component script that allow the management of voice command through the *SpeechInputHandler;* the module allows you to define keywords that generate an input event when recognized.

Figure 56: User Interface (UI) interaction example

3.9.1.1 Application

After identification of game engine and tools for developing, next step was to design the Universal Windows Platform (UWP) App architecture.

UWP App was designed with three mainly macro-blocks:

- A block to align digital and real world
- A block to manage interaction system
- A graphical user interface block to interact with digital contents

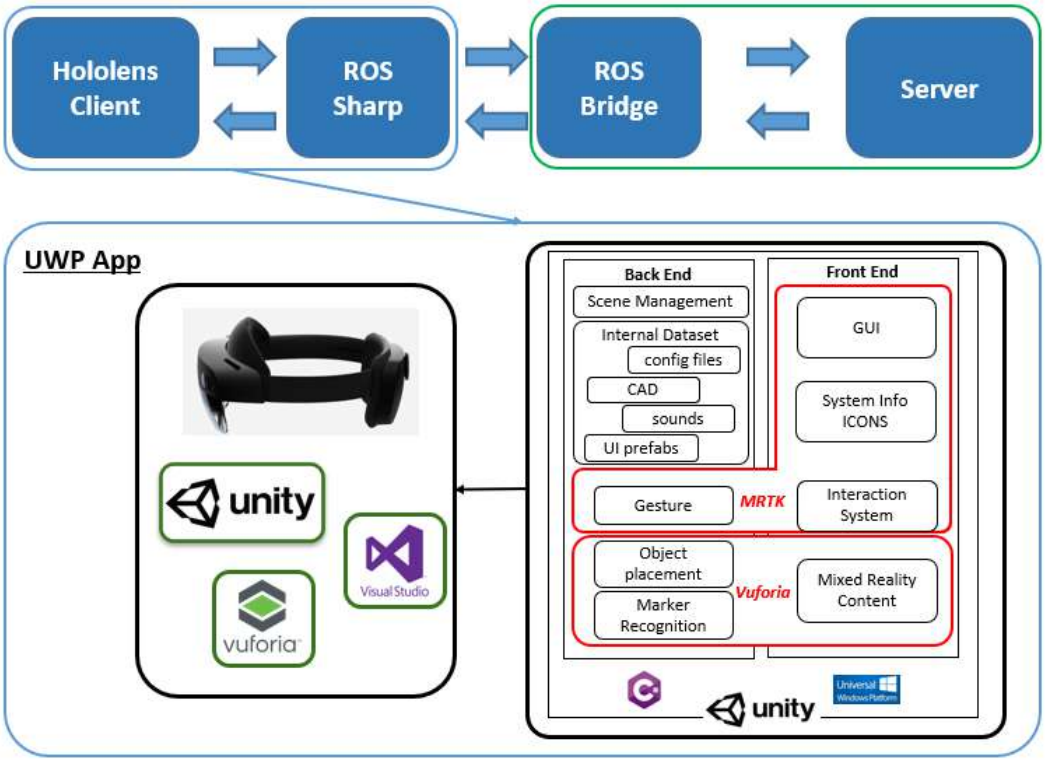


Figure 57: UWP App Architecture

The first block was covered by an instance and a customization of Vuforia Asset. This sub-system has the assignment to manage the virtual reality content (not linked to head position) in a three-dimensional space to allow visualization of digital contents overlapped to real world. This is possible by using of a 2D (in our case) or 3D Marker that represent a reference frame useful in overlapping procedures. Framing the marker through HoloLens camera the system recognizes the features from it and create a hidden reference frame. All the digital contents associated to marker will be visualized on see-through display and will be anchored to the marker position.

In our application, this solution could cover the requirement of a graphical support to indicate at operator where is the specific area of assembly workplace and where mount the component in assembly process. What kind of objects load on virtual scene depends on requirements and complexity of application, because it could be using an animated 3D ghost of the real component (with higher computational load) or a simple indicator, based on simpler geometrical objects (with lower computational load). In UWP holographic application we used a 2D marker (simpler to install on robot gripper) and basic geometrical 3D object as indicator for guided assembly. The choice of marker positioning on robot gripper is useful in order to have solid movement between the work surface and the components assembly positions relative to marker itself.

The second block was cover by gestures and input management of MRTK; we preferred to use a non-vocal system to interact with graphical virtual components, to prevent errors due to the inherent noise of assembly areas in industrial environments. Therefore, we opted for a more traditional and effective choice

such as using the hands (not necessarily free) to interact with UI. Pointers generated by tracking of hands, are used to interact with GUI and 3D Objects; they could be *Far Pointers* for far away objects through use of cast lines generated by hands, or *Near Pointers* for near objects that are close enough to user to grub touch or manipulate objects.

Finally, the last block is represented by all graphical virtual objects used in the scene and visualized by user through the see-through display. This set of virtual elements is divisible into two macro-blocks, with different scope and functionality. The first one is the System Info Icons, a set of buttons and icons used to activate the step-by-step guide and to inform operators on system status in real-time; the second one is the step-by-step GUI, an interactive window to support operator in the assembly process.

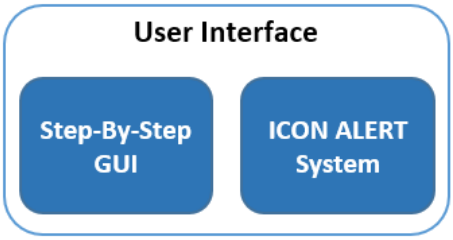


Figure 58: UI schema

For the icon system, we have assumed the use of an interface that is always present in the operator's field of view in order to inform the user in real time of any anomalies in the system. The system has the aim to inform operator on OSM system status, on ergonomics, on own ID group height cluster and on robot status (whether stationary or still in motion).

As always in the FOV, a minimal design user interface with a low cognitive load was chosen, so as not to distract operator during assembly of the components and to easily identify them through dedicated colours and a pre-established layout position.

For the OSM system, a multi-icon solution was chosen to easily inform the operator of the thermal status, heart rate and distraction status: in particular, for distraction status, a red blinking icon was identified with the simultaneous execution of a high-pitched sound (played directly from the HMD device speakers), for the thermal status a fixed icon reporting textually the status and a background whose colouring varies according to the value of the status itself, and finally, for the heart rate a fixed white icon with a textual reference to the heart rate value.

For ergonomics, a solution with a single and non-fixed icon was chosen, which is displayed exclusively and for the time necessary following the recognition of an inappropriate position of the operator.

For height cluster and relative operator’s ID group, a solution with a single and fixed icon was chosen; it changes colour when system recognize operator’s height and icon textually show the ID group.

Finally, for the robot status, and therefore to allow to start assembly phase, a solution with a single fixed and interactive icon with a colouring system has been chosen; in fact, at the start of the activity (with the robot stopped in home position or still moving) it informs the operator not to approach the assembly

position, then (after receiving the ROS message that the robot has reached the assembly position) it enables the operator to call the guided support/teaching system in MR

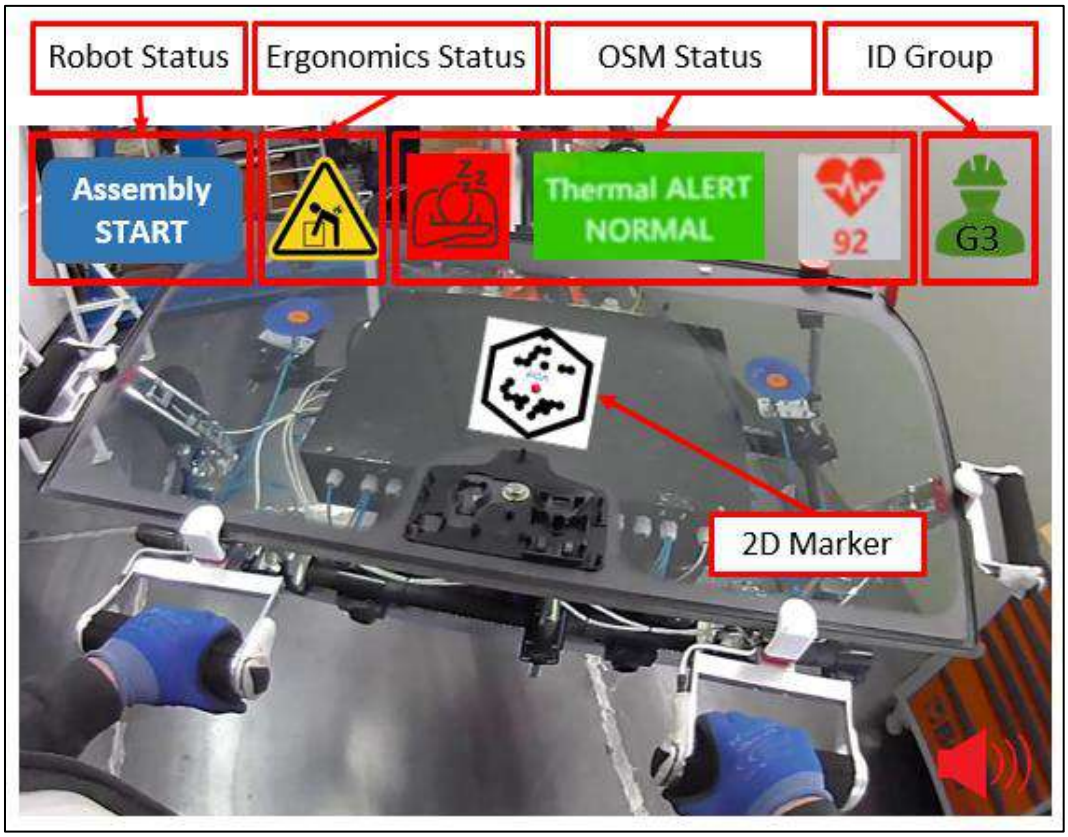


Figure 59: UI Icons System - design phase

In the image below a schematic representation of Icon System operative flow in UWP Application.

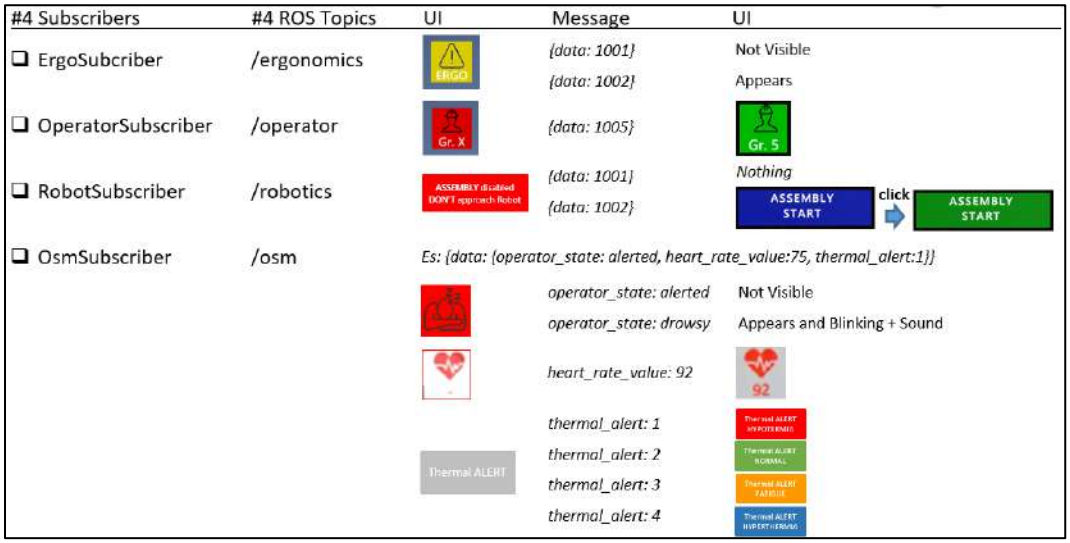


Figure 60: UI Icons System - Schema

The second macro-block of UI is represented by Step-by-Step GUI; it is a digital and interactive representation of a paper guide. The graphical interface is developed as a floating window following operator to be into the FOV always; at development phase the GUI could be anchored in a specific point into 3D space environment so that operator's FOV is free from this supporting information on assembly phase.

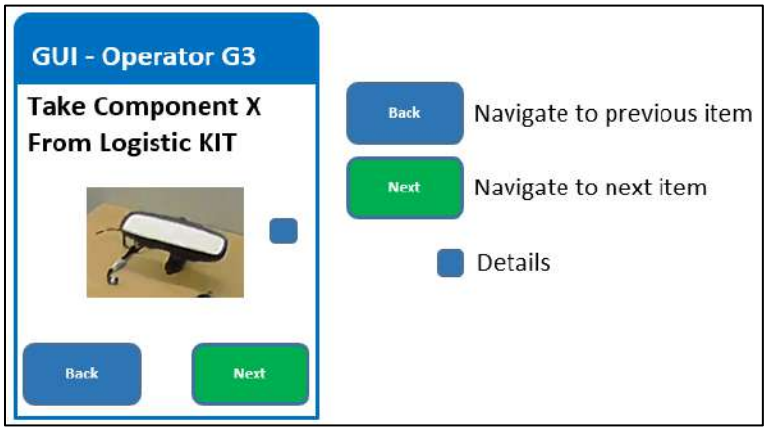


Figure 61: UI Step-By-Step GUI System - design phase

After operator receives the ROS message that the robot reached the assembly position, he can start the assembly process or, via UI robot status icon, begin the guided/teaching process.

The guided/teaching process starts with a small window suggesting to start the alignment task between real and digital world, through framing of the 2D marker. After the recognition of the marker, the main interaction window invites us to start the guided path.

The QR-code information is used to load the correct assembly process; in this way, the operator does not have to know the assembly sequence a priori or the sub-set of components to be picked up but is supported step by step by the graphical information. In parallel to the textual and visual information of the GUI, the system loads the 3D visual components defined to indicate the precise assembly position.

As we shall see later, the holographic system is intended to both support the inexperienced operator in workstation teaching or in performing a complex task, thus, depending on the operator's degree of competence, increase the level of information detail to support the operator by providing as visual content linked to the GUI the assembly worksheet of the component and a demonstration video showing the assembly of the component in a position close to the assembly area.

3.9.2 Scenario, sequence of operations and information flow

3.9.2.1 Assembly Worksheets

The activities that each operator must carry out for the assembly of the various components are described in detail below. These descriptions are contained in the Assembly Worksheets available to each operator in order to standardize the assembly procedure

Operation #1: Rain sensor assembly on the bracket already present on the windscreen

- Take the sensor with the respective hooks from the line side container
- Remove the protective cap (RIF. C)
- Position the sensor, with the connection side facing upwards, by manual pressure in the appropriate seat until it is mechanically hooked with the three clips (RIF. D and RIF. D1)

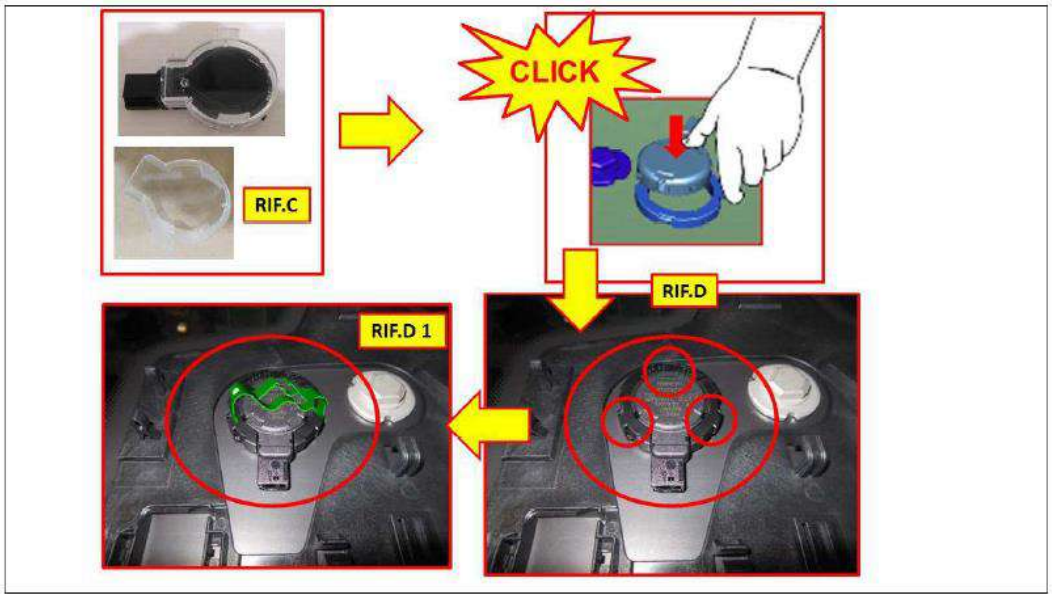


Figure 62: Assembly worksheet detail - Rain sensor

Operation #2: Humidity sensor assembly on bracket already present on the windscreen

- *Take the humidity sensor from the line side (REF. G)*
- *Remove the protective cap*
- *Apply the sensor by manual pressure in the special seat already present on the windscreen with the connection side facing upwards (REF. H)*

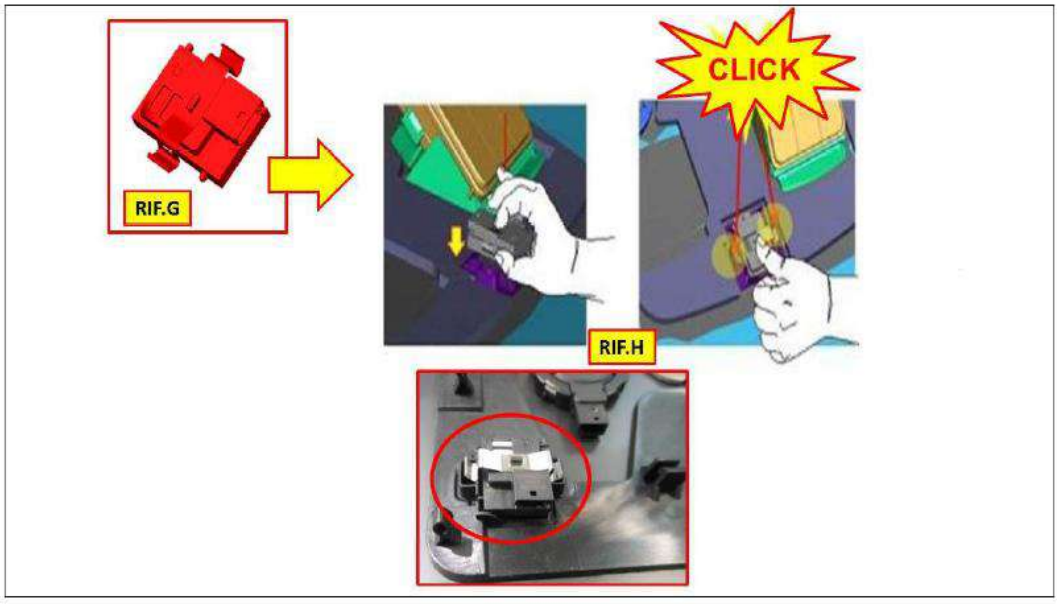


Figure 63: Assembly worksheet detail - Humidity sensor

Operation #3: Rear view mirror assembly on bracket already present on the windscreen

- Take the internal rear-view mirror from the line side container (RIF. I)
- Before mounting the mirror on the windscreen, rotate the support stem 90° counterclockwise, orthogonal to the mirror (with the control button facing upwards) (RIF. Z)
- Apply the rear-view mirror to the bracket already present on the windscreen and rotate it by approximately 45° to hook the mirror onto the base (RIF. J)
- Rotate the mirror downwards to avoid the photocells of the automatic system

Figure 64: Assembly worksheet detail - Rear view mirror

Operation #4: Carry out the mirror connection harness

- Take the cable from the line side container (RIF. O)
- Connect the connector to the humidity sensor until the mechanical hooking occurs (RIF. P)
- Connect the connector with the rain sensor until the mechanical hooking occurs (RIF. Q)
- Connect the connector to the rear-view mirror until the mechanical coupling occurs (RIF. R)
- Insert the connection bridle in the clip under the mirror (RIF. S)

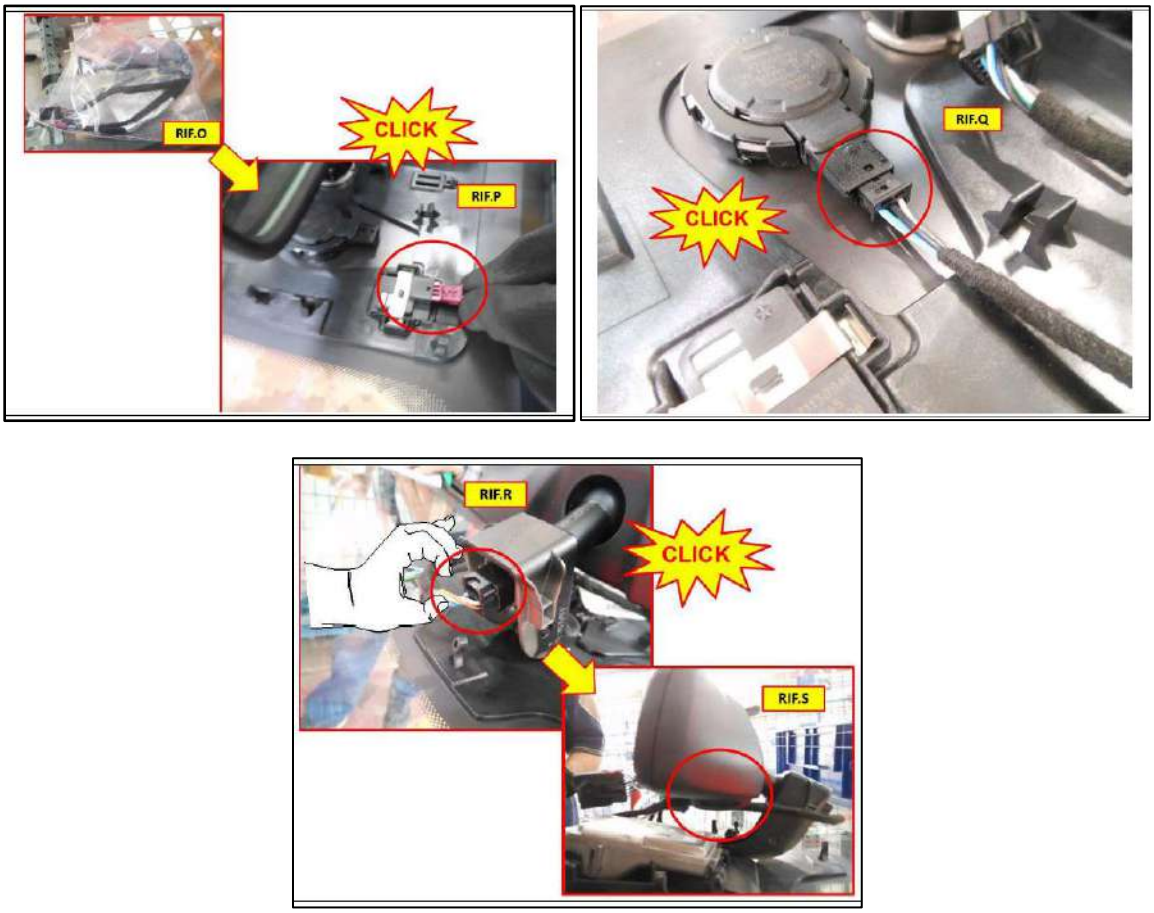
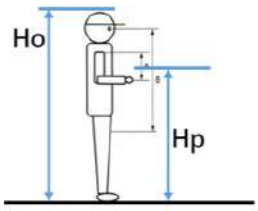


Figure 65: Assembly worksheet detail - Connection Cable

During physical tests with robot assembly cell, it was notice that at *Operation #4*, specific in *step 4*, to connect cable to the rear-view mirror until the mechanical coupling occurs, the operator required a different approach to assembly work plan; in particular, to connect cable's connector behind the rear mirror, operator required a lower work plan to operate more efficiently. To remedy at this particular situation, due to a specific complex assembly step, we had to revise the ergonomics analysis phase to be sure to provide the best solution in term of assembly work plan. For this, in physical cell we have lowered the height of assembly work plan by 100mm. The solution, validated in virtual environment with Process Simulate, has allowed us to remain in the Strike Point Zone of the operators.



Expected scalar	Ho= height of the Operator	Hp= Height of assembly plane
1001	h < 1544	965
1002	h < 1641	1035
1003	h < 1727	1100
1004	h < 1829	1165
1005	h > 1829	1250

Figure 66: Operators' ID group classification updated

3.9.2.2 Application operative flow and results

In this section is described the operative flow of MR UWP Application on HoloLens 2 and the main results obtained. Below a schematic view of application flow.

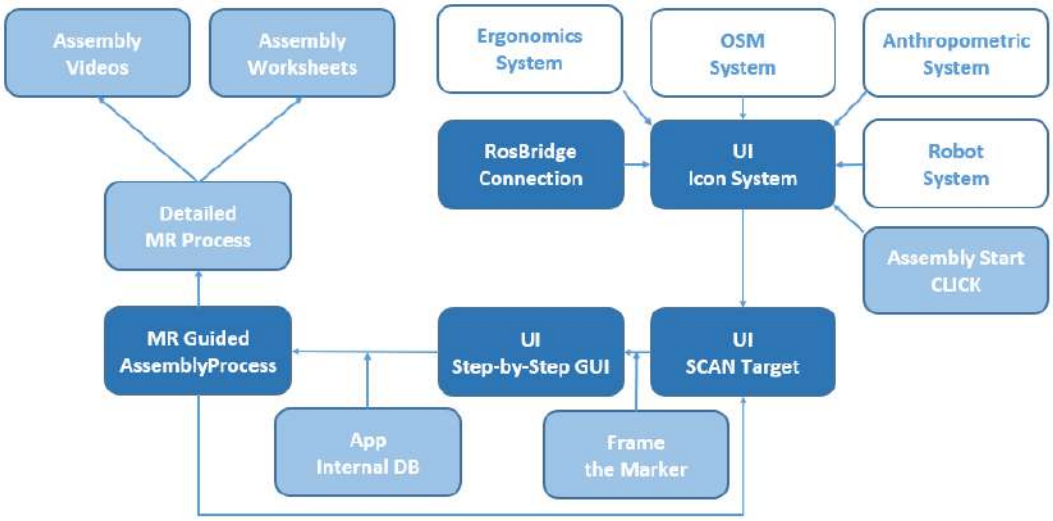


Figure 67: UWP App operative flow - schema

The holographic application deployed on the HoloLens 2 device, starts with the UI icon system loaded but in standby mode, waiting for the connection to the ROS network. This feature increases the flexibility of the application in case of modular replacement of the Ros bridge server.

Figure 68: UWP App - RosBridge connector

Once the connection to the server has been initiated, the system automatically subscribes to the ROS topics *ergonomics, robotics, operator* and *OSM*; in this way it is listening on the ROS network and ready to receive information from the system.

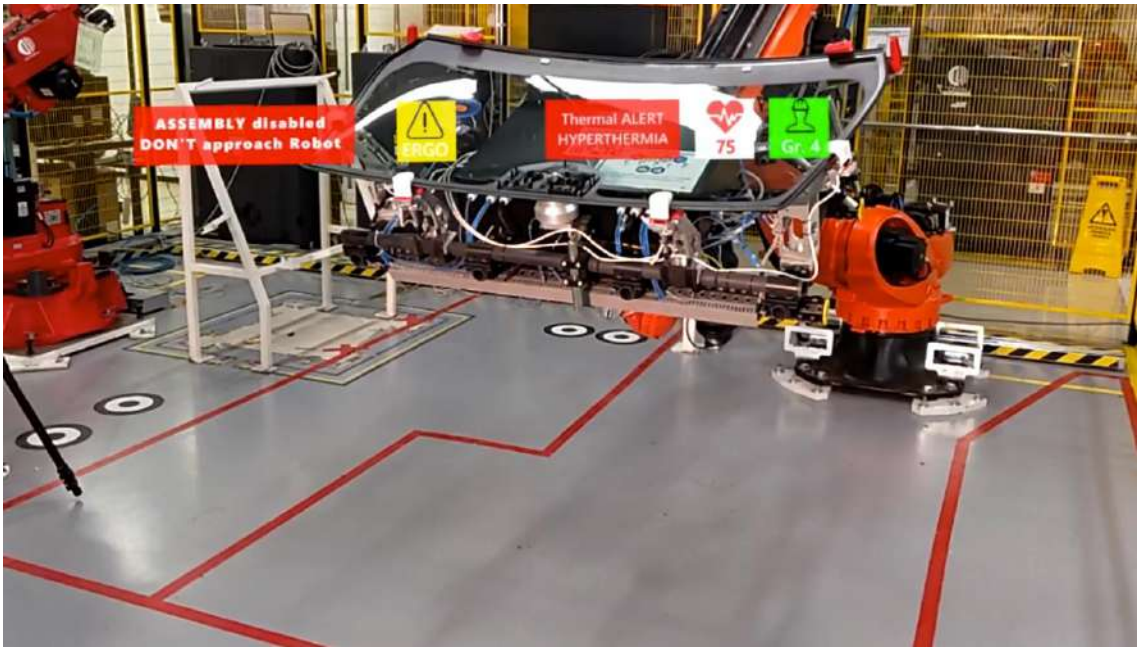


Figure 69: UWP App - Not Approaching phase - UI Icons System updated

The *UI Icon System* sub-system that is responsible for informing the operator and notifying the information received on the network is updated in real time through the receipt messages. The operator (positioned outside of assembly zone) remains on hold until the robot picks up the component and positions the windshield in the assembly station, adjusting the final height of the work plan to one of the five levels defined by the ergonomic analysis; once the operation is complete, the operator receives the OK to proceed through the change of the robot status icon.

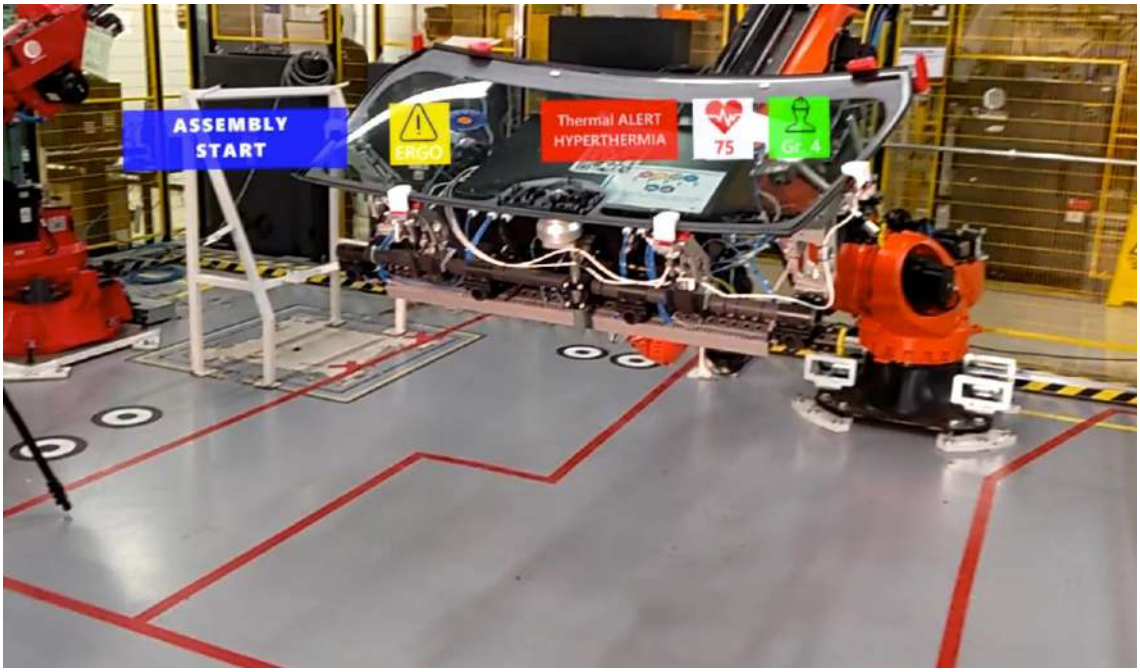


Figure 70: UWP App - OK to proceed, robot is in assembly zone (height for Group ID 4)

Through pressing gesture on this interactive holographic button, the operator enables the task of identification of the use case and alignment of the real world with the digital world through the Scan Target module; the task consists in framing within a target anchored to the HoloLens camera (therefore integral to the operator view) and centred on the FOV, the 2D marker positioned on the robot gripper.

Figure 71: UWP App - SCAN Target task

The recognition task is performed autonomously by the Vuforia system; when the target object is recognized, the *UI GUI* task is started, and then the main floating window of the graphical interaction system is displayed.

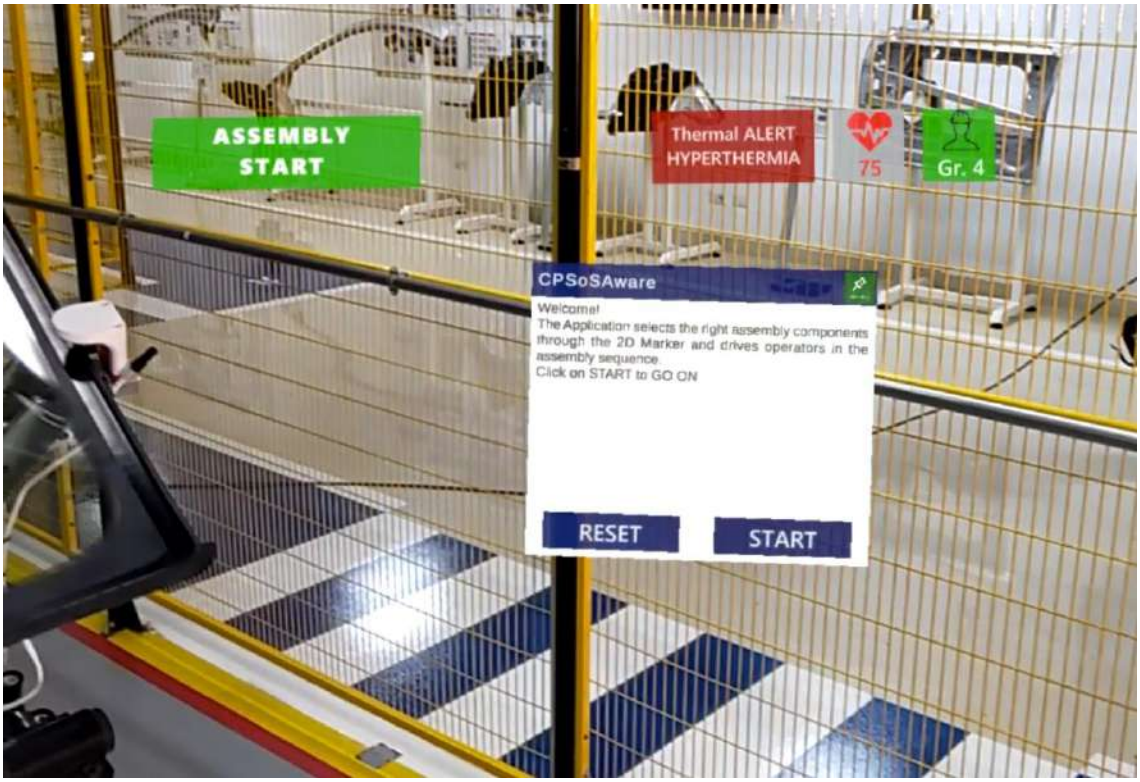


Figure 72: Marker recognized - UI Step-by-Step GUI starts

The GUI guides the operator in the execution of the task through:

- a textual description of the task
- an identification image of the component to be assembled
- a squared content in MR 3D superimposed on the windscreen indicating the area to be assembled

The operator is allowed to:

- fix the GUI in a random position in the 3D space (it is recommended to place it in the immediate vicinity of the assembly area so that it is always visible with minimum effort)
- navigate the assembly sequence backwards and forwards
- request a higher level of detail of the information content for the component to be assembled, through a feature on the GUI itself.

Figure 73: UWP App - GUI details - main window (up) and 3D MR content (bottom)

The request for further details starts the process of searching the internal database for information on the current task, as planned in the design phase; if present, we can see:

- the pdf of the component assembly worksheet (displayed next to the GUI)
- the explanatory video of the component assembly process (positioned in the immediate vicinity of the working area).

In this way, the system supports even the most inexperienced user in the operations to be performed.

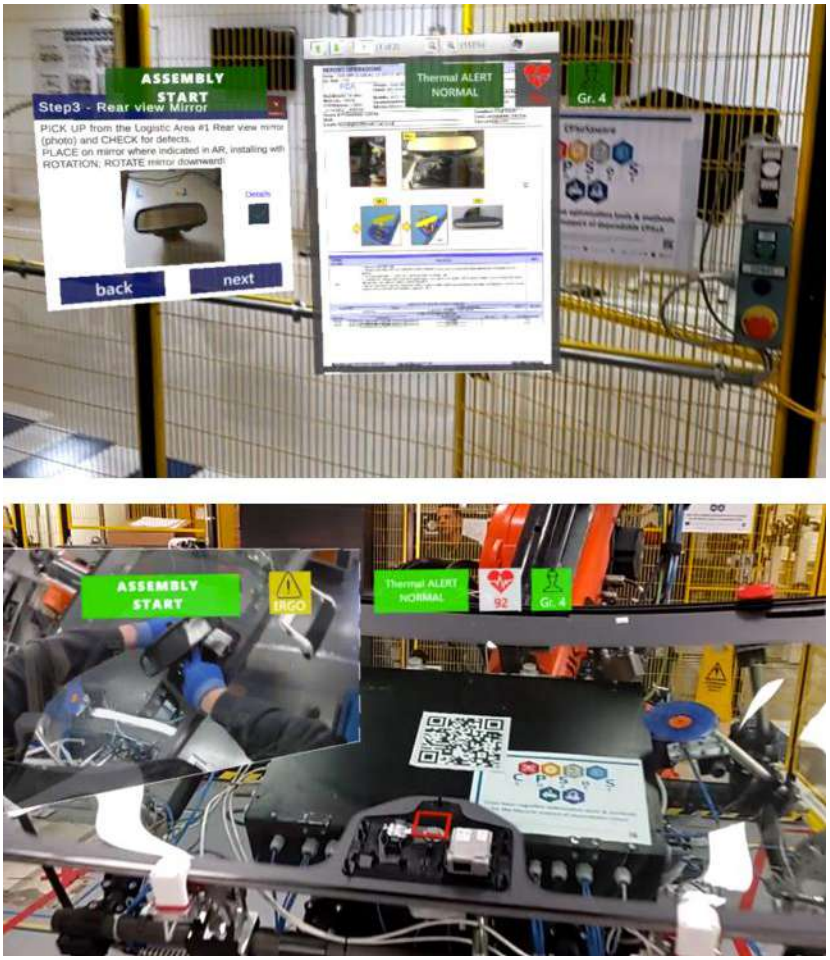


Figure 74: UWP App - Detailed info - assembly worksheet (up) and video (bottom)

When the assembly sequence is finished, the operator could navigate the entire sequence executed through the GUI to check on that all steps have been completed or push the physical button of the assembly cell to communicates to the robot that the assembly sequence is finished and then go back to the main window view of the GUI and then reset the assembly sequence. In this way the app returns to the SCAN target state for a new assembly process.

4 Connected and Autonomous L3-L4 Vehicles

Vehicular electronics has currently gained growing recognition, and independent driving is a passionate argument. To gain secure and reliable knowledgeable conveyance structures, accurate standing sciences need expected built to determinant in the various types of doubts such as pedestrian behaviour, chance objects, and types of roads and their backgrounds. In CPSoSAWARE, we inspect the multitude of sciences necessary to build an independent bicycle and have tested the answers grown during the whole of the execution of the project. CPSoSAWARE, has brought most advanced level in atmosphere detection, pedestrian discovery, course preparation, motion control, and vehicle cybersecurity for independent taxis. The association has studied the various projected sciences and equate their approaches. For a car to enhance completely independent, these technologies need expected correct enough to gain public trust and show huge accuracy in their approach to answering these questions.

The Society of Automobile Engineers (SAE) delineates six levels of industrialization [1]. Level Zero Automation displays no automation. At level Zero (0), the human operator acts the complete Dynamic Driver Task (DDT) by way of alive security methods. Level One Automation has the Automated Driving Systems (ADS) act either long or sideways control of the vehicle, but not the complete Object and Event Detection and Response (OEDR). The DDT retreat is the motorist, so the chauffeur must be continually present in the automated vehicle and able to have or do take control of the automobile at whatever time. Systems like Electronic Stability Control, Anti-lock Braking, and Adaptive Cruise Control happen under Level One Automation. In Level Two Automation, the ADS can act in both long and sideways control of the automobile but is complementary as Level One and two cannot act the complete OEDR. The DDT retreat repeated is the chauffeur, so they must forever show in the bicycle and within financial means take control of the cab at some instance. In addition to the methods noticed in Level One, Automatic Emergency Braking and accident stop structures lie under Level Two Automation. In Level Three Automation, the ADS acts the overall DDT. The DDT fallback is the human operator (driver), and the trainer gets informed if the Operational Design Domain (ODD's) limits will be surpassed. The driver still gets informed if some act-appropriate arrangement forsakes. Level Four Automation does not demand any human mediation. The ADS of a cab at Level Four Automation will act the whole DDT in limited ODDs-places accompanying decent maps and boulevard buildings. The chauffeur can happen if they be going to, but the ADS will attain a littlest risk on some DDT Fallback. Level Five Automation is the last level of mechanization. There is complete mechanization, and the ADS can act the whole ODD outside some restraints. DDT retreat is further controlled for one ADS and the only opportunity a trainer takes control of the car is when they request it.

Sensors Used in Autonomous Vehicles

The autonomous Vehicles engaged in CPSoSAware Automotive PIlot are equipped with a multitude of sensors operating in a broad spectrum of the spectral band. More specifically we have:

Cameras:

The car engaged for CPSoSAWARE is equipped with cameras operating at the visual spectral band to supply a 360-degree view of the environment. They are effective at object discovery and classification, and this data is transferred to the ML-based algorithms for further use. However, afore-mentioned cameras are sensitive in poor illumination conditions and produce high volume of data to be processed. Thus, redundant sensing modalities are necessary for fully ADS approaches.

RADAR:

Corresponds to a sensor that uses electromagnetic waves to reckon determinants like distance, speed, and angle. With the help of radar transmitters, AVs can receive electromagnetic signals and endure the reflected waves by way of sonar receivers. Radar works well in a diversity of weathering conditions, and long distances. However, it may erroneously recognize objects.

LiDAR:

A sensor that uses light as a medium to measure distances is determined by evaluating the time it takes for the light to get reflected on the receiver. Lidar Sensors emit ray of light beams that hit the surrounding objects, get reflected and tracked to a photodetector. The composed beams are converged together like a point cloud. This creates a 3D modelling of the environment. Although LiDAR is an effective and adept sensor, it is still highly priced.

GNSS:

It details the map and arrangement of satellites, providing facts related to the location, speed of ego-vehicle, navigation attributes, and traffic. GNSS uses triangulation to discover the position of recipients and calculates the distance between satellites in the geostationary co-ordinate system and the vehicle. These signals, nevertheless, are feeble and maybe altered through interference.

Ultrasonic sensors:

They are temporary parking sensors and are price efficient. They have a restricted operational range but are ideal for reduced-speed positions. These sensors are mainly used to measure the distance of an object in the vicinity by utilizing quick waves, and the reflected signals transformed into electric signals.

GPS:

It is a sensor that supports consumers on issues related to geo-localization for navigation

By discussing the opportunities as well as the obstacles of autonomous driving technology, we aim to shed light on future possibilities

4.1 Human in the loop control use case in single vehicle scenario

The following section contains a description of the steps made to investigate the overall performance of the DMS application developed under the CPSoSaware project. Methodological aspects, as well as the results of the investigation are covered. During the study, datasets from N = 8 participants were collected and discussed; performance of the application was also tested from the UX perspective.

4.1.1 Purpose of conducted tests

The purpose of the study was to test and validate the DMS smartphone application. For that purpose, a collection of video recordings was registered. Video sequences were recorded with the use of two smartphones and an interior dash camera, and were collected among a total number of *N = 8* participants. Naturalistic driving study was performed in Warsaw, Poland. Together with the video sequences and associated JSON output files, synchronized datasets derived from the raw files, as well as additional visualizations of application performance, were provided.

Along with naturalistic testing, static testing based on Euro NCAP testing recommendations was conducted during each study. Supplementary document containing the order of the sequences is provided in *Appendix 1*. Application performance was also assessed from the UX perspective (see *Appendix 2* for a detailed description of tested use cases).

Data gathered in the scope of the tests include:

- Video recordings from tested DMS solutions
- Contextual video recordings
- JSON output files associated with video recordings from tested DMS
- Synchronized datasets derived from JSON output files
- Plots of selected variables from synchronized datasets

Moreover, a python script used for synchronization and plotting was created.

4.1.2 Testing methodology

Initially, the testing protocol, introduced in previous deliverables, was oriented at conducting the study in a driving simulator. However, in the scope of 2021, a new Euro NCAP protocol dedicated to the evaluation of Driver State Monitoring systems was released (see European New Car Assessment Programme (Euro NCAP), 2022). As the Euro NCAP protocol gives detailed instructions on how the distraction testing for the Driver Monitoring System should be conducted, it was decided that its methodology should be followed. Due to the optimal conditions for the app testing and the release of the protocol, it was decided that the best solution to optimally test the application's capabilities would be to conduct naturalistic driving scenarios preceded by static sequences consisting of head and eye movements (according to Euro NCAP definitions and guidelines for distraction testing). The application's functionality was then tested under laboratory conditions for the detection and recognition of certain events and actions in predefined use cases in order to assess the UX-related aspects of performance.

4.1.2.1 Ethical statement

The study was conducted in compliance with the Declaration of Helsinki from 1964 and its successive revisions, as well as with the Code of Ethics of the Polish Psychological Association. Every participant filled in a written informed consent before taking part in the study.

4.1.2.2 COVID-19 related measures

Due to the SARS-Cov-2 pandemic, special safety procedures were applied during the whole data collection procedure. Majority of participants were driving in their own cars and equipment that was transferred between the cars was systematically disinfected after every test drive. Additionally, before the test drive was conducted each participant was asked whether they have had any contact with any potentially infected person. Only participants who responded negatively were admitted to participating in the study.

4.1.3 Tools & equipment

4.1.3.1 Tested DMS solution

The research described in the foregoing section aimed to test the Driver State Monitoring system based on the Android application installed on the smartphone. The application uses the front camera of the smartphone directed at the participant's face to constantly monitor the driver and extract landmarks of the facial features (e.g., eyes, mouth, and face shape). The algorithm of the application was aimed at detecting events such as eye closure, yawning, or distractions (pose detection). This algorithm was applied to monitor the occurrence of drowsiness or distraction signs. Two smartphones were used in the data collection procedure: Samsung Galaxy A50 (manufacturer: Samsung Electronics, software version: Android 11), and Samsung Galaxy A13 (manufacturer: Samsung Electronics, software version: Android 12).

4.1.3.2 Heart rate monitoring

For the purpose of monitoring heart activity, a wearable device was used. It was a Samsung watch (model: Galaxy Watch4, manufacturer: Samsung Electronics)—a smartwatch that was mounted on the participant's wrist. The device was used for the purpose of long-time, non-invasive heart activity estimation and monitoring. This made it possible to collect data about the driver's heart rate and its change in time course both in static and naturalistic context.

4.1.3.3 Contextual cameras

Three contextual cameras (model: A139 3CH G, manufacturer: VIOFO) were used in addition to the two phones. Cameras overlooking the road ahead and behind the test vehicle were RGB cameras used for lateral behavior monitoring, i.e. vehicle position and lane keeping monitoring, allowing for indirect monitoring of driver state, as well as for collecting contextual information about road events. The camera directed at the driver was an RGB camera with the addition of infrared illumination. The cameras were used to record the course of the drives in the naturalistic scenario. The recordings serve as an additional source of information and data about the scope and environment of the test drives.

4.1.3.4 Test vehicles

Due to COVID-19 safety measures and the need to create an environment as natural for the driver as possible, each participant had been driving their own, or familiar to them, vehicle. The exact models of the cars used in the study are summarized in Table 14.

During each drive, two smartphones with the application turned on were directed with the front camera toward the participants. One of the smartphones (Phone 1) was positioned on the instrument cluster as it was recommended (excluding two participants—P03 and P04, who had the Phone 1 positioned on the windshield next to the left pillar, Position C in Figure 75). The second smartphone (Phone 2) was mounted in the place chosen by the participant—in the position in which participants usually have their smartphone with a map/music installed. Among participants, there were two locations, presented schematically in Figure 75 and exact information about positioning chosen by each participant is summarized in Table 14:

- Windshield under the rear mirror (Position A)
- Infotainment (Position B)

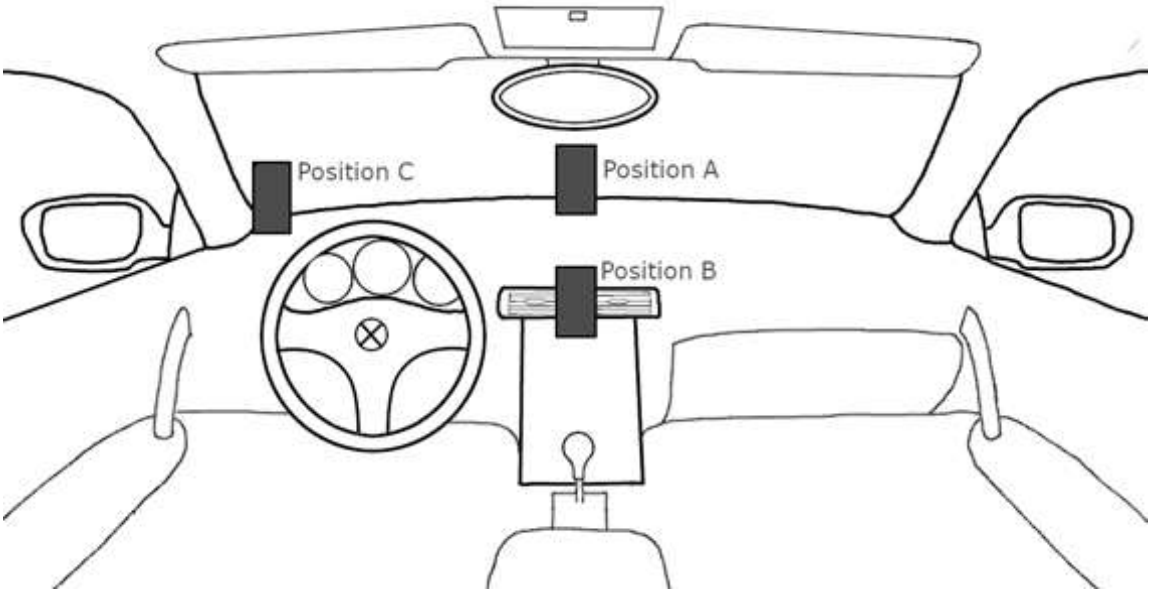


Figure 75: A schematic representation of a dashboard in an average car with phones' positioning during test drives. Source: Robotec.ai.

Table 14: Exact vehicles' models and phones' positioning used in each participant's test drive.

Participant's ID	Vehicle's model	Position of Phone 1	Position of Phone 2
P01	Ford Mondeo MK4	Instrument cluster	Position A
P02	Ford Mondeo MK4	Instrument cluster	Position A
P03	Ford Focus	Position C	Position B
P04	Ford Focus	Position C	Position B
P05	Audi A8 D3	Instrument cluster	Position A

P06	Peugeot 307	Instrument cluster	Position B
P07	Ford Kuga	Instrument cluster	Position A
P08	Ford Kuga	Instrument cluster	Position A

In addition, three more cameras (see subchapter *Contextual Cameras*) were placed in the car:

- Camera 1 — overlooking the street in front of the car (placed on the right side of the rear mirror, see Figure 76).
- Camera 2 — overlooking the street behind the car (placed on the rear window; not present in Figure 76).
- Camera 3 — aimed at the driver — equipped with IR illumination, so it was possible to compare the performance and results of the application with the IR recording during varying lighting conditions (placed on the front window above the driver, see Figure 76).

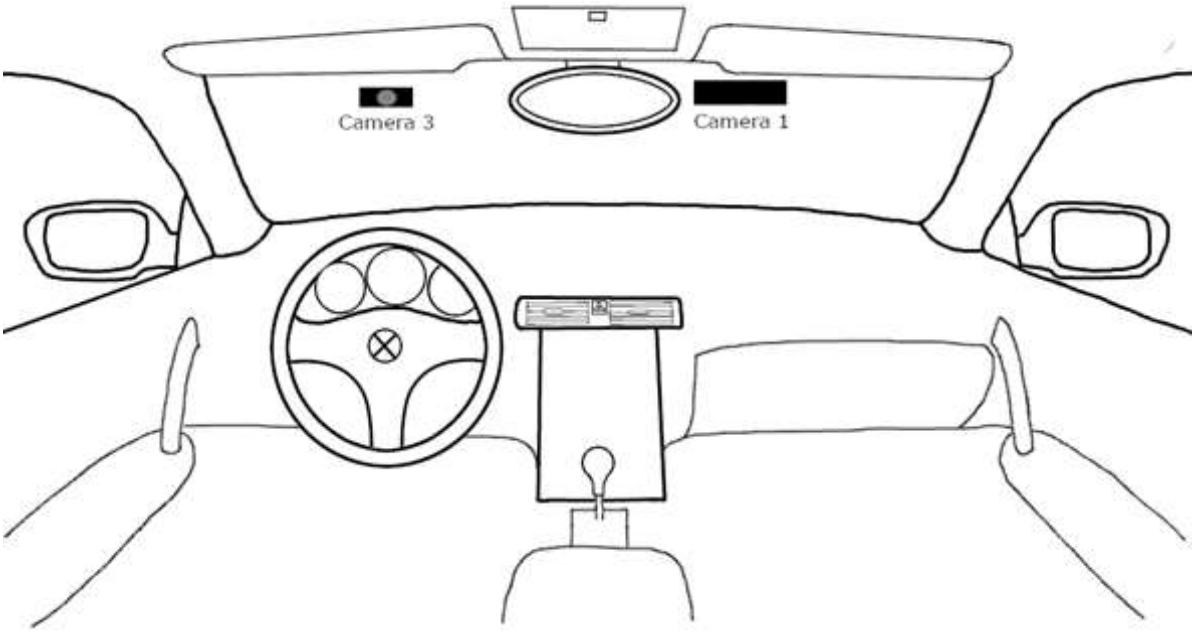


Figure 76: A schematic representation of a dashboard in an average car with marked placement of two out of three context cameras. Source: Robotec.ai.

4.1.4 Procedure

4.1.4.1 Questionnaire

Before the test drive started and cameras were installed in the test vehicle, each participant was asked whether they have a valid driver's license and whether they were aware of existing and applicable regulations of the traffic. Then, they were asked to provide information about their age, height, vision defects, driving experience, and type of work (whether they were shift workers).

4.1.4.2 Static scenario

Before the test drive in a natural environment, participants were asked to perform a static scenario, which consisted of sequences of eyes and head movements (see *Appendix 1*). The sequences of movements were designed according to the Euro NCAP guidelines, recommendations, and definitions of distraction (Euro NCAP, 2022). The sequences covered two different gaze movement types — "owl" and "lizard". "Owl" type refers to cases in which participants tend to turn their heads to perform peripheral gaze, whereas the "lizard" type is associated with a tendency to perform peripheral gaze by rolling the eyes, but not the head. The sequences covered gazes toward different objects inside the car cabin, as well as in the outer environment. Participants were asked to perform long-duration gazes, frequent and of short-duration gazes, as well as gazes toward their phone, located in different positions. An example checklist of all static sequences can be found in *Appendix 1*.

4.1.4.3 Naturalistic driving scenario

After finishing the static scenario each participant was asked to perform a naturalistic driving part. Participants started their drive from the chosen localization in or near Warsaw (for participants P01, P02, and the P04 starting point was the same) and then were asked to drive on either route proposed by the researcher or the route of their choice. Since all participants were experienced and active drivers, a safety backup driver was not present during every drive. Test drives were approximately 1 hour long and were conducted during the day (with the exclusion of participant P03, who was testing the performance of the DMS application during the evening hours). During the test drives both urban roads and expressways were included.

Research instruments applied in the naturalistic driving scenario include the DSM application tested with two smartphones, wearable-based heart activity monitoring, two external context cameras, one internal context camera, and a demographic survey.

4.1.4.4 Controlled environment testing

Aside from testing of the application in the driving context, the solution underwent testing in a controlled environment. Several use cases had been planned and then executed to see the application's performance. The purpose was to observe whether the application performs well without drastic and rapid light changes and without unnecessary additional movements inherent in driving under natural conditions. A list of tested use cases along with tests' outcomes can be found in *Appendix 2*.

4.1.4.5 Data synchronization

For each participant, five recordings (two smartphone screen recordings and three contextual cameras' recordings) were manually synchronized and cut to a selected frame of reference. The obtained precision of video synchronization is approximately 30 milliseconds. Data synchronization process was performed by two researchers: AS (P01-P04), OZ (P05-P08), and the software used to process the video recordings was OpenShot Video Editor (OpenShot Studios, LLC; licensed under GPL v. 3).

Logged data were subjected to quality check and the application's output JSON files were supervised and matched with the associated recordings.

4.1.4.6 Data summarization

In order to summarize collected dataset, and to graphically represent selected indices recorded during the experiment, the dataset was subjected to a simple analysis. Demographic data were aggregated and discussed and output JSON files associated with the DSM application's recordings were pre-processed and properly synchronized with the interior contextual camera recordings. Selected variables were then processed and plotted together in order to visualize and further qualitatively discuss the obtained outcomes. As there were two smartphones used in every recording session in parallel, there were two JSON files — one from each smartphone.

4.1.5 Results

4.1.5.1 Participants' sample and test drives information

The study sample consisted of *N = 8* participants. The sample was nearly equally balanced in terms of gender, with 5 male and 3 female participants. The participants were divided into four age groups specified in the following manner: <25 years, 25-40 years, 41-60 years, and >60 years. The sample is summarized in Table 15.

Table 15: Participants' age to gender distribution.

Gender	Age group			
	<25	25-40	41-60	>60
Male	P06	P04	P05, P07	P03
Female	P01	P02	P08	

There were no participants with a height below 170 cm. In addition, there were 3 participants wearing different types of glasses during the test drives (Figure 77A; P01 — photochromic corrective glasses with UV filter; P03 — plain, corrective glasses; P06 — sunglasses).

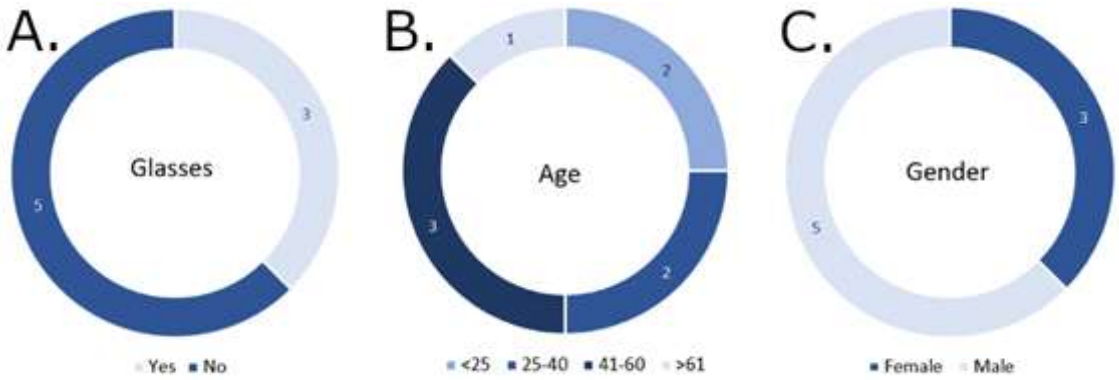


Figure 77: Participants' A. glasses usage, B. age, and C. gender distribution. Source: Robotec.ai.

Test drive duration varied in range of 44-90 minutes. The exact duration of each drive can be seen in Table 16.

Table 16. Exact durations of each participant's test drive.

Participant's number	Duration of the test drive (HH:mm:ss)
P01	01:09:43
P02	01:11:10
P03	01:29:36
P04	00:56:51
P05	01:12:05
P06	01:23:34
P07	01:02:58
P08	00:44:10

4.1.5.2 DMS application

Output from the tested application was stored in JSON files. Each file consisted of specific data for subsequent frames of the application's recording in use. For each frame there were stored: boolean data for alert being fired, detected distraction on the participant's left side, detected distraction on the participant's right side, yawning, and eye closure (for all those variables possible outcome was "true" or "false"), as well as the information about the frame number, frame timestamp, frame ID, output from heart monitoring device (heart rate for each frame) and the number of detected faces. An example of JSON file content is presented in *Figure 4*.

```
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 3190,
  "frame_timestamp": "2022-10-04T12:31:37.138569Z",
  "frame_uuid": "db3b0e11-8796-4d05-a533-6cbd828530c3",
  "heartRate": "71,0",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 3191,
  "frame_timestamp": "2022-10-04T12:31:37.625884Z",
  "frame_uuid": "c627d758-62fe-4f2b-b0d8-fc704902f9ac",
  "heartRate": "68,0",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
```


Figure 78: DSM Application's JSON file content. Source: DSM Application under test.

In *Appendix 2* there are described use cases for the application functionality. Each case is a description of a single event/action undertaken with the application in a controlled environment. Those use cases are separate from the naturalistic driving scenarios. Each table has a test case name meaning the name of the event/action that is described within the table, the hardware and system version of the phone used for testing, the purpose of the case testing, the description of the test, steps undertaken during the test, assessment criteria and the outcome of the test.

4.1.5.3 Analysis

***4.1.5.3.1** Analysis of JSON files*

Analysis of the output JSON files was conducted for each participant separately. Since there were two phones with the application running during the drive, there were two data streams under analysis – one from the phone placed in the position "Windshield" or "Infotainment" (Position A/B, *Figure 1*; encoded as "WINDSHIELD" on the charts) and one from the phone placed in the "Instrument cluster" position or in Position C (*Figure 1*; encoded as "HUD" on the charts).

***4.1.5.3.2** Timestamp differences*

For each participant and each phone that the application was running on, the timestamp differences between the two subsequent frames were calculated. This calculation was performed to monitor how even the time intervals are and to visually check for any periodicity or unusual drops in the application's data

sampling, as there was no constant sampling rate. For each participant, two charts of the timestamp difference were created.

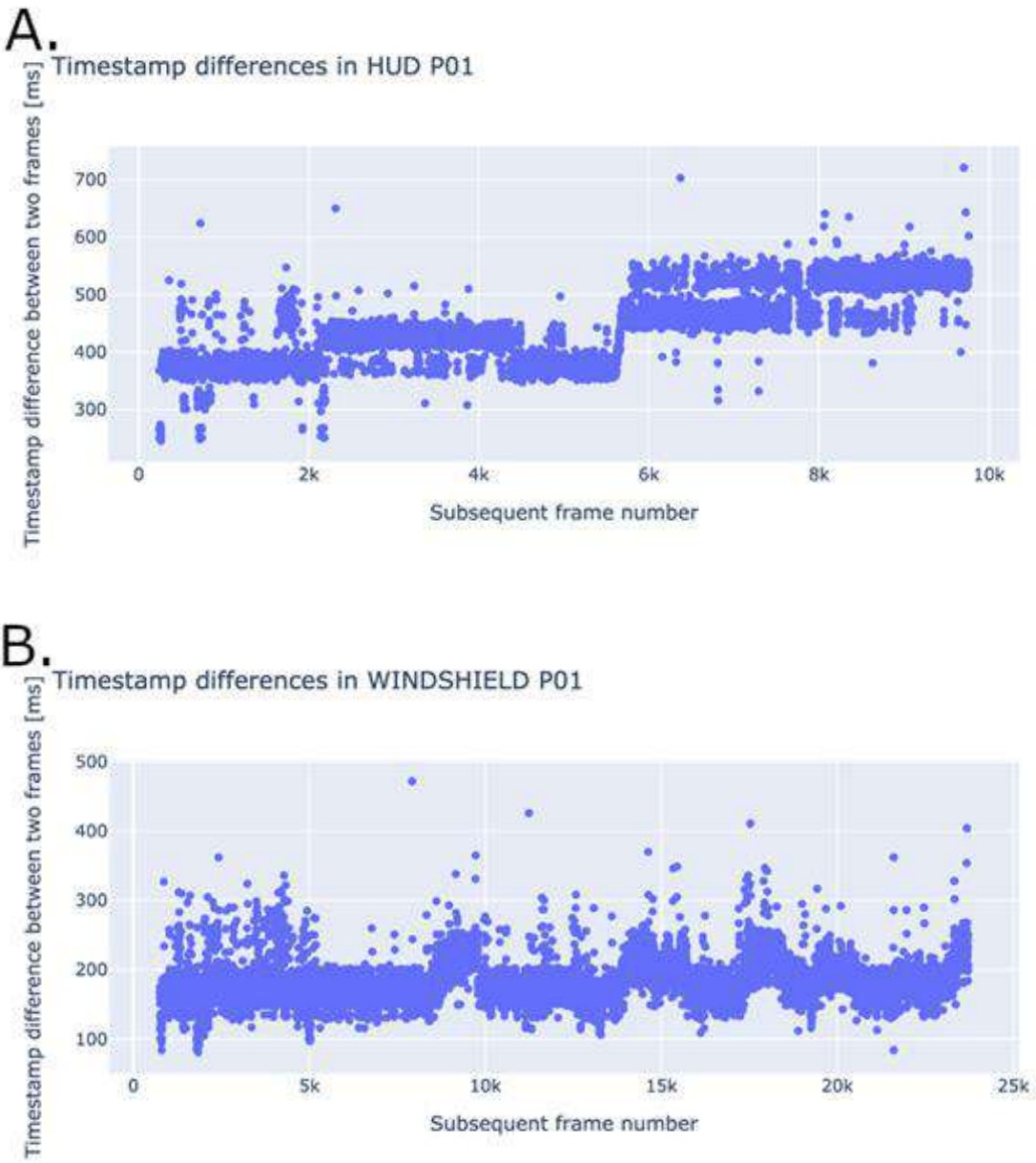


Figure 79: . Timestamp differences (in milliseconds) for phones in position C (5.A, top) and position A/B (5.B, bottom) for participant P01.

As can be seen in Figure 79, timestamp differences varied not only between different application runs (see A *vs* B, Figure 79) but also within every session itself. In extreme cases for P01, the differences reached above 700 ms. It should be further investigated what are the differences in data sampling, depending on

the phone positioning, and whether there are sampling fluctuations dependent on power or memory consumption of the phone.

4.1.5.3.3 Heart activity

For each participant monitoring of heart rate activity was conducted with the use of the smartwatch (see *3.3.2. Heart rate monitoring* for details). For each participant chart of heart rate estimation at any given moment of the test drive was created. Since the wearable device tended to disconnect resulting in estimating heart rate as constant or equal to zero, the data collected for the variable *Heart rate* in JSON files seems to have limited reliability. Examples of such events can be seen in Figure 80.

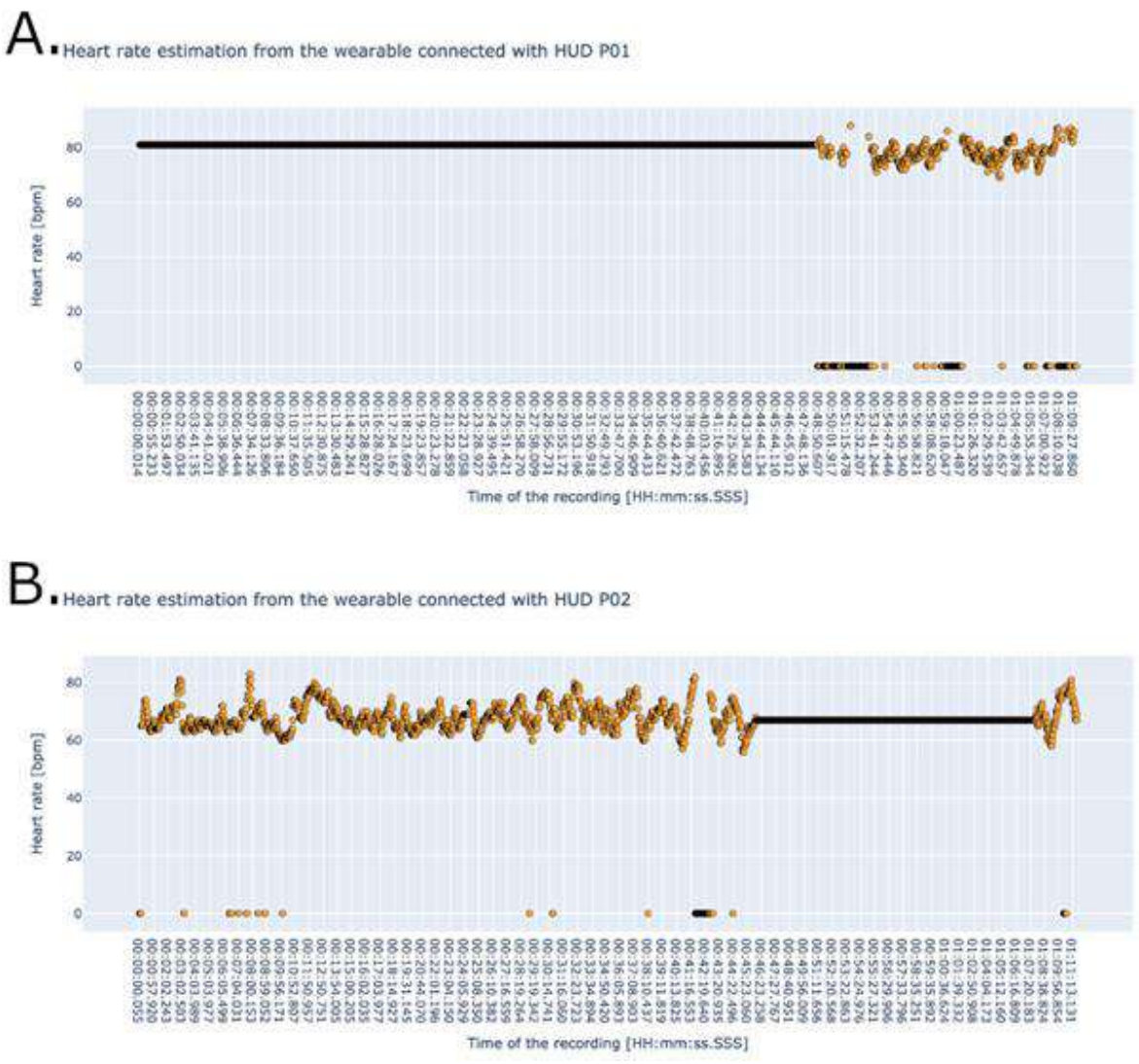


Figure 80: Heart rate (beats per minute) changes of participant P01 (6.A, top) and P02 (6.B, bottom) during the test drive.

In both cases, the wearable device was connected to the phone at the beginning of the drive and either did not work and needed reconnecting (P01) or was connected properly and stopped giving the correct heart rate (P02).

4.1.5.3.4 *Alert detection*

For each participant, graphical description of all the alerts was provided. In order to observe if, and eventually to what extent, differences could be observed between the two smartphone positions with respect to alerts' occurrences, the events were depicted in parallel (see Figure 81 as an example).

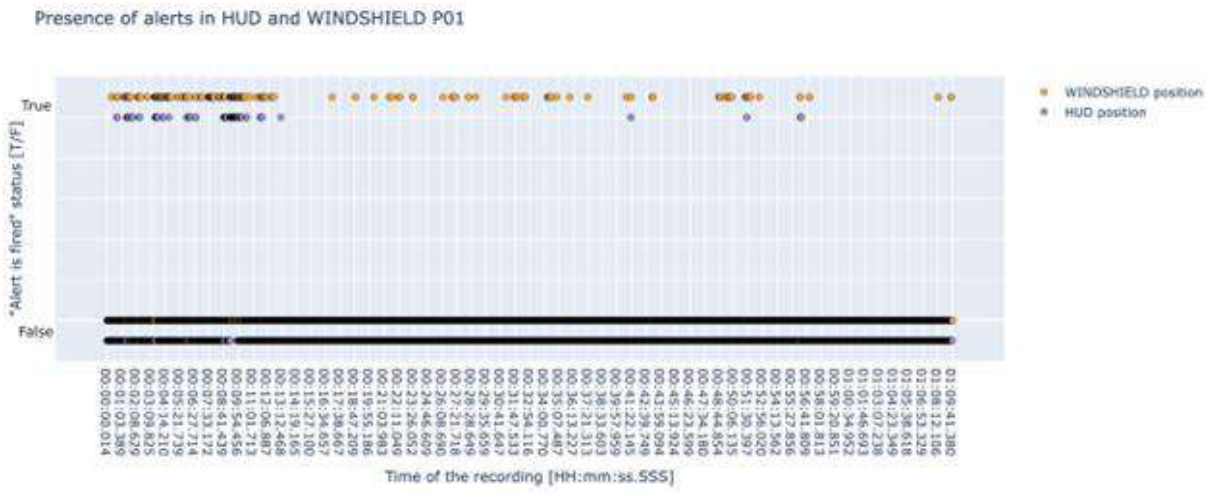


Figure 81: Comparison of the presence of alerts during the test drive for phones in position "Instrument cluster"/C (HUD) and position A/B (WINDSHIELD) for participant P01.

As can be seen in Figure 81, for participant P01 alerts occurred significantly more often on the phone in position A/B ("Windshield"/ "Infotainment"), than in position C ("Instrument cluster"). This was a tendency that occurred in charts for all the participants. This observation indicates that recommended position (being position "Instrument cluster") is in fact the position in which the application works optimally and placing the phone in any other position will impede the application's performance. However, further steps need to be taken to ensure that the application performs well in other placements than "Instrument cluster", as such positioning has its limitations, e.g. potentially being distractive in the center of the driver's field of view, or being for drivers a less preferred location for a smartphone than windshield or infotainment are.

4.1.5.3.5 *Distraction detection*

For each participant comparison of the alerts and the "distraction status" have been performed. The output JSON files contained two variables about the driver's distraction. These were *distractionLeft* and *distractionRight* variables, indicating the presence and direction (left/right) of distraction. Both variables could represent *True* or *False* for each data sample. For each test subject, two graphs were made (one from each phone) showing at which point in the test drive these variables had *True*/*False* values, and when the alert notifications happened.

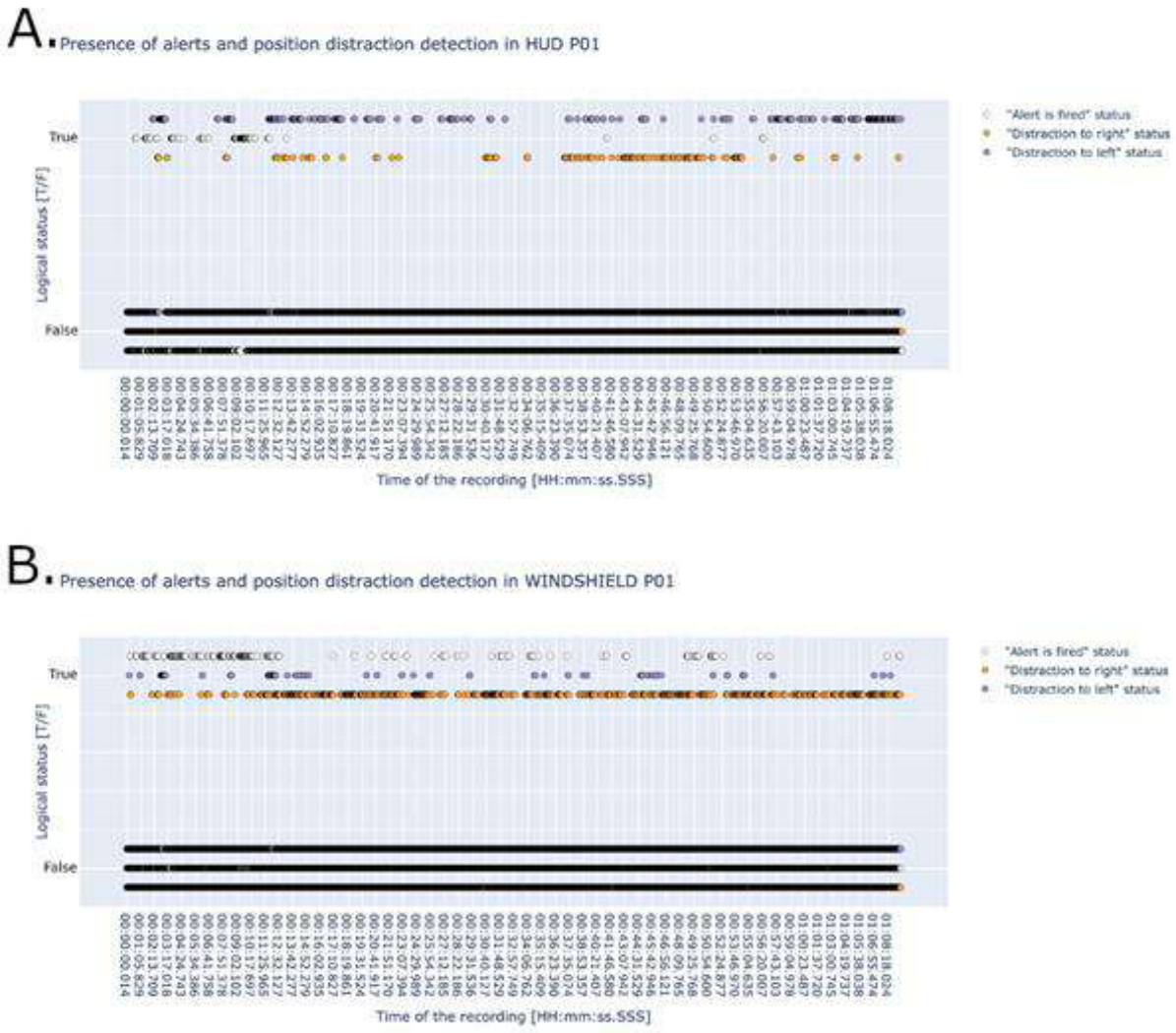


Figure 82: Presence of alerts and changes of the distraction statuses during the test drive for phones in position "Instrument cluster"/C (9.A, top) and position A/B (9.B, bottom) for participant P01.

4.1.5.3.6 Markers of drowsiness

For each participant comparison of the alerts and the potential markers of drowsiness have been conducted. The output JSON files contained two variables about the driver's drowsiness – *yawning* and *eyes_closed*. Both variables could take values of *True* or *False* for each data sample. For each test subject, two graphs were made (one from each phone) showing at which point in the test drive these variables had *True*/*False* values, and when the application's alerts occurred.

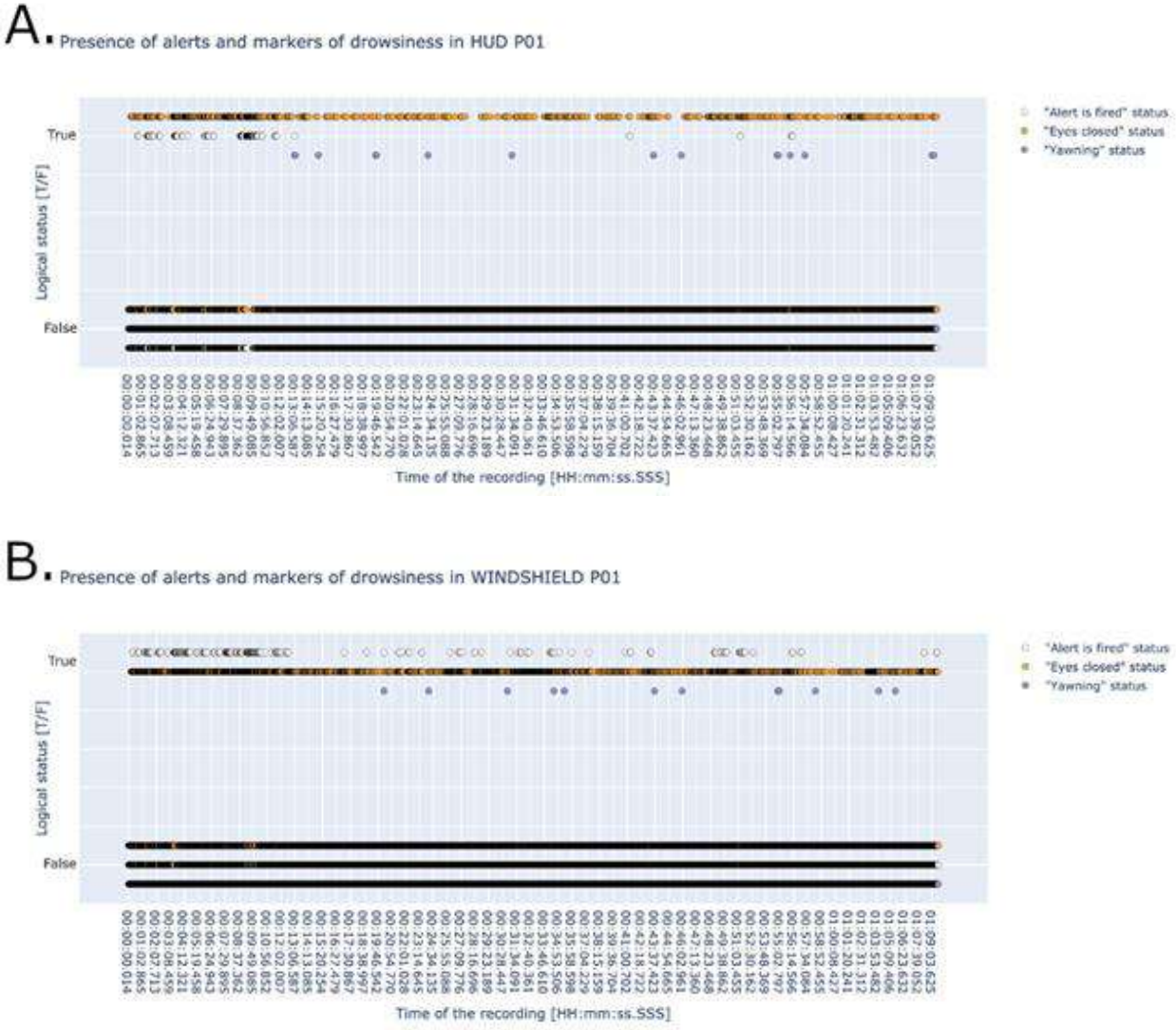


Figure 83: Presence of alerts and changes of the drowsiness markers ("Yawning"/ "Eyes closed") during the test drive for phones in position "Instrument cluster"/C (10.A, top) and position A/B (10.B, bottom) for participant P01.

As can be seen in Figure 83, "Eyes closed" status occurred more often on the phone in position A/B („Windshield"/"Infotainment") than on the phone in position "Instrument cluster"/C. This tendency was repetitive throughout all the participants' charts. "Yawning" status had a similar incidence on both phones, although surprisingly it occurred relatively frequently at different moments on both phones.

4.1.6 Conclusions

Concluding the above analysis, it should be noted that the tested DMS application, which is currently under development, works properly in a controlled environment. During the naturalistic driving scenario, its performance could be corrected in several areas, such as the absence of solutions other than the RGB

camera or false positive alerts and notifications, but these are shortcomings that can certainly be offset and compensated for in further phases of application development.

The application works as intended in largely correctly recognizing some visual signs of drowsiness, such as closing eyes and yawning, and signs of distraction, such as turning the head away from the driving direction. As such, the tested DMS application is a valid component of the CPSoSaware project, as well as, after necessary modifications and improvements, an independent application marketed to individual customers. The DMS application for smartphones is also a valuable chance to promote driver monitoring solutions among potential users and the public.

4.1.6.1 Future directions for the DMS application development

In this section, the directions in which the development of the tested application could take place are proposed and described. Each subsection refers to a feature of the application that could be modified or improved, and this process would be beneficial for the development of the entire application and its optimal performance. They include both the user's point of view (UX) and technical aspects.

4.1.6.2 Eyes opening

It was noted that the app determines 100% eye-opening not necessarily when the subject has a 'neutral facial expression'. Meaning that for some subjects, the app showed 100% eye-opening most of the time, even when the eyes were slightly closed. Given that the app's operation relies to some extent on eye markings, the algorithm should ultimately be refined and sensed enough to properly determine the level of eye closure.

4.1.6.3 Running in the background

The application does not save data when it runs in the background. From interviews with respondents, it seems that the average user during the drive wants to have a map opened or be able to answer incoming calls. Tests of the application indicate, however, that this is not possible in the current version, resulting in not generating alerts in such cases.

4.1.6.4 Application and smartwatch connection

The application and the watch happen to disconnect or improperly send the data, probably when the watch fails to read the heart rate. There have been instances of the application shutting down on the watch, as well as needing to reconnect. It would be beneficial for the application performance to report that the watch has disconnected, or that the connection is faulty. In the current version of the application, the only indicator that the connection is not working as it should be, is a constant heart rate that lasts for a long time or a heart rate at a constant equal to zero. However, to notice this the driver must focus on the phone while driving, which would be an inappropriate outcome. Optimally, the driver could receive a subtle, unobtrusive feedback from the application about a failing connection to the watch. Data packets' reception and a proper report in the JSON file could be improved to manage the troubleshooting phase during development. Moreover, it could be taken into consideration whether, at least during the developmental phase, it would be more suitable to keep a constant sampling rate in order to check further

for occurrence of e.g. false positive or true negative events, and calculate properly metrics of sensitivity and specificity, while compared e.g. with manual data annotation or behavioural labelling.

4.1.6.5 Notification and alarm

During test drives, notifications tend to occur several in a row at very short time intervals. To prevent such situation the application may have limits on the number of times the alarm can occur in a unit of time. In interviews with test subjects, it was mentioned several times that the numerous pop-up notifications in the centre of the field of view (such as “Instrument cluster” position) may be very distracting while driving. The wake-up sound used in the application seems to be too obtrusive as well and, in the long run, it may be a nuisance and an additional source of distraction. Similarly, the text messages that appear could be more subtle in their delivery. Perhaps a more unobtrusive audible signal would suffice (e.g., when alerts are repeated in rapid succession, they could gradually increase in intensity), and a visual signal (e.g. an icon that is self-explanatory – a coffee cup or a simple exclamation point) instead of a text notification.

4.2 Cyber Security Use Case

The automotive sector is going through a significant development trend regarding autonomous driving. Sensors, communication systems, actuators, etc., are some of the components becoming more and more common in vehicles due to this. This has led to more complexity, which has increased the attack surface and hence the number of ways cyber-attacks can occur. This leads to attackers having access to the vehicles from outside. Several security-related projects have been conducted as a result of these concerns. Bold initiatives are being taken by technology giants, automobile manufacturers, and governments all over the world to build safer and more affordable AVs and bring them to market quickly. Cooperation is essential for tackling the issue of cybersecurity. Based on the different levels of details, these attacks were classified and presented by Florian Sommer et al. [3].

In the following subchapters, the project shows different types of possible attacks and how these can be mitigated. Starting with a scenario, in which the communication through levels is shown and how the system is used in different ways to defend against an attack, based on contextual information.

After that we have some other chapters showing different scenarios, *Cyber attacks* on the camera sensor and *Cooperative awareness scenario*, in which we zoom in and show a more localized view of the system for simplicity, but the reader should understand that these can be also integrated in the generic architecture presented in the first scenario.

4.2.1 Context Aware Defence Against Cyber Attacks

The use case describes a situation in which a car gets a cyber-attack. The attack could be targeted to only one car, all the cars in a certain area (toll area/tool booth), or all the cars with a device from a certain manufacturer. In this lab demo, the project showcases how, using the architecture proposed by the project Figure 86, a vertical can mitigate these possible situations. But this architecture can be adapted to different situations.

This demo will showcase event detection and how the system reacts by executing corrective actions. Also, this demo will demonstrate the intercommunication mechanism between the different layers (vertical communication) and also between the different CPS in the same CPS layer (horizontal communication).

4.2.1.1 Demonstration Setup

The demo follows the CPSoSAware architecture, with the different layers introduced in this deliverable in chapter 2. Since being a lab demo, some of the components are distributed at different locations at partners premises.

The diagram in

Figure 84 depicts the high-level architecture of the solution and where each of its components are physically located.

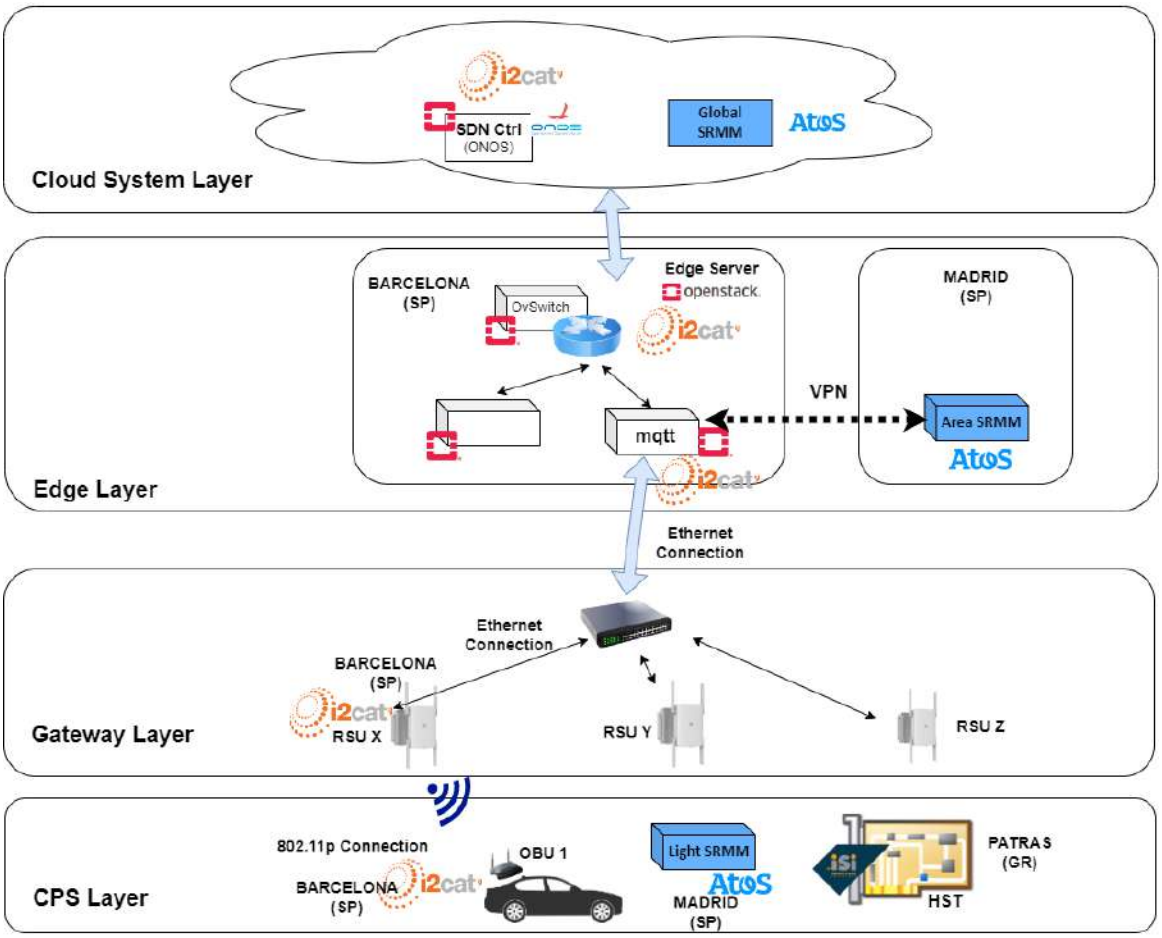


Figure 84: CPSoSAware architecture and partner premises

The following images, grouped by layer, show some of the real components and their interfaces:

- Cloud System Layer: It contains an SDN controller in charge of controlling the flows and the traffic at the Edge Layer in the Edge Node (
- Figure 85, Figure 86, and Figure 87. In this case, the chosen SDN controller is ONOS, an open-source controller widely used in industry. In the same Cloud System Layer, we find the Global SRMM instance (Figure 88), which collects events from global sensors and alarms from lower SRMM. It is in charge of global context intelligence as it receives information from the whole system and has an overall picture of the system.

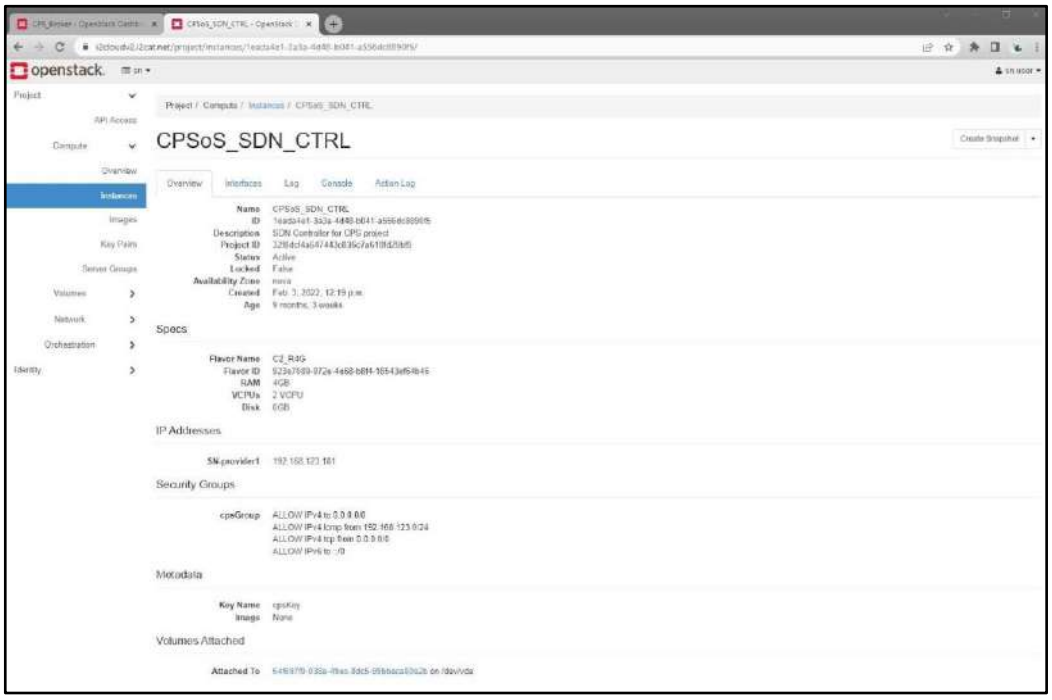


Figure 85: SDN controller, virtualized at the Cloud System Layer.

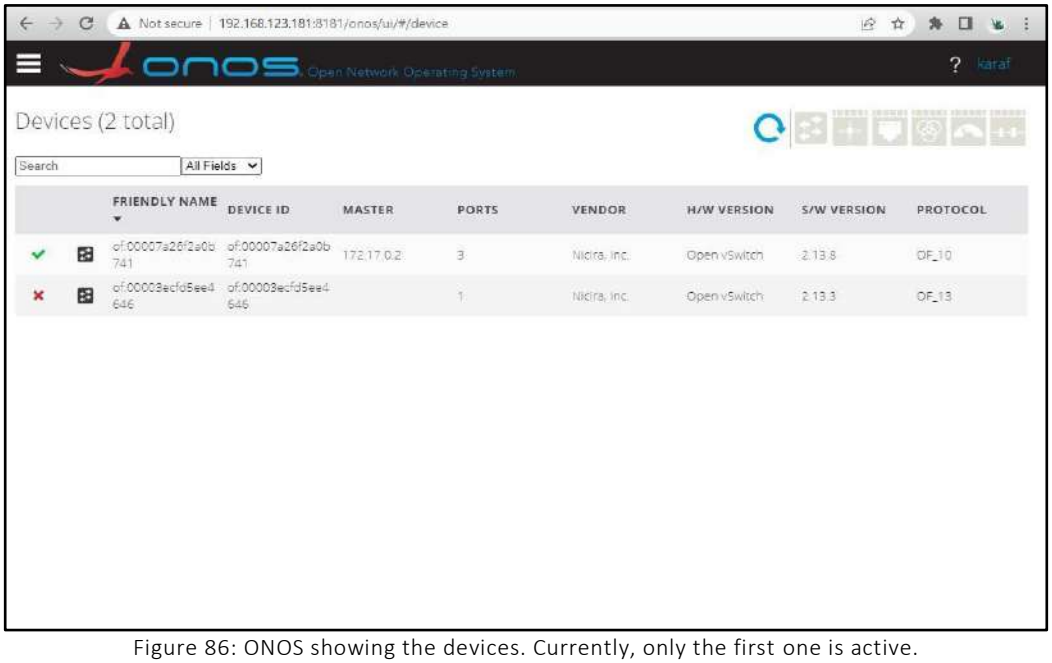


Figure 86: ONOS showing the devices. Currently, only the first one is active.

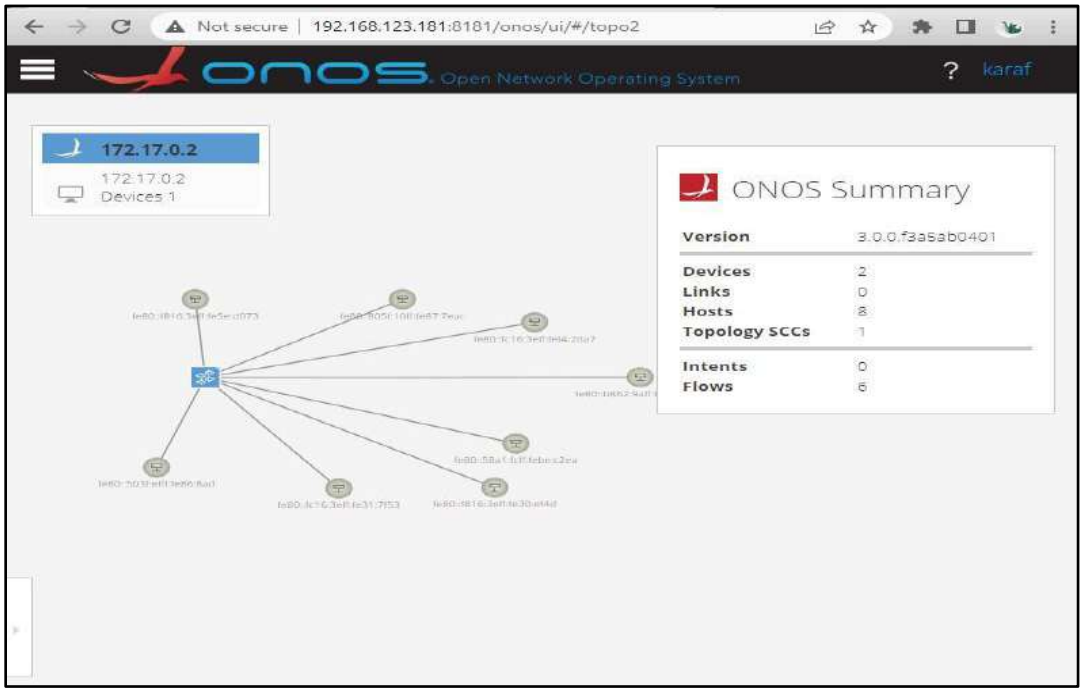


Figure 87: ONOS Showing the topology. OVS switch with all the hosts connected to it

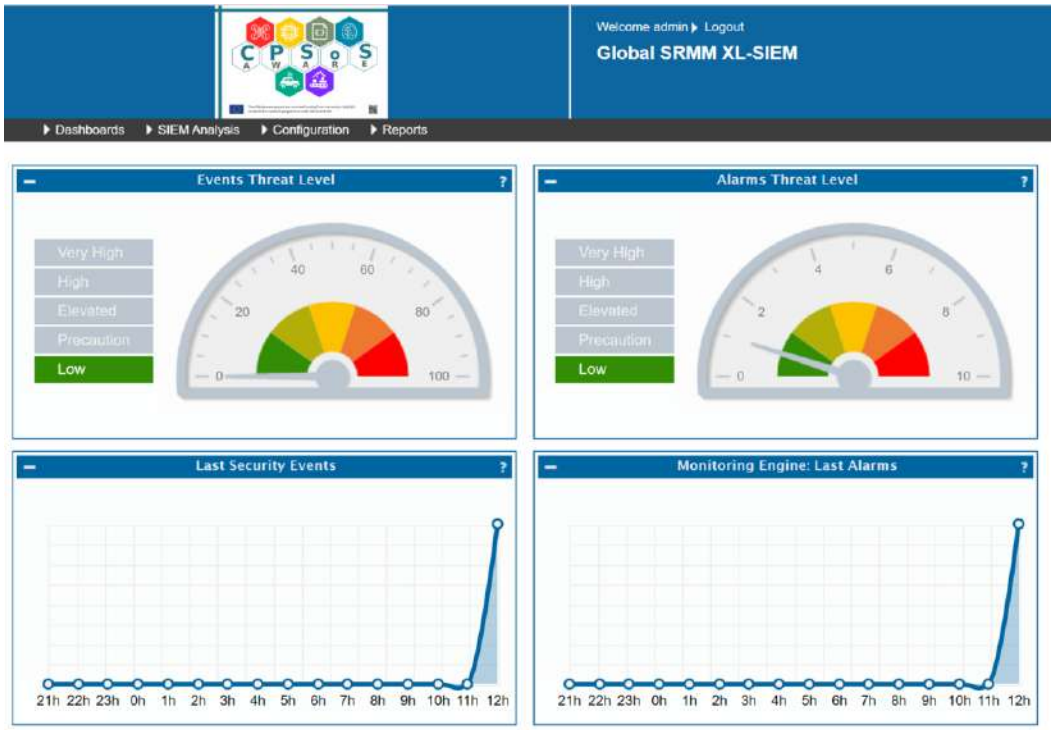


Figure 88: Global SRMM, executive view

- **Edge Layer:** It contains the Edge Node, an Intel based server (Figure 89), in which Openstack has been deployed to benefit from virtualization and guarantee tenant isolation in case we would have several verticals sharing the same Edge Node (Figure 90 and Figure 91). On that Edge Node, deployed as a Virtual Machine (VM), we have a MQTT broker (Figure 92), which forwards messages to and from the RSU and an area instance of the SRMM in a different virtual machine. As the previous SRMM, the Area SRMM collects events from area sensors and alarms from lower SRMMs. It detects attacks in the area context, such as roads or streets, and sends alarms to other Area SRMMs and the Global SRMM.

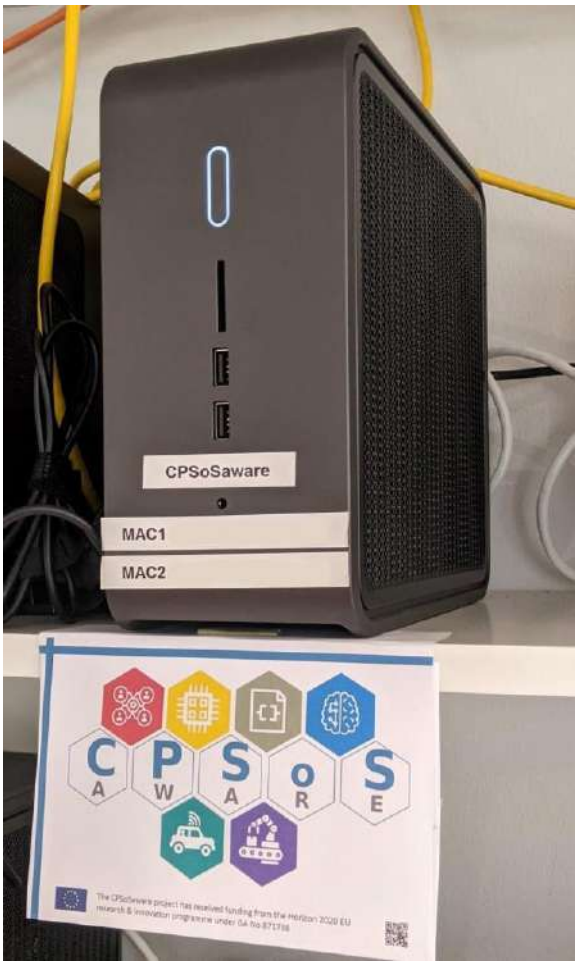


Figure 89: Edge Server belonging to Edge Layer.

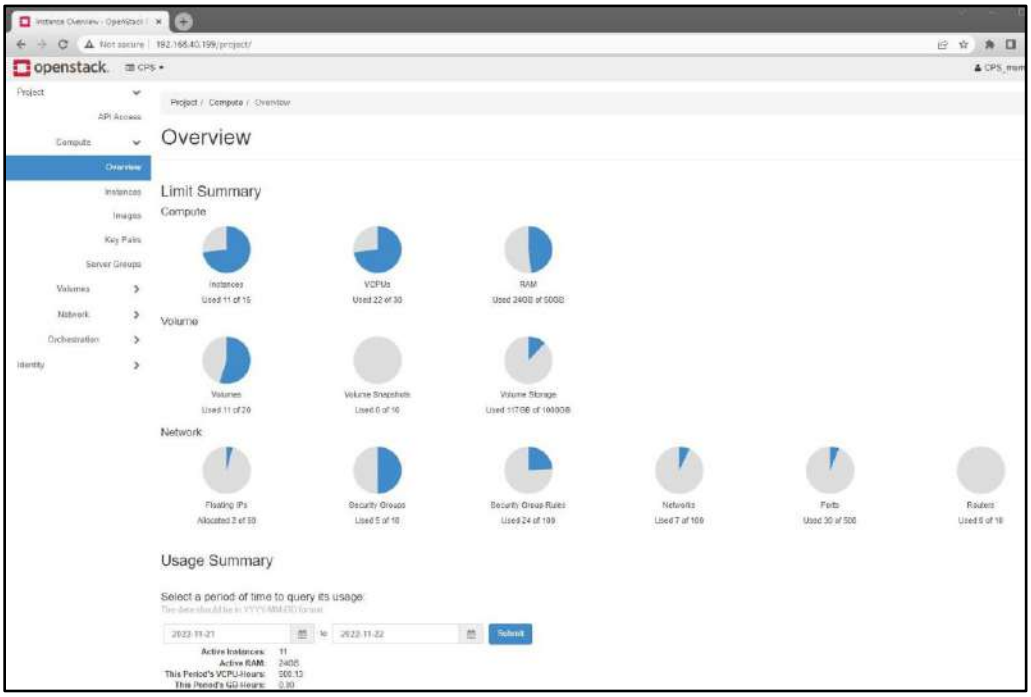


Figure 90: Resources at the Edge Node.

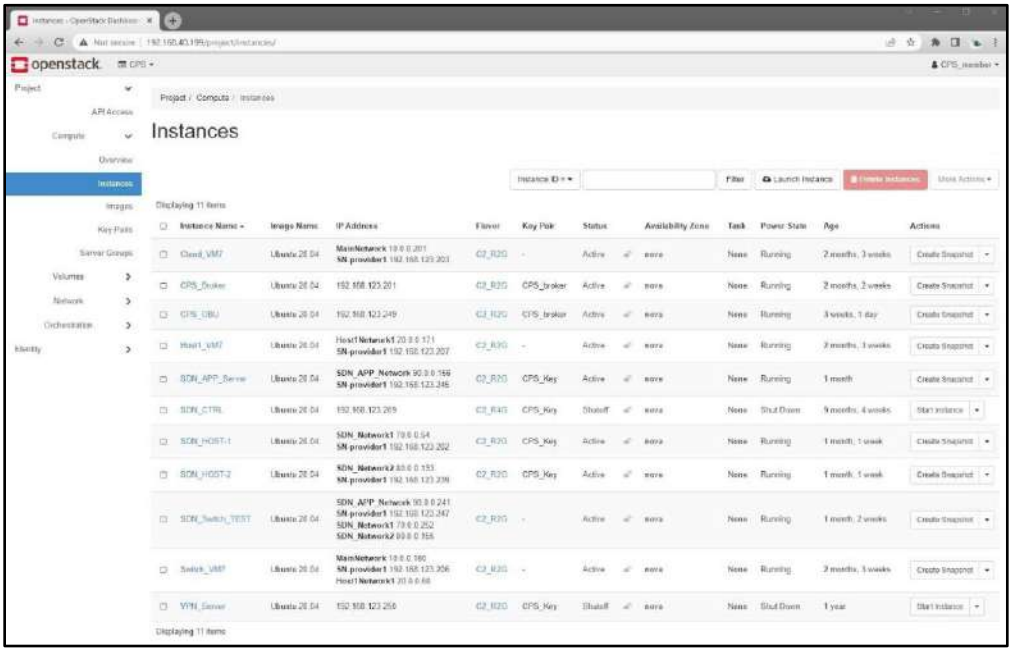


Figure 91: Instances and hosts running on the Edge Node.

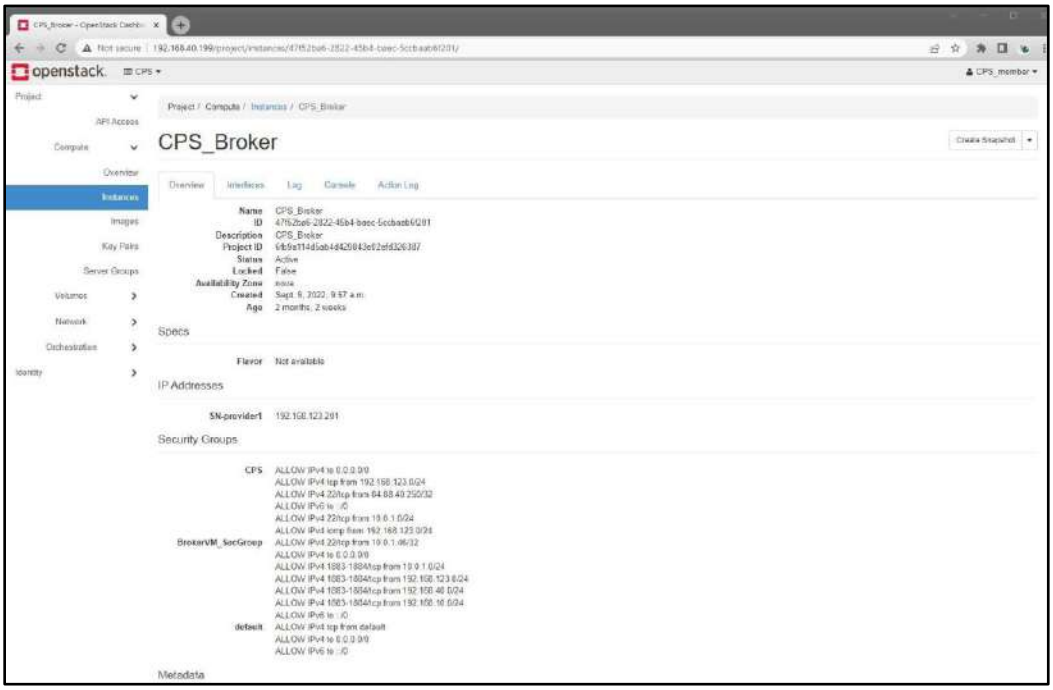


Figure 92: Detail of the MQTT Broker running on the Edge.

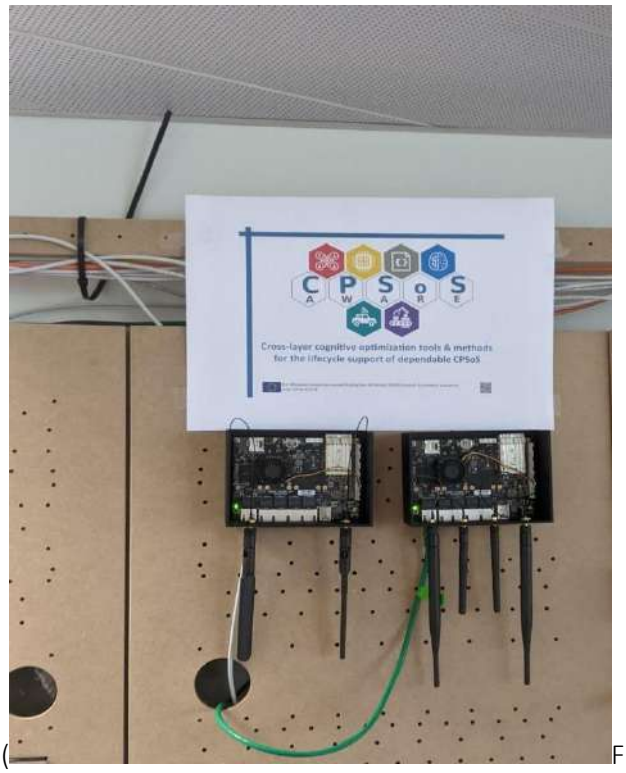
- **Gateway Layer:** The Edge Layer is connected to the Gateway Layer in which the RSUs are deployed. As mentioned before, the RSU is the element which implements the radio interface that enables the infrastructure to be connected with vehicles (

Figure 91). The communication between the

RSU and the elements in the Edge Layer is performed by a MQTT broker. Therefore, when a V2X message is published into the MQTT broker by the RSU, the broker pushes this message to the Area SRMM, which previously has been subscribed to the specific topic. The downlink communication works in a similar manner, the SRMM publishes a message, the RSU receives it through the broker and transmits it to the vehicles. The communication between the Gateway Layer and the CPS Layer is implemented using IEEE 802.11p.

- CPS Layer: It is the lowest layer of the architecture, in which the different CPSs are located. In our case, the CPSs are within the vehicles. They have different functions: OBU (On Board Unit) connects the other devices with the infrastructure and/or nearby vehicles; Local SRMM (Figure 94) collects events from other devices (CPSs or sensors), correlates them and raises alarms, which are sent to other SRMMs through the OBU, when it detects an attack; and Hardware Security Tokens (HST) acting as trust anchors in each car, check validity and install valid versions of firmwares.

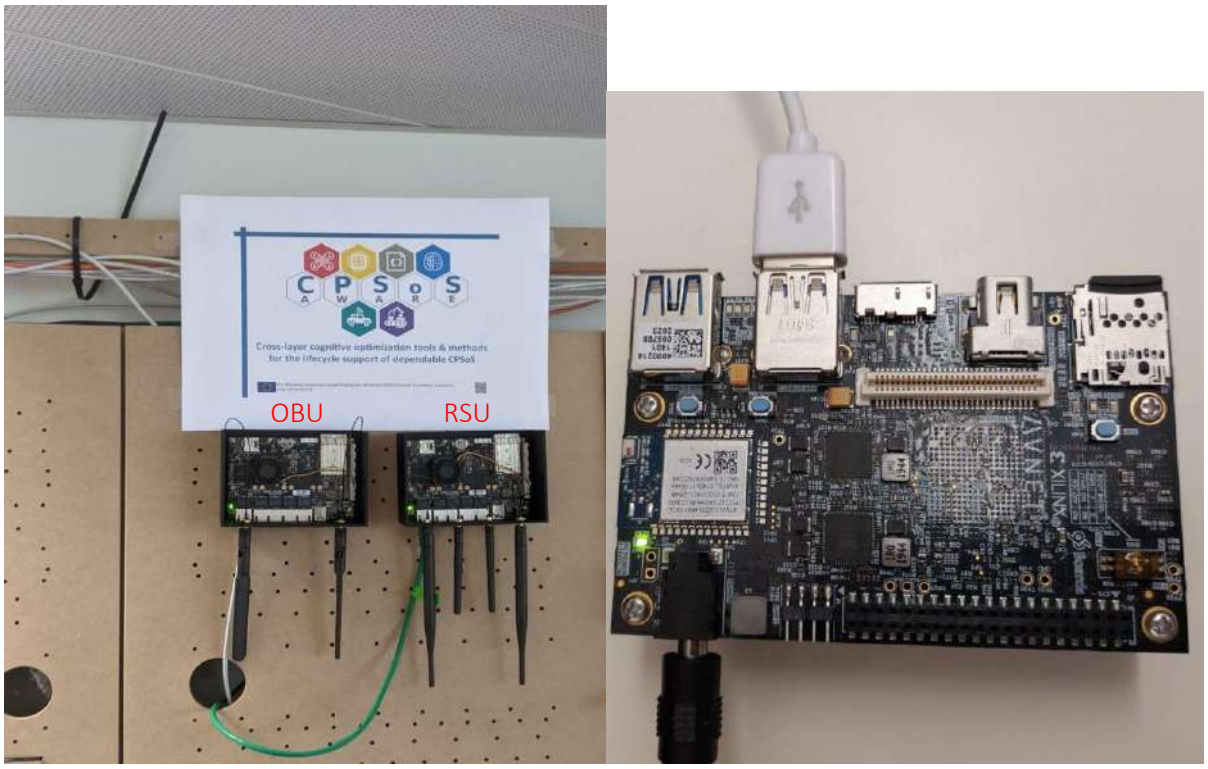


Figure 93: OBU and RSU (left) implement the IEEE 802.11p communication channel between the infrastructure and the vehicle. The RSU is connected Ethernet to the Edge Node. The HST (right) performs security checks and through its security sensors checks for security issues (eg. Firmware validity)

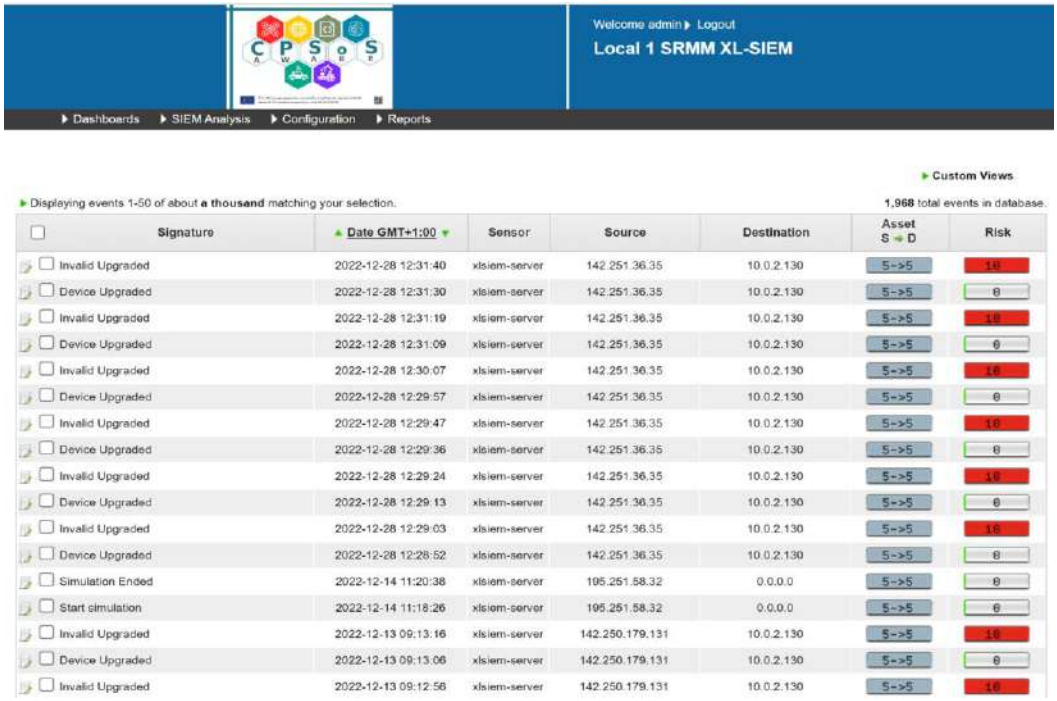


Figure 94:Event view of Local SRMM

4.2.1.2 Scenarios

There are three scenarios, representing the behavior of the system depending on different contexts.

Scenario 1: Defence against attacks in a local context

The first scenario shows the behavior of the system in a local context, inside a vehicle. For this example, we assume that the car has a couple of vulnerable devices. The first one may be a communication device, for example a Bluetooth, that directly forwards messages to the internal communication bus. While the second device accepts any setup command from the internal channel without verification or authentication. Figure 95 presents the sequence diagram of an attack, and Figure 96 shows the alarms raised by the Local SRMM. The numbers on the left part of the picture (outside the picture) map the alarms to the events in the sequence diagram.

The use case begins when an attacker sends a message to bluetooth and bluetooth forwards it to the second vulnerable device (event 1). This message orders the device to upgrade its firmware from a remote repository with a malware-containing firmware. This process ends with a *Successful upgrade* message in the log. SRMM agent identifies this message and sends to the Local SRMM a security event with the relevant information (event 2). Figure 97 shows the content of this security event. This produces a low risk *CPS_Device_Upgraded* alarm, which has an associated action that requests validation of the installed firmware to Hardware Security Token (HTS) (event 3). The result of this process is written to the log, which in turn is parsed by the agent and sent to the Local SRMM (event 4). As the installed firmware is not legit, the validation results in an ERROR and the SRMM raises a high-risk *CPS_Invalid_Upgraded* alarm. It is sent to the OBU which forwards it to the RSU to reach Area SRMM (event 5), (more details in following

scenarios). At the same time, the associated action requests the HST to install the last known safe firmware version (event 6). When this process ends, a new *Successful upgrade* event is written to the log and sent by the agent to Local SRMM (event 7). The generated low-risk alarm requests a new validation (event 8). But, in this case the validation is correct (event 9), and no further action is required.

This scenario demonstrates:

- SRMM detects sequences of events that represent a threat to the system
- SRMM executes the actions associated with alarms and the associated actions to mitigate the attacks
- HST capabilities to validate the firmware and to install the last legit version of the firmware

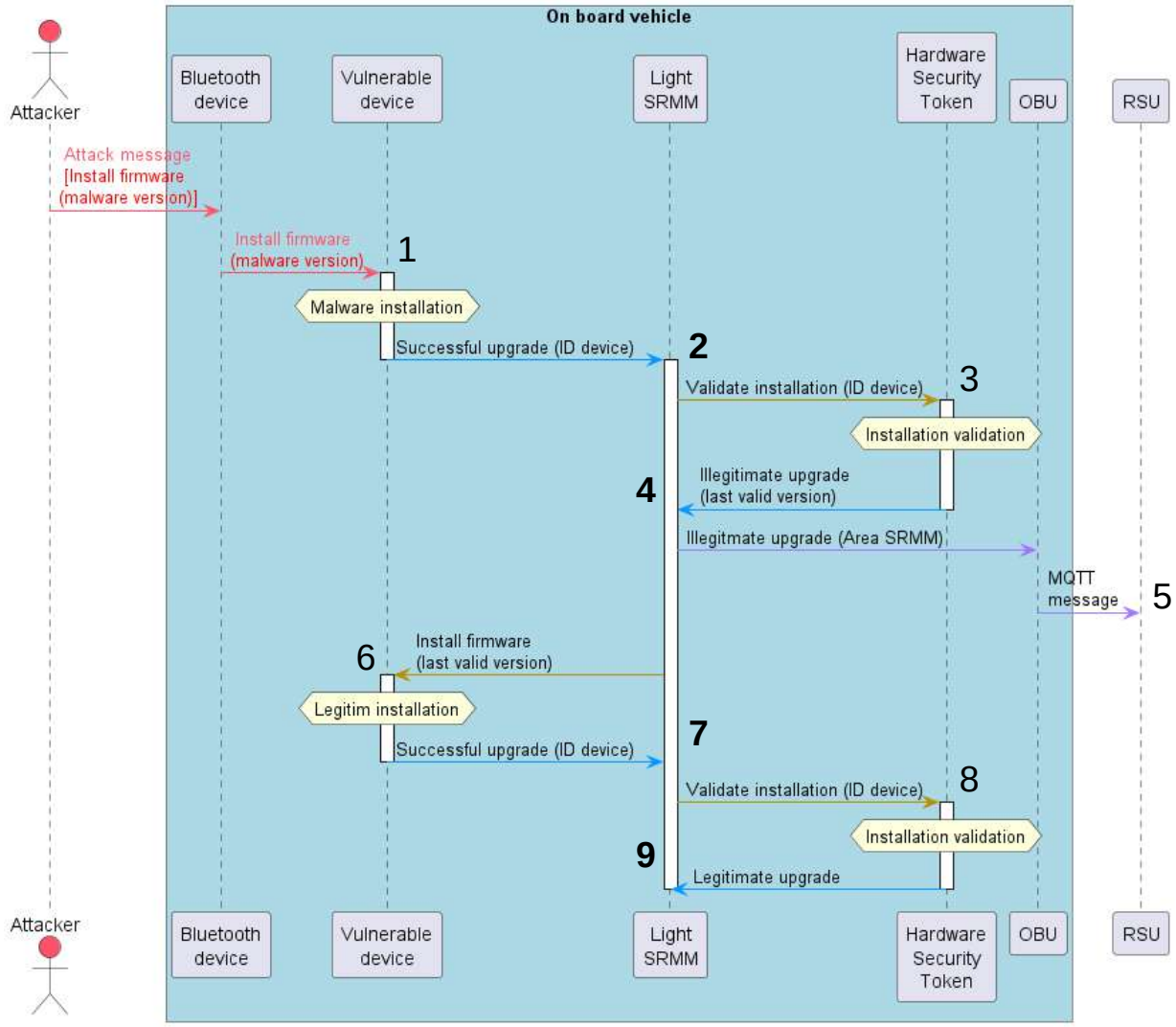


Figure 95: Local context scenario

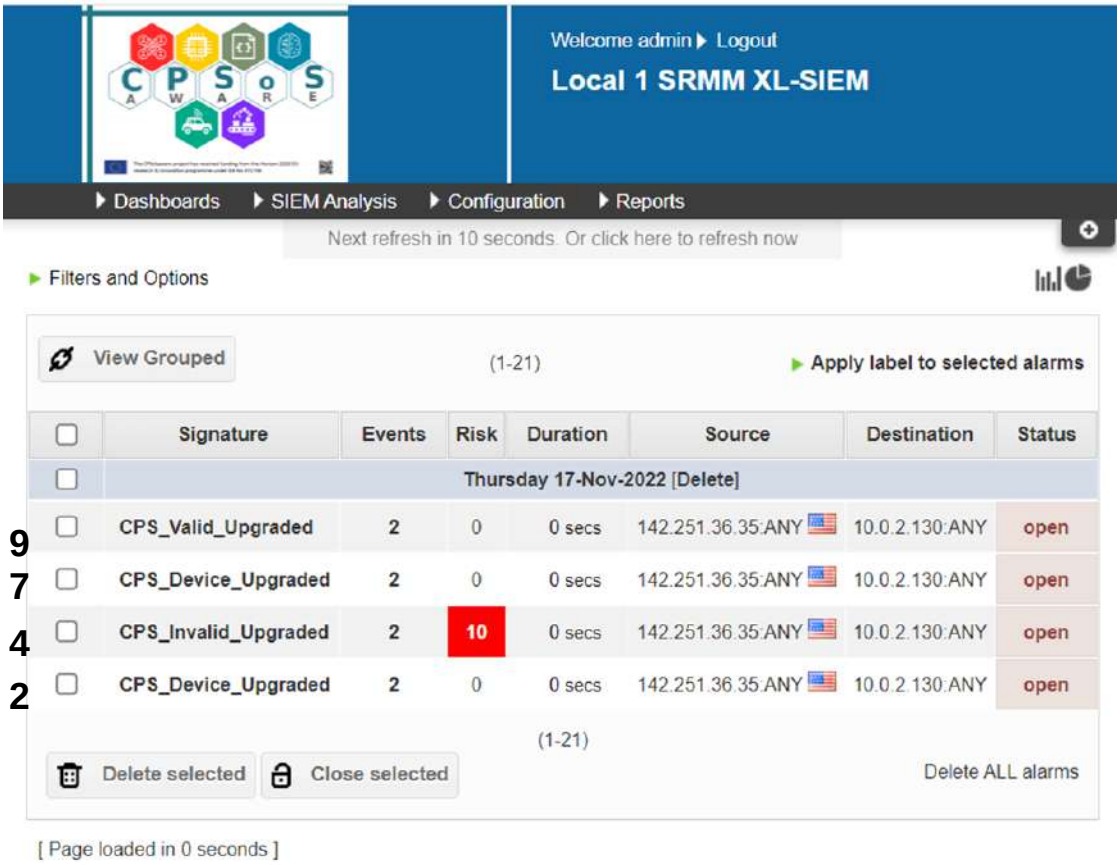


Figure 96: Alarms raised by Local SRMM

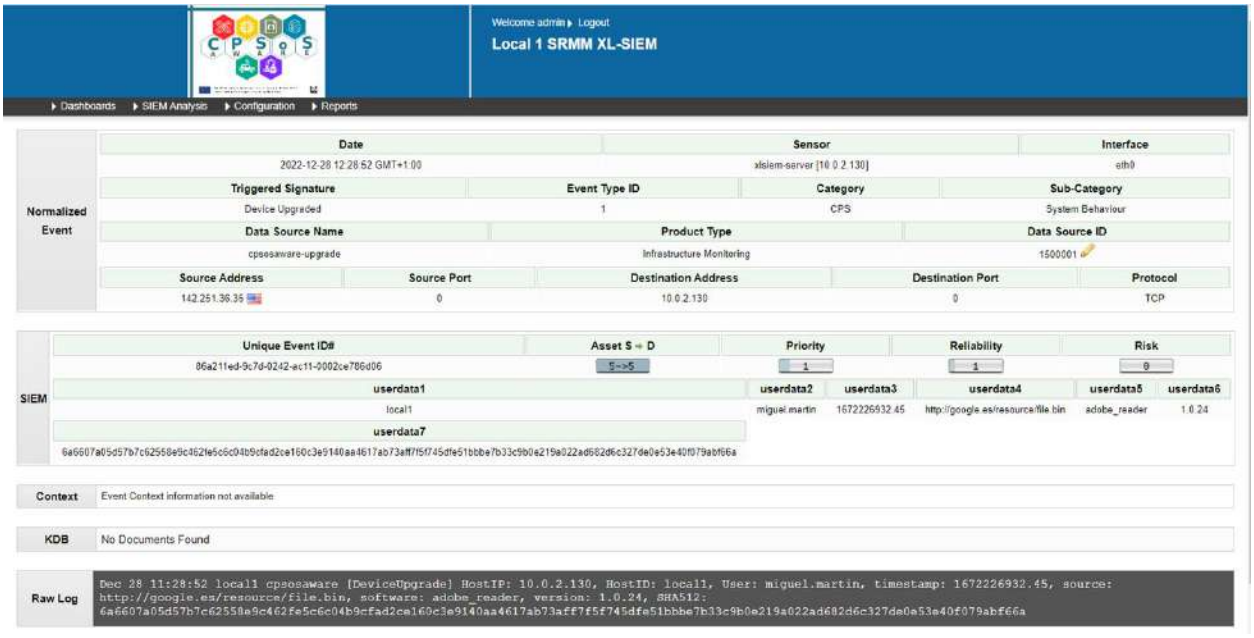


Figure 97: Successful upgraded security event

Scenario 2: Defence against attacks in a area context

The second scenario shows the behavior of the system when an attack occurs on different vehicles within an area, a limited zone such as a road or a city, where the Local SRMMs report to Area SRMM. Figure 98 presents the sequence diagram for the area context scenario while Figure 99 shows the alarms that the Area SRMM raises. Successful attacks are mitigated locally by the Local SRMMs but, to prevent further attacks, the Area SRMM blocks communication between the malware repositories and all vehicles in the area when it detects an area attack.

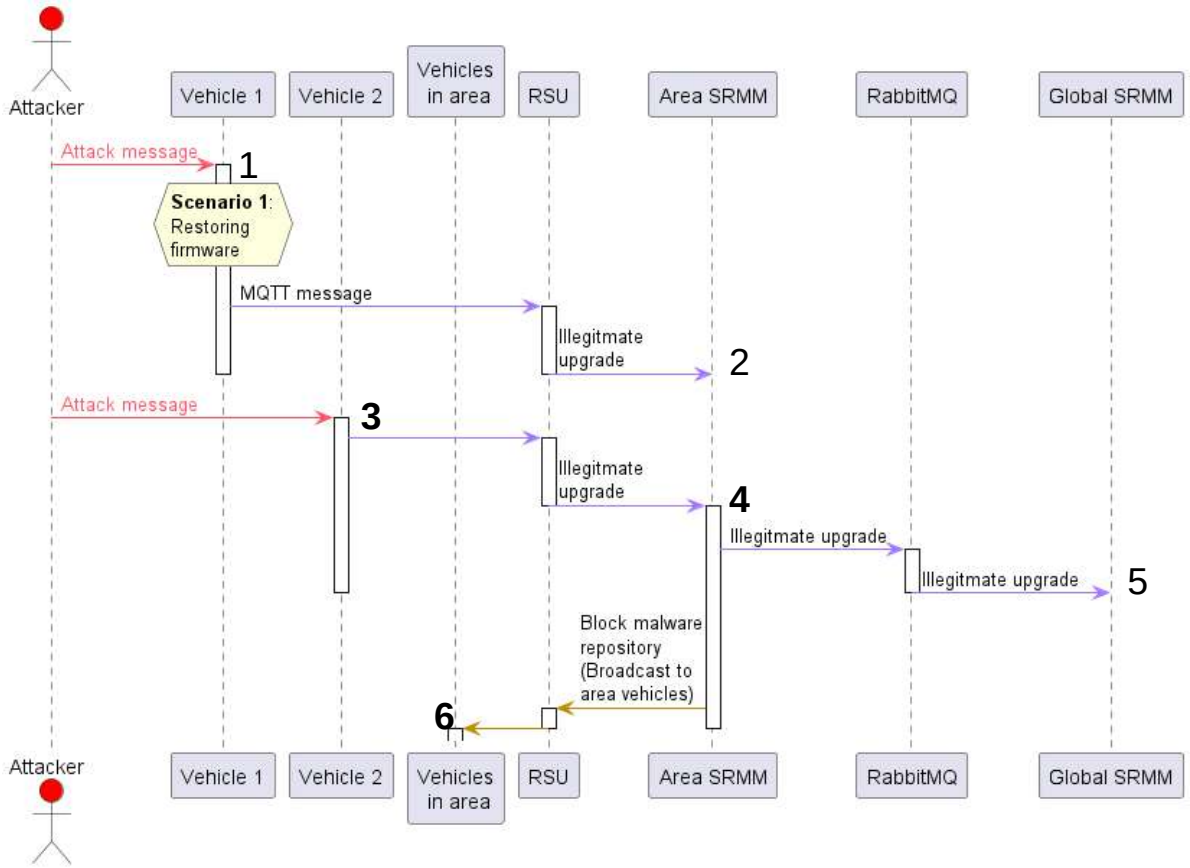


Figure 98: Sequence of events for area context attack

In this case, the pattern for raising an area alarm is having two attacks on different vehicles, (events 1 and 3). When a Local SRMM detects an attack, it mitigates the attack, reinstalling the legit/safe firmware (scenario 1), and sends the *Illegitimate upgrade* alarm to Area SRMM via the OBU. The OBU forwards the alarms to the RSU over radio connection and the RSU forwards them to Area SRMM at the Edge Node, (events 2 and 4). When the Area SRMM detects two local *Illegitimate upgrade* alarms, it raises an area *Illegitimate upgrade* alarm which, in this case, is sent to Global SRMM (event 5), and to all Local SRMMs that are within the Area SRMM. This alarm has an associated action that blocks the communication between all vulnerable devices and the remote repository within the area (event 6).

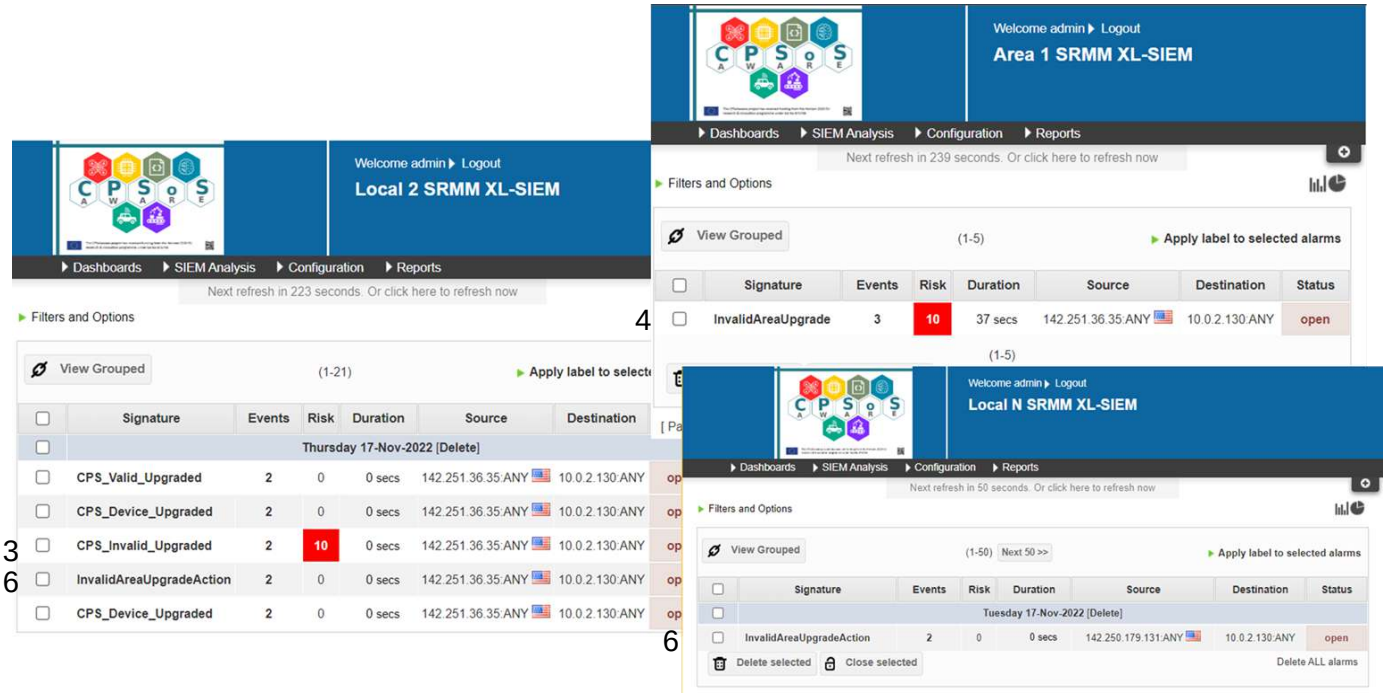


Figure 99: Alarm SRMM displays that show alarms generated when the attacks reach the area context

This scenario demonstrates:

- V2X communications, sending events from Local SRMMs to Area SRMMs and vice versa via the OBU-RSU-EdgeNode path
- Area SRMM correlates alarm from Local SRMMs
- Associated actions prevent further attacks

Scenario 3: Defense against attacks in a global context

The third scenario shows the behavior of the system in a global attack context, showcasing the Global SRMM which has an overview of the entire system. In this scenario, the Global SRMM receives events and alarms that affect the system as a whole. Figure 100 represents the sequence of events that occur in the global context, while Figure 101 shows the alarm views of some SRMMs, where the generated alarms are represented.

The scenario begins with two *Illegitimate upgrade* alarms occurring in two different Area SRMMs (events 1 and 3), and reaching the Global SRMM (event 2 and 4). The second alarm complies with the global rule, raising an *InvalidGlobalUpgrade* alarm. This alarm is broadcasted to all Area SRMMs (event 5), which forward it to all Local SRMMs (event 6), as *InvalidGlobalUpgradeAction.* These low-risk alarms have an associated action that blocks the communication between all vulnerable devices and all remote repositories used in any attack, preventing further attacks. At the same time, the Global SRMM opens a ticket with the incident details to create a report for manufacturers to develop a new firmware version that fixes the vulnerabilities and eliminates the attack vector.

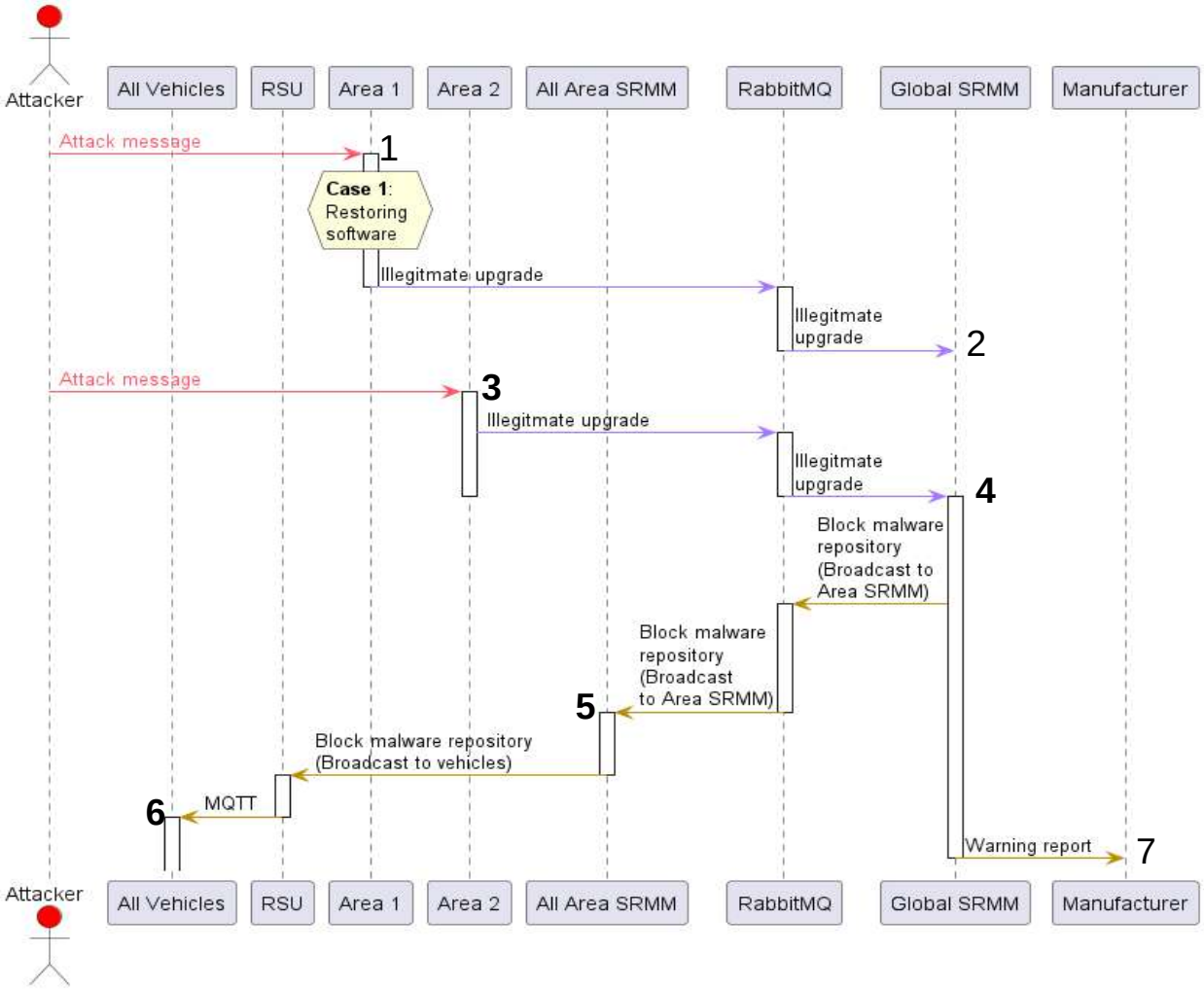


Figure 100: Sequence of events for global context attack

This scenario demonstrates:

- System sends events from Global SRMM to Area SRMMs
- Global SRMM correlate alarm from Area SRMMs
- Associated actions prevent further attacks

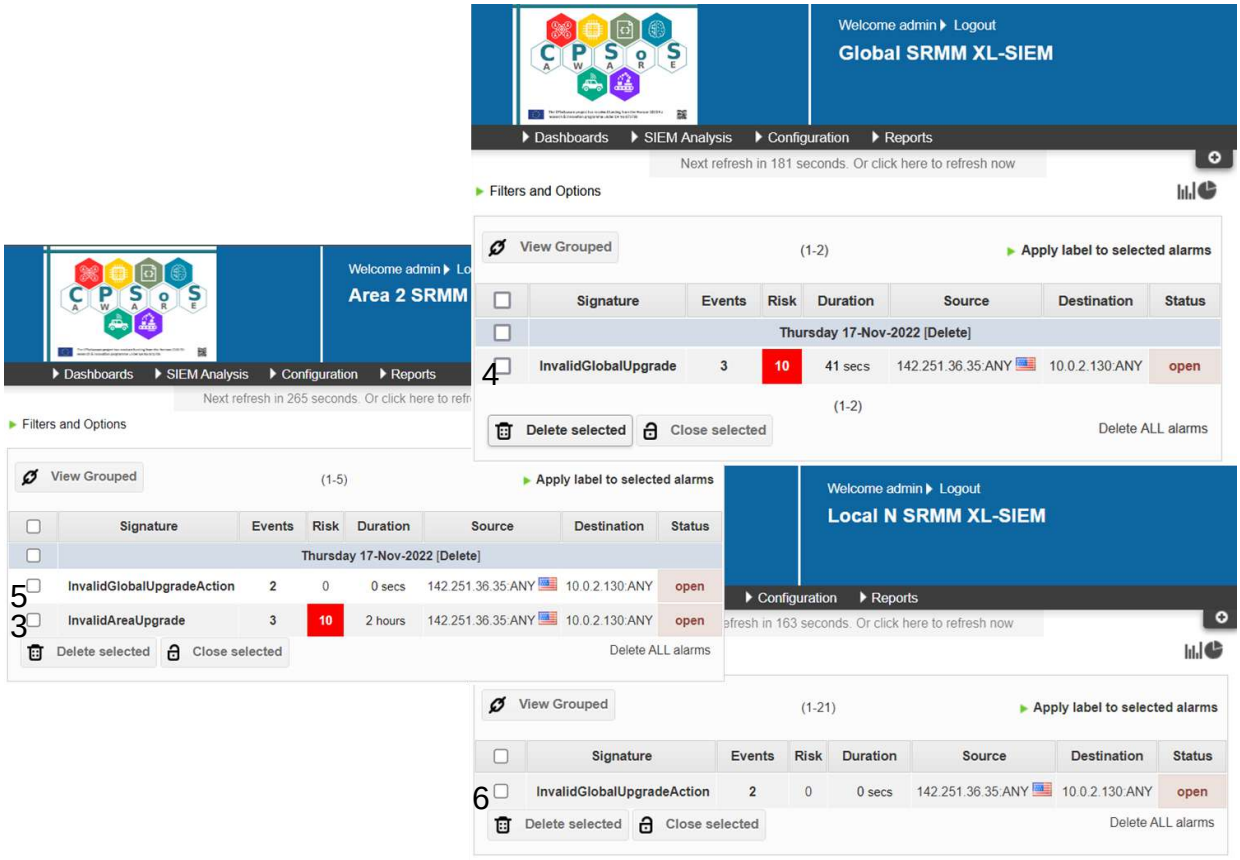


Figure 101: Alarm SRMM displays that show alarms generated when the attacks reach the global context

4.2.1.3 Network based attacks

The target of the scenario is to simulate the situation, when the V2X communication between the vehicles is under cyberattack. To properly model such situations, the simulation suite consisting of several modules integrated together was introduced.

4.2.1.4 Simulation setup overview

The simulation suite for network-based attacks is created by integration of the following components:

- RoSi simulator – central AV simulator providing sensor data, vehicle simulation and sensor simulation
- V2X Simulator – co-simulator responsible for modelling V2X communication. V2X message broadcasted by the vehicle contains the following data:
    - Location
    - Heading
    - Velocity
    - Perception engine output – list of all visible objects and their position
- Robotec_AD – Autonomous Driving software controlling the vehicles, utilising V2X data in case of cooperative awareness
- Cybersecurity module – module integrated with SRMM for sharing data used by Robotec_ad with the service responsible for cyberattacks detection. In this module, 2 attacks on V2X communication can be simulated: freeze attack and disturbance attack
- SRMM - component analyzing perception and v2x data to detect cyberattacks, by applying anomaly detection algorithm. If the attack is detected, the software stacks controlling vehicles being affected by the cyberattack are informed about the potential danger

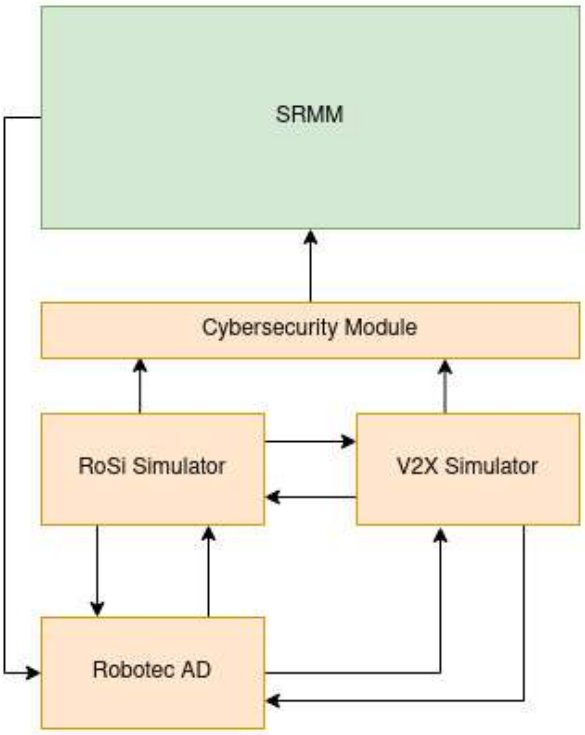


Figure 102: Diagram presenting integration of software components used in the simulation setup for network-based attacks simulation and detection.

The data sent to the SRMM module have the following structure:

- [RTC-V2X] - the message starts with the proper tag to distinguish it from the other messages processed by the SRMM
- “type” - there are two types of messages - “visible” and “v2x”. “Visible” is the simulation of a camera or lidar – there are vehicles in sight, seen directly by ego vehicles. This type can be understood as an output of the perception engine of the ego vehicle. “V2X” is the expanded awareness, there are vehicles within radio range with what they see. When the type is “visible” there are no “v2x_vehicles” field, only “visible vehicles”
- “timestamp” - timestamp in seconds
- “name”, “x”, “y”, “z”, “velocity”, “heading” - name, position, and movement of observed vehicle
- “v2x_vehicles” - list of vehicles within radio range
- “Visible vehicles” - list of vehicles in sight

4.2.1.5 Freeze attack

A freeze attack simulates a man-in-the-middle attack, when V2X data for one vehicle is not changing in time. The data received by other objects shows that the object under attack has unchanged velocity, heading, position and perception engine output, even when it is in reality moving.

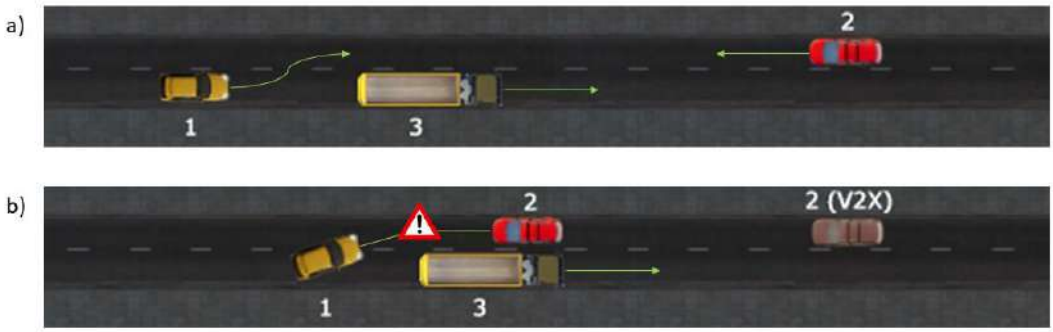


Figure 103: Example of a freeze attack. Vehicle 1 prepares to perform an overtaking maneuverer based on a V2X message received from vehicle 3. A freeze in the communication of vehicle 2 position can cause an incorrect decision of vehicle 1 and thus a possibly hazardous traffic situation.

4.2.1.6 Disturbance attack

A disturbance attack simulates a man-in-the-middle attack, when data for one of the vehicles is intentionally corrupted. The position transmitted via V2X communication interface is significantly different from the position calculated by the vehicle, which might create dangerous road situations if other traffic agents use V2X information for path planning or other AV/ADAS systems.

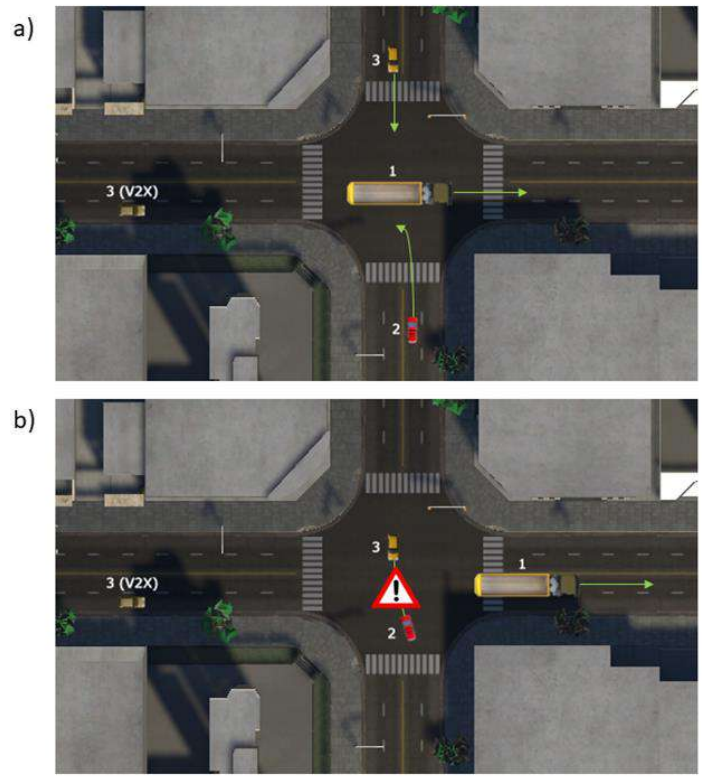


Figure 104: Example of a disturbance attack. Vehicle 2 prepares to turn left at the intersection. Vehicle 2 is not aware of vehicle 3 presence, because of limited visibility (vehicle 1) and vehicle's 3 significantly different position in the V2X message.

4.2.2 Cyber-attacks on the camera sensor

The demo scenarios executed as part of pillar 1 demonstrations involve a malicious user holding a remote control through which it attacks to the sensors data, disturbing both the quality of the sensor signal and the timestamps of the data recorded. As illustrated in Figure 105, the scene is interpreted in an undistorted way in the right-hand side, while the scene understanding is significantly degraded when the malicious intervention occurs between the scene and the perception engine (shown in the bottom right).

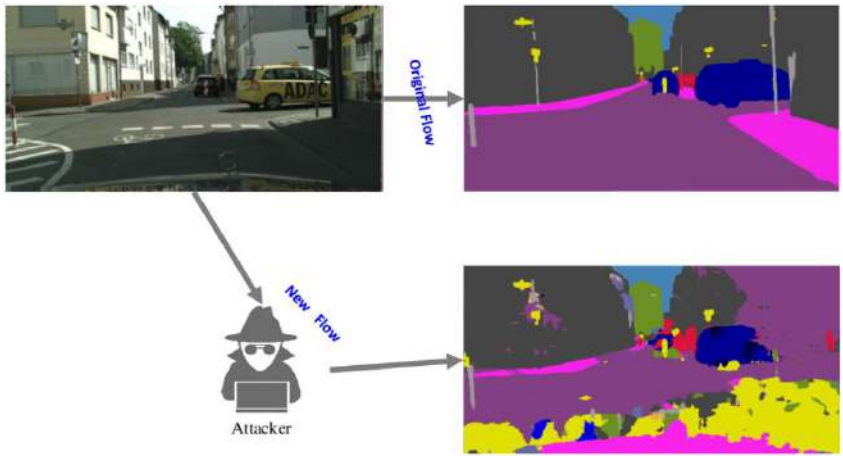


Figure 105: Schematic representation of the attack performed on the real vehicle. A malicious user attacks to the sensor data. Scene perception disturbance is illustrated on the bottom right.

	Use Cases- CyberSecurity Use Case
1	Attack on the Camera Sensor Layer: This scenario would involve a cyber-attack based on activating some malicious software which got installed during the software update process. Throughout this use-case the camera sensor could be attacked in a number of different ways, which could vary between adding noise lying on specific bands of the frequency spectrum/ introducing morphological deformations/ on the whole or parts of the image.
2	Attack on the Camera Sensor Layer by de-synchronizing the data: Throughout this scenario, the cyber-attack will be geared towards disturbing the association between the captured frames and the timestamp assigned to them. This will cause the failure of the perception engine, as all the architectural modules performing stochastic filtering on the scene observations will be affected by error. This use case should study the potential and the limitations of the cyber-attack detection and mitigation engine in assessing and recovering the failures.
3	Attack on the Camera Sensor by a remote agent: In addition to the scenario, the cyber-attack detection and mitigation engine will be used to detect and mitigate the camera signal distortion in the case that a malicious remote agent interferes with the test vehicle by knowing the IP of the processing unit and sharing some erroneous data. More specifically, this use case will assume that the remote agent sends via V2X communication: time zone/ daylight related data in order some sensor parameters (e.g.: gain/exposure time) to be tuned accordingly.

Table 17: Attack scenarios for the real vehicle demo as presented in deliverable D6.2

The types and the scenarios of the attacks were defined in D2.2 deliverable contributed by CPSOSAWARE's consortium at the beginning of the project and is also presented in Table 17 above.

The scenarios presented in Table 17 were used to investigate the potential and limitations of the cyber-attack detection and mitigation engine across a wide range of perception functions related to:

- Moving Object Detection
- Self-Localization
- Occupancy Grid Mapping/ Object Boundaries Definition
- Fully autonomous parking.

The number and the extend of perception engine components tested, exceeded the initial planning as this was specified in Wp2, WP3 and WP4. More specifically, while in the afore mentioned work packages it was planned to have the cyber-attack detection and mitigation engine tested only for use cases (1) and (4), we finally tested it across perception components related to object detection, navigation and automated parking.

The primary navigation of autonomous vehicles depends on the effectiveness of the sensor processing techniques applied to the data collected from various visual sensors. Therefore, it is essential to develop the capability to detect objects like vehicles and pedestrians under challenging conditions such as like unpleasant weather, poor illumination conditions, high speed, and motion on uneven ground. Thus all the scenarios discussed through sections 4.2.2.1-4.2.2.6.

4.2.2.1 Robustified cooperative awareness for mitigation and detection of cyberattacks

In a typical cooperative localization scenario, vehicles exchange and receive messages from other vehicles in order to re-estimate their positions. More specifically, using the module of AI accelerated multimodal scene analysis and understanding, vehicles extract relative measurements towards nearby vehicles and establish V2V connections. Through relative measurements, each vehicle formulates its differential coordinate and transmits it to a fusion center. The latter in turn receives all these differential coordinates from vehicles of the cluster and along with V2V connections, formulates a linear least squares optimization problem which re-estimates the positions $\boldsymbol{x}^{(t)}$ of vehicles:

$$\boldsymbol{L}^{(t)}\boldsymbol{x}^{(t)} = \boldsymbol{\delta}^{(t)},$$

where Laplacian matrix $\boldsymbol{L}^{(t)}$ indicates the connectivity links among the vehicles, and vector $\boldsymbol{\delta}^{(t)}$ contains the differential coordinates along with GPS positions of vehicles (also known as anchor points). See more details in [9][10] . This method is named Centralized Laplacian Localization (CLL). However, due to cyberattacks, i.e., spoofing or jamming, aiming to compromise GPS normal operation, the resultant positioning outliers will cause significant degradation in localization accuracy. Therefore, we propose a robust alternative of CLL, aiming to identify which vehicles have been attacked but also to mitigate the effect of spoofing by reducing the GPS bias. In order to achieve that, we impose additional sparsity constraints, assuming that only a subset of cluster's vehicles have been compromised. The new cost function takes the form of:

$$argmin_{\boldsymbol{x}^{(t)},\boldsymbol{o}^{(t)}} \left\| \boldsymbol{L}^{(t)}\boldsymbol{x}^{(t)} - \left(\boldsymbol{\delta}^{(t)} - \boldsymbol{o}^{(t)}\right) \right\|_2^2 + \lambda \left\| \boldsymbol{o}^{(t)} \right\|_1$$

Target vector $x^{(t)}$ *and sparse* $o^{(t)}$ correspond to the positions of vehicles and outliers of GPS caused by spoofing, respectively. Since our goal is to reduce the impact of spoofing, target vector $o^{(t)}$ has to be removed from the anchors part of $\boldsymbol{\delta^{(t)}}$. The sparsity constraint is imposed by the l_1 norm of outliers' vector, while λ controls the trade off between position estimation and sparsity constraints. The cost function can be minimized using CVX software. Furthermore, in order to identify compromised vehicles, we follow a 3-steps approach: 1) Measure the Euclidean distance between GPS and re-estimated positions, 2) Distances exceeding a threshold equal to $10m$ are set to zero, 3) Apply 2-means clustering on the group of distances, creating two groups of distances which correspond to spoofing and non-spoofing. In that way, the proposed collaborative defense mechanism mitigates and detects potential cyberattacks against GPS spoofing. The deployed module is named Robust CLL (R-CLL).

4.2.2.2 Demonstration Setup

4.2.2.3 Scenario-1: Moving Object Detection

Moving object detection is the task of identifying the physical movement of an object in a given region or area. Over last few years, moving object detection has received much of attraction due to its wide range of applications like video surveillance, human motion analysis, robot navigation, event detection, anomaly detection, video conferencing, traffic analysis and security. In addition, moving object detection is very consequential and efficacious research topic in field of computer vision and video processing since it forms a critical step for many complex processes like video object classification and video tracking activity. Consequently, identification of actual shape of moving object from a given sequence of video frames becomes pertinent. However, task of detecting actual shape of object in motion becomes tricky due to various challenges like dynamic scene changes, illumination variations, presence of shadow, camouflage, and bootstrapping problems. Throughout the demo for CPSOSAWARE, we have performed the experiment as illustrated in Figure 4. An object is moving in the area behind the ego vehicle various trajectory shapes, speeds and postures and the object detection perception module is detecting the object while the camera sensor is attacked in the way described in Table 17.

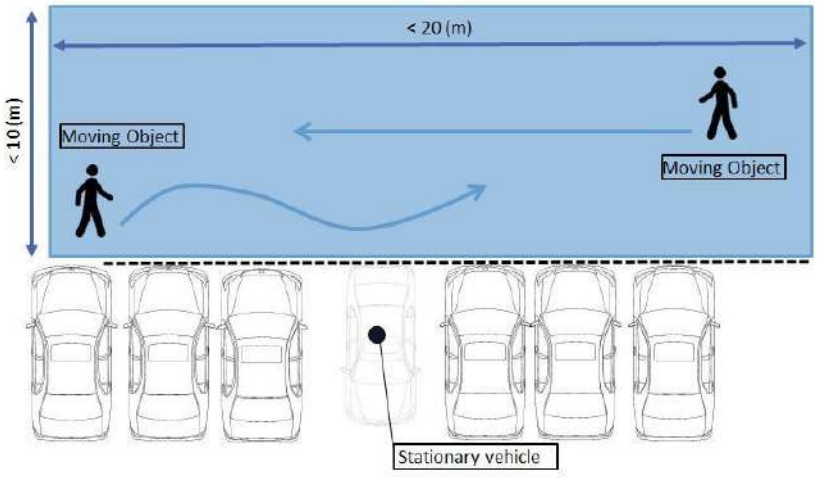


Figure 106: Schematic representation of the parameters of the experiment testing CPSOSAWARE's cyber-attack detection and mitigation engine in the moving object detection use case.

Figure 107, Figure 108 and Figure 109 illustrate instances of the experiments as well as the output of the moving object detection perception modules.

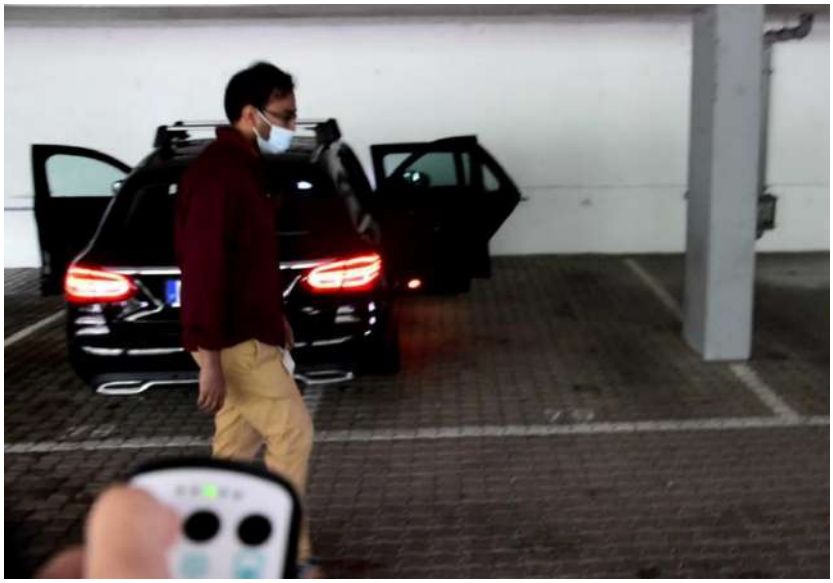
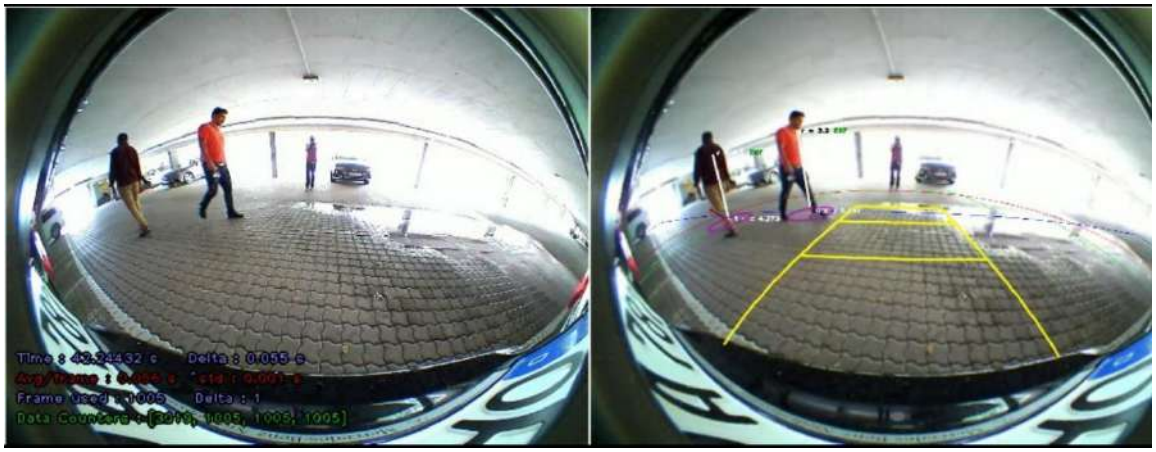


Figure 107: Malicious user attacks to the Moving Object Detection perception component. While the autonomous vehicle, engaged for CPSOSAWARE, is static pedestrians are moving

Figure 108 and Figure 109 illustrate the output of the moving object detection module at the presence

(a) **(b)**

Figure 108: Output of the moving object detector perception engine, at the absence of cyberattack. (a) camera output at the viewing layer, (b) Output of the perception layer: moving pedestrians crossing the vehicle trajectory. The bounding box highlights the object's boundaries. The direction, speed and time to collision is disclosed.



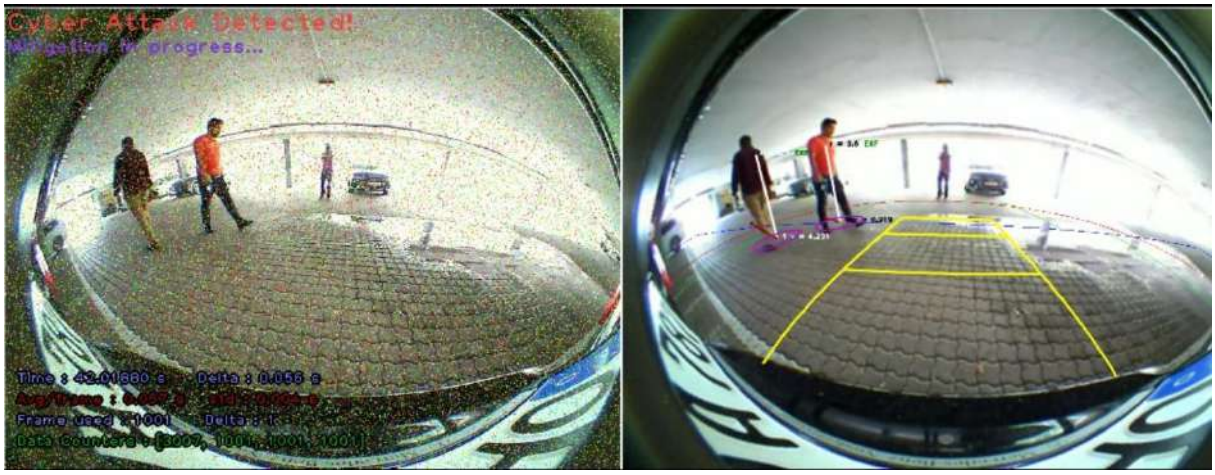
(a) **(b)**

Figure 109: **Output of the moving object detector while the vehicle is not under cyberattack. (a) viewing layer, (b) Output of the perception layer: moving pedestrians out of vehicle trajectory. The direction, speed and time to collision is disclosed.**

As part of visual inspecting the efficiency of CPSOSAWARE's cyber-attack detection and mitigation engine,

Figure 110(a)-(d), illustrate the output of moving object detection when the vehicle is submitted to cyber-attack. **Figure 110** (a), (c) present the output of camera view, when the vehicle is cyber attacked, while **Figure 110**(b), (c) display the output of the moving object detection module after the CPSOSAWARE's cyber-attack detection and mitigation engine. As seen, the localization of moving object as well as the speed and direction of moving objects are still detected.

(a) **(b)**



(c) (d)

Figure 110: (a). (c) Camera sensor cyber attacked, the cyber attacked Is detected and the mitigation engine is under progress. (b), (d) output of the moving object detection module after the cyber attack mitigation engine has been performed.

Subsequently, 4.2.2.4 summarizes the demo output of investigating the efficiency of CPSOSAWARE's cyber-attack detection and mitigation engine in self localization operation.

4.2.2.4 Scenario-2: Self Localization

As above, the scenario involves an attacker holding a remote control through which he activates some malicious software, already installed in the car (**Figure 111**).

Figure 111: The autonomous vehicle engaged for CPSOSAWARE is cyber attacked by an external agent, through a remote control.

In order to assess the potential and the limitations of the CPSOSAWARE's detection and mitigation engine employed to robustify the autonomous navigation function, CPSOSAWARE's test vehicle, as part of the self-

localization demo, the vehicle moves on a closed loop, on uneven ground at velocities ranging from 0-15 km/h. While the car moves, the camera sensor is attacked, the location as well as the trajectory of the vehicle is estimated through the camera-based odometry. The accurate mitigation of the attack is assessed through measuring the divergence between the start and end point of the trajectory. In case of successful mitigation, the end and start point should coincide.

In the figures presented below, **Figure 112** illustrates an instance of the camera based cyber-attack as it is displayed in the viewing layer (top left), the 3D modelling of the scene after the mitigation of cyber-attack is displayed on bottom left image. The trajectory estimation along with the occupancy grid map is presented on the right image. Moreover, **Figure 113** illustrates the trajectory estimation (output of self-localization module). The green trajectory is derived by the visual odometry (camera sensor), while the vehicle is under attack. The red trajectory compromises the output of the localization module based on the flex ray sensor. As it can be observed, according to the camera-based localization solution, the vehicle moved on a closed trajectory, which corresponds to the shape of the trajectory performed by the vehicle. However, the red trajectory (estimated by the flex ray sensor) illustrates a considerable divergence. Thus, it can be observed that despite the camera being attacked, the camera-based localization solution performs better than the flex-ray. The inaccuracy of the flex ray odometry is basically due to insensitivity of the mechanical sensors at low speed (<8kph) and steep yaw turning.

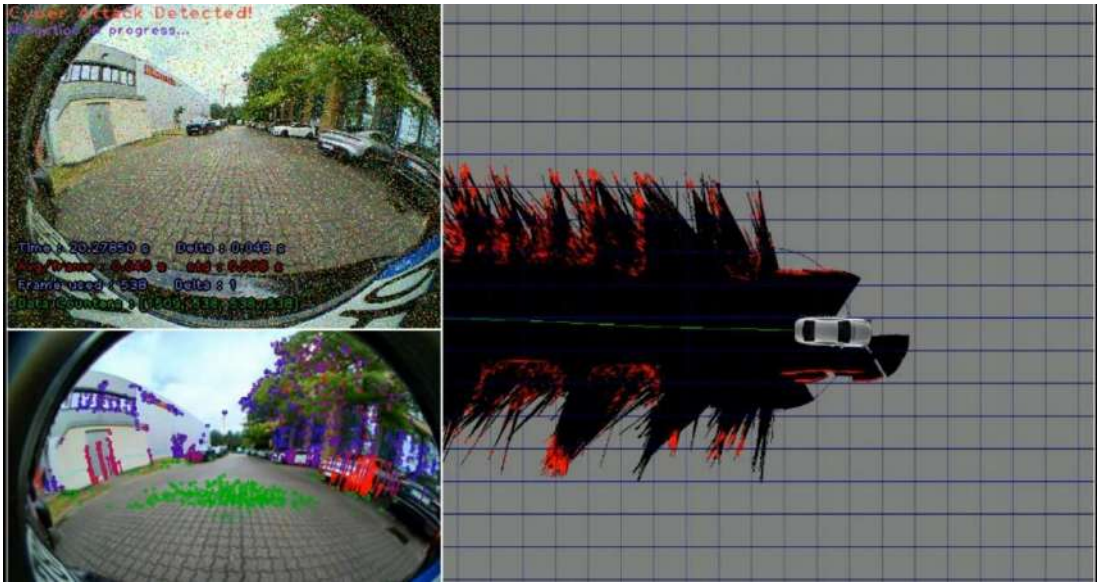


Figure 112: Camera based cyber-attack illustrated in the viewing layer (top left), bottom left: 3D modelling of the scene after the mitigation of cyber-attack. Right image: trajectory estimation and occupancy grid map.

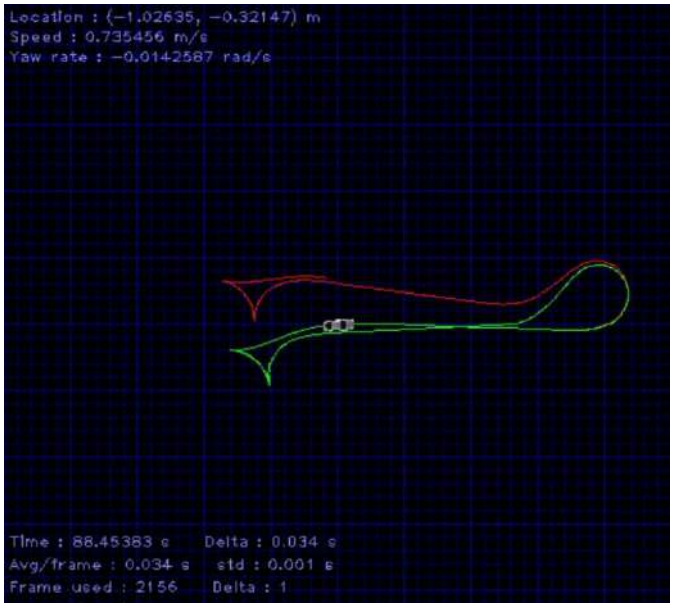


Figure 113: Trajectory estimation as output of self-localization function. The green trajectory is the visual odometry (camera sensor) output, while the camera is cyber-attacked. The red trajectory is the output of flex ray odometry.

4.2.2.5 Scenario-3: Self Localization

As discussed in CPSOSAWARE Deliverables D6.2 and D6.4, It is of interest to understand the impact of noise on the performance of autonomous vehicle in its autonomous navigation. Concretely, we would like to understand how much degradation of the Occupancy Grid Map (OGM) occurs (Figure 116) when the camera is cyber attacked as well as how much the front of the obstacles is distorted (Figure 13). For obstacle avoidance in navigation, it is practically efficient and much more meaningful to compare the outcome of obstacle polygons extracted between OGMs. This is because the polygons are fundamentally used for calculating time-to-collision and planning the local path to avoid possible collision. On the other hand, for parking applications, it is much more meaningful to measure the free-space and obstacle's boundaries within surrounding environment to examine whether a possible parking-slot is detected, and an optimal trajectory can be planned accordingly. To make this clear, let's review an example in Figure **114**. In both figures, the point cloud created by a camera-based 3D reconstruction solution is fed into the OGM. Obstacle polygons Figure **115** are boundaries of high objects (>15 cm height), which are extracted and marked in red by a pre-determined solution.

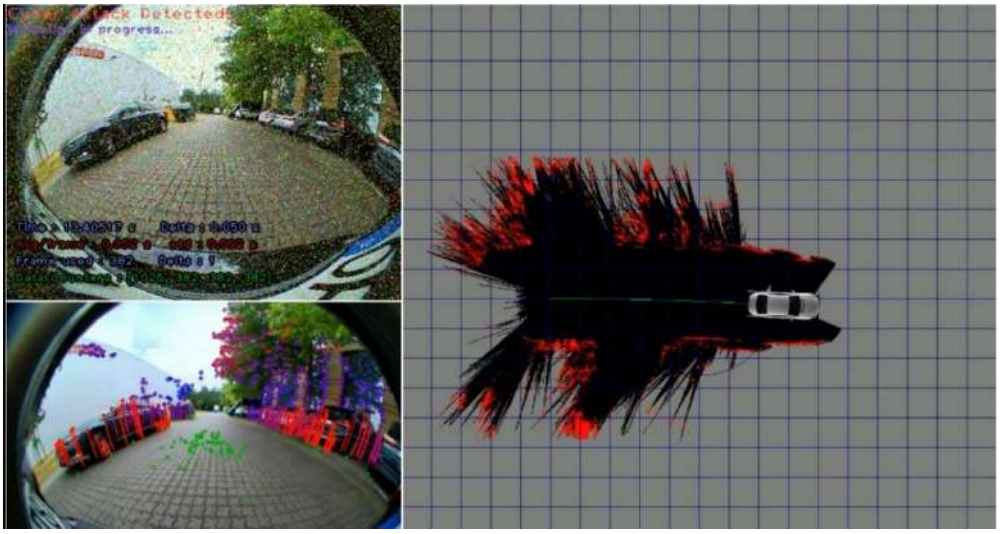


Figure 114: Output of the CPSOSAWARE's cyber-attack immunization engine at the OGM level. Top Left: output of the cyber-attack at the viewing layer. Bottom Left: 3D Environmental sensing layer output. Right Image: Occupancy Grid Map built during the car moving as displayed in the left image.

Figure 115: Output of boundary estimation while the camera is cyber attacked. Top left: Output of 3D environmental sensing. Bottom left: Cyber attack at the camera sensor. Right image: Polygons enclosing the point clouds clustered as discrete objects.

As it can be assessed by the Figure 114 and Figure 115, the efficacy of CPSOSAWARE's cyber-attack immunization engine can provide robust scene understanding under adverse illumination conditions both outdoors Figure 114 and indoors Figure 115 as well as at diverse weathering conditions, like raining Figure 114.

As the demonstration developed into exploring the robustness of CPSOSAWARE's cyber-attack immunization engine, at first the critical modules contributing to Automated Emergency Braking (e.g: Moving Object Detection, Self-Localization) and obstacle avoidance (self-localization, Occupancy Grid Mapping) were assessed and at a next step, the robustness of CPSOSAWARE's solution in addressing cyber-

attacks to the Fully Automated Parking is also investigated. The investigation of this function was out of the scope of the Grant Agreement. However, as the derived results regarding lower-level functions outperformed consortium's expectations, it was decided to also investigate the potential and limitations of CPSOSAWARE's solution at this higher level of functionality

4.2.2.6 Scenario-4: Occupancy Grid Mapping/ Object Boundary Distortion

As discussed in CPSOSAWARE Deliverables D6.2 and D6.4, It is of interest to understand the impact of noise on the performance of autonomous vehicle in its autonomous navigation. Concretely, we would like to understand how much degradation of the Occupancy Grid Map (OGM) occurs (**Figure 114**) when the camera is cyber attacked as well as how much the front of the obstacles is distorted (**Figure 115**). For obstacle avoidance in navigation, it is practically efficient and much more meaningful to compare the outcome of obstacle polygons extracted between OGMs. This is because the polygons are fundamentally used for calculating time-to-collision and planning the local path to avoid possible collision. On the other hand, for parking applications, it is much more meaningful to measure the free-space and obstacle's boundaries within surrounding environment to examine whether a possible parking-slot is detected, and an optimal trajectory can be planned accordingly. To make this clear, let's review an example in Figure 116. In both figures, the point cloud created by a camera-based 3D reconstruction solution is fed into the OGM. Obstacle polygons Figure 117 are boundaries of high objects (>15 cm height), which are extracted and marked in red by a pre-determined solution.

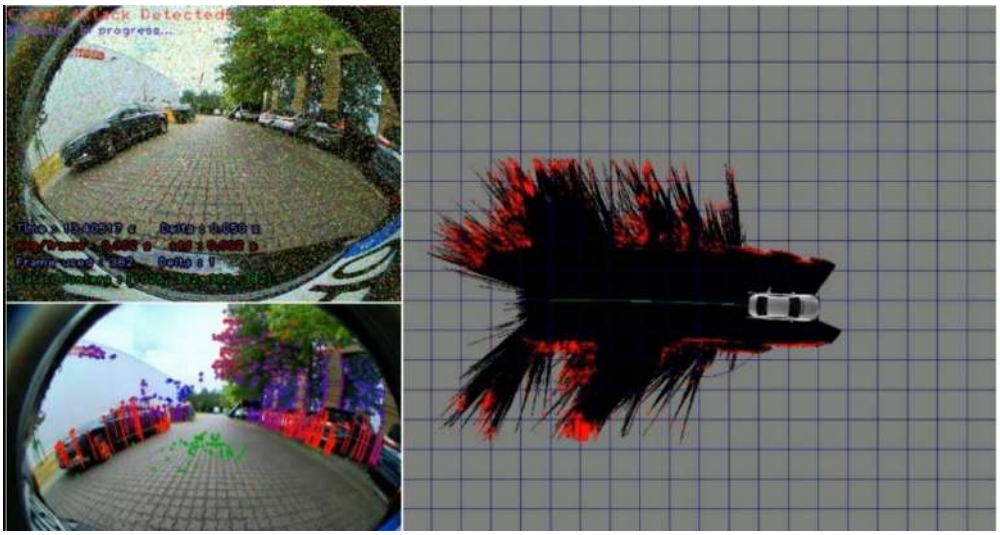


Figure 116: Output of the CPSOSAWARE's cyber-attack immunization engine at the OGM level. Top Left: output of the cyber-attack at the viewing layer. Bottom Left: 3D Environmental sensing layer output. Right Image: Occupancy Grid Map built during the car moving as displayed in the left image.

Figure 117: Output of boundary estimation while the camera is cyber attacked. Top left: Output of 3D environmental sensing. Bottom left: Cyber-attack at the camera sensor. Right image: Polygons enclosing the point clouds clustered as discrete objects.

As it can be assessed by the Figure 116 and Figure 117, the efficacy of CPSOSAWARE's cyber-attack immunization engine can provide robust scene understanding under adverse illumination conditions both outdoors Figure 116 and indoors Figure 117 as well as at diverse weathering conditions, like raining Figure 116.

As the demonstration developed into exploring the robustness of CPSOSAWARE's cyber-attack immunization engine, at first the critical modules contributing to Automated Emergency Braking (e.g: Moving Object Detection, Self-Localization) and obstacle avoidance (self-localization, Occupancy Grid Mapping) were assessed and at a next step, the robustness of CPSOSAWARE's solution in addressing cyber-attacks to the Fully Automated Parking is also investigated. The investigation of this function was out of the scope of the Grant Agreement. However, as the derived results regarding lower-level functions outperformed consortium's expectations, it was decided to also investigate the potential and limitations of CPSOSAWARE's solution at this higher level of functionality.

4.3 Cooperative awareness scenario

The concept of cooperation has received considerable interest in the Connected and Automated Vehicles (CAVs) community during the last decade. The basic idea behind such concept was to suppress the effects (on the perception) of occlusion created by additional traffic agents in the scene. More recently, the research efforts were geared towards a unifying approach of introducing a new viewpoint involving more processing. High-quality localization, the potential of traffic agents to determine their poses in a sound and accurate way (i.e., positions and orientations), related to the surrounding objects or to geographic standards for judging or deciding, is important. The GNSS, as a well-known modality, is not always accessible or trustworthy, due to a multitude of reasons related to signal blockages, multipath thinking, and compressing [1]. One potential approach for localization in GNSS-denied areas is to investigate the possibility of utilising map matching methods, supported that there is an environmental model derived from a prior map illustrated as a scalar field. Scalar fields correlate a scalar value accompanying every point in the area. Possible application might involve gravity deformations [2] [2], magnetic disturbances irregularity [3][24], topographic [26], and olfaction [27], to name any. Methods utilising scalar fields for localization, agents' regularization, dead-reckoning issues, growth through correlating measurements derived by on-board sensors with the earlier prior field maps involving terrain-aid traveling [28] and magnetic disturbance–based traveling [24].

CPSoSAWARE has engaged a multitude of modules for environmental modelling, self-localization, vehicle control, Driver System Monitoring, and cyber-attack immunization. All the modules above are involved in Level-3 and level-4 autonomous. Such modules synergy was also considered in CPSoSAWARE. Figure 118 illustrates all the modules developed in the automotive pillar of CPSoSAWARE as well as the synergies and the associations.

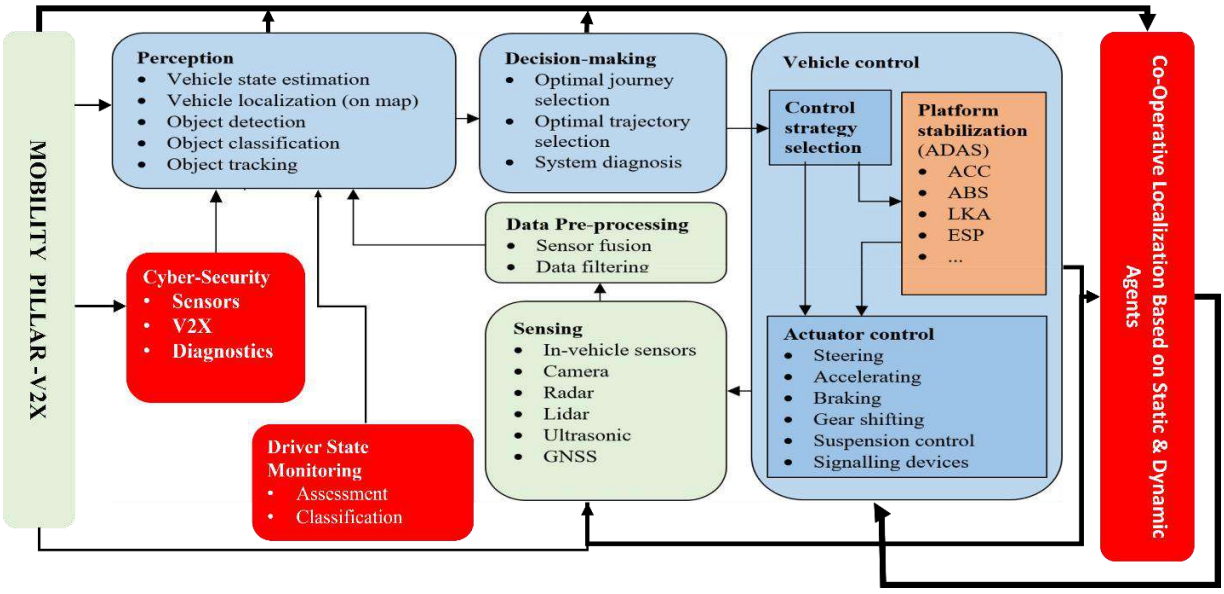


Figure 118: CPSoSAWARE Automotive Pillar Components

Through the demonstration of the Co-Operative Localization use-case, the solutions have been tested in both the simulator environment (section 4.3.1) and the real vehicle (section 4.3.3).The full pipeline of the

architecture developed for co-operative awareness in CPSoSAWARE, along with the contribution of the consortium partners is depicted in Figure 119

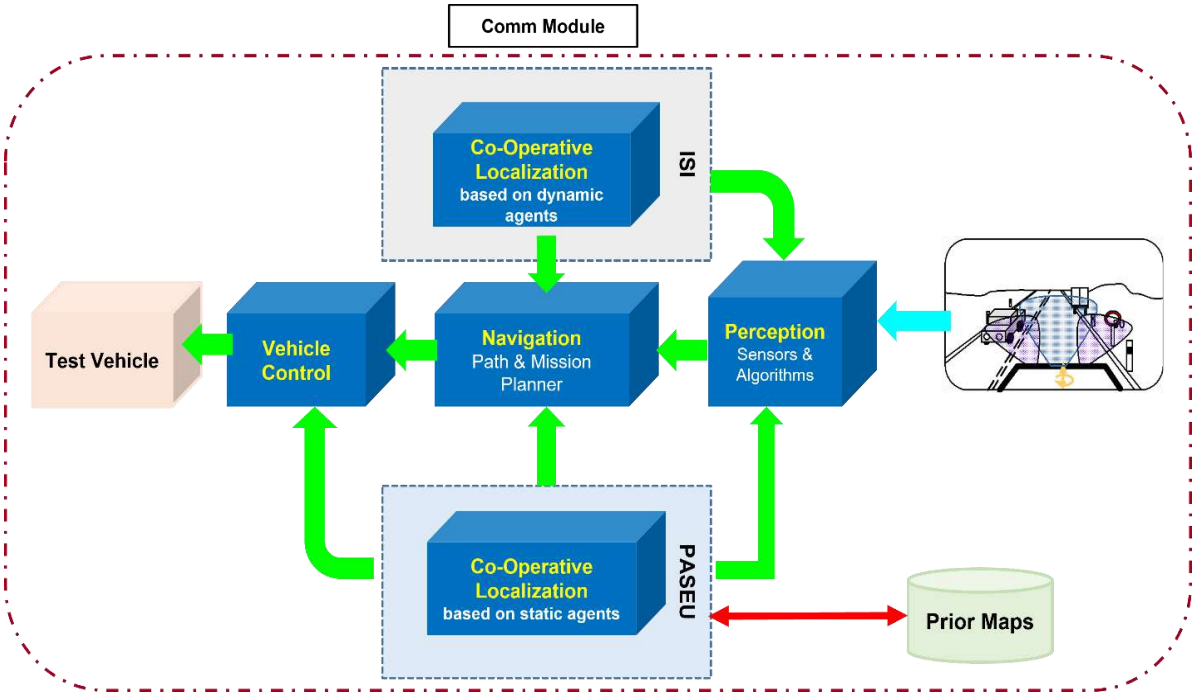


Figure 119: Architectural pipeline of the co-operative awareness solution. The schematic illustrates the complementarity of the solutions demonstrated in the simulator and the real vehicle.

Through the co-operative awareness solution, developed in CPSoSAWARE, the definition of the topology and the recognition of the actions of traffic agents is supported through the complementary solutions developed by ISI and PASEU as depicted in the figure above, which are analysed in the sections 4.3.1 and 4.3.3 respectively.

4.3.1 Simulation Based Evaluation

Carla is an open-source autonomous vehicle platform developed by the Urban Computing Foundation and is designed to be used as a testbed for developing and evaluating autonomous vehicle technologies. Carla provides a realistic simulation environment for testing and evaluating autonomous vehicle algorithms, including perception, localization, planning, and control. It includes a physically and visually realistic virtual city environment, as well as a variety of sensor models and vehicle dynamics models. The Carla platform includes several tools and libraries for working with autonomous vehicles, including a server component that runs the simulation and provides an interface for connecting to the simulation from external clients and a Python client library for connecting to the simulation server and interacting with the simulation.

We take advantage of the fact that Carla can be used to test and evaluate a wide range of autonomous vehicle technologies, including perception algorithms and localization algorithms. It is particularly well-suited for testing and evaluating algorithms that rely on sensor data, such as lidar and camera data, as it provides a realistic and flexible simulation environment for generating and processing this data. Combined

with ROS framework it constitutes a great end-to-end modular platform for testing our solutions in realistic scenarios. The component that links Carla with ROS is the Carla-ROS bridge which facilitates the bid-directional communication between the two parts.

In the context of the implementation of the following scenarios we have developed applications in the form of ROS nodes which can be easily plugged into the framework. These ROS nodes usually consume data generated by Carla (i.e. images, lidar) and produce results. ROS nodes fall into two categories:

- Application nodes which implement in Python an algorithmic solution (i.e. monocular visual odometer etc.)
- Auxiliary nodes which are used for other non-algorithmic purposes like visualization or logging.

A view of the utilized framework is illustrated in Figure 120.

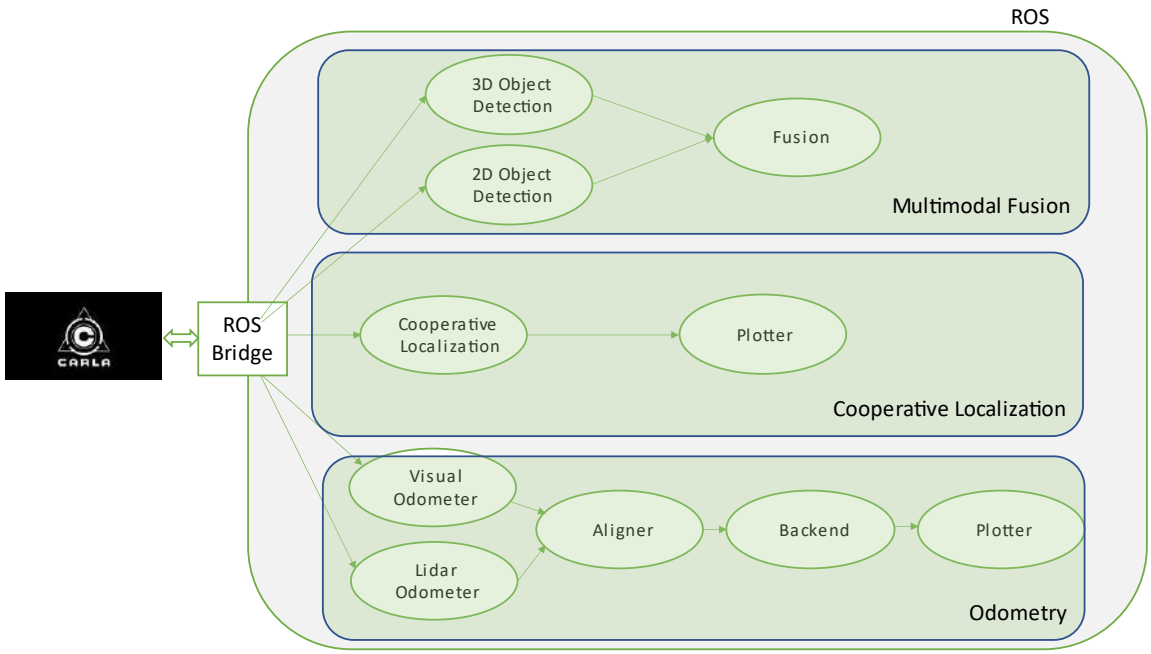


Figure 120 Carla ROS framework

The ROS nodes were hosted in three different physical machines, two desktop PCs and one NVIDIA Jetson TX2 device. The configuration of each machine is described in the table below.

	PC1	PC2	Jetson TX2
OS	Ubuntu 20.04	Ubuntu 20.04	Linux4Tegra (based on Ubuntu 18.04)
CPU	i7 8700k	i7 8700k	Arm Cortex-A57 (quad-core) @ 2GHz + NVIDIA Denver2 (dual-core) @ 2GHz

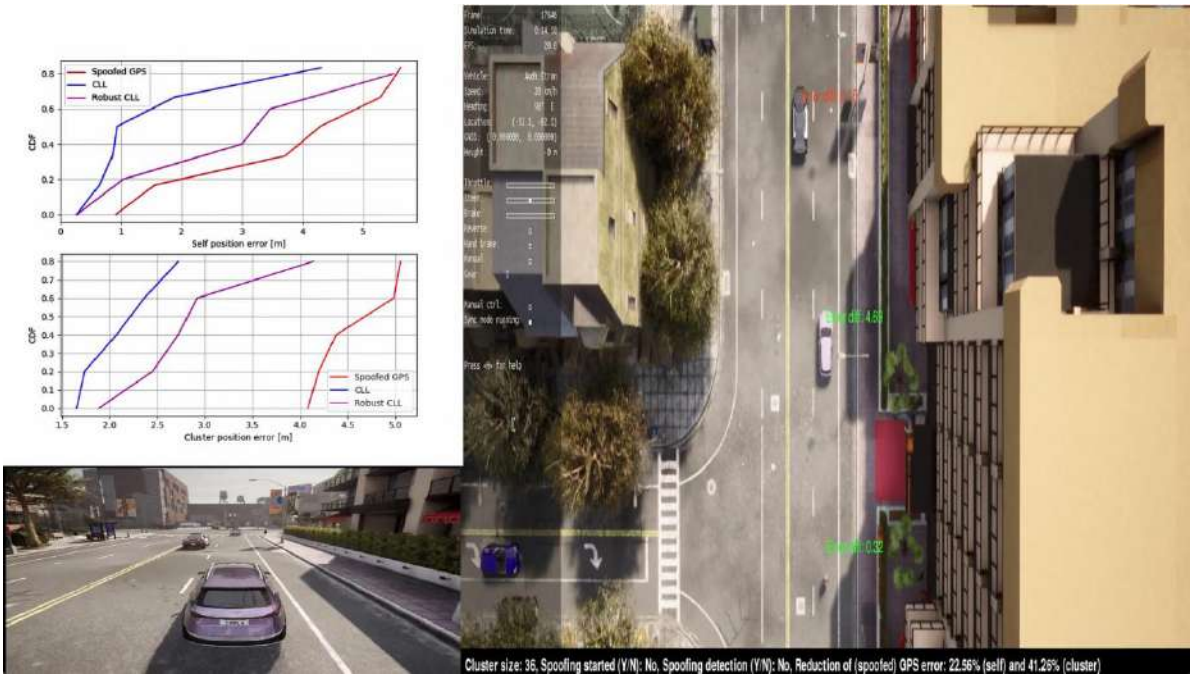
GPU	rtx2080ti	Rtx3060	256-core Pascal @ 1300MHz
RAM	32GB DDR4	32GB DDR4	8GB LPDDR
Main Storage	1T SSD	512G SSD	32GB eMMC 5.1

PC1 was also the host of Carla and the ROS core, which is a pre-requisite for any ROS environment.

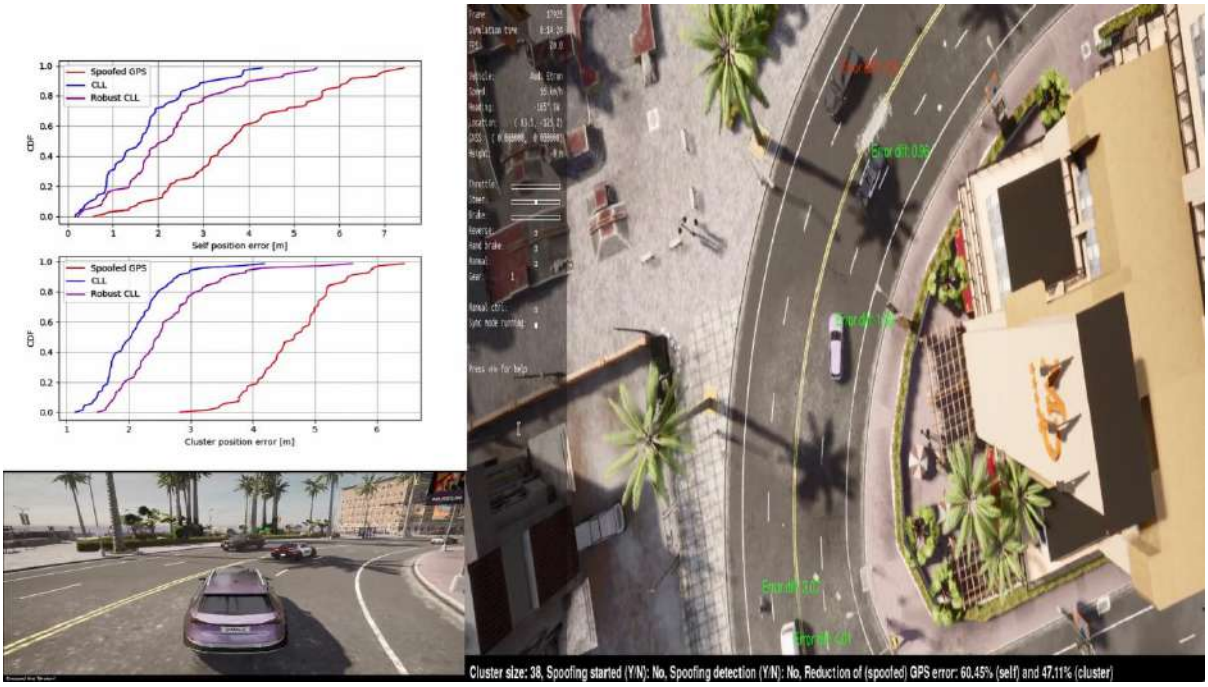
Concerning the network setup all the ROS nodes were deployed in the Local Area Network in which also the Carla simulator runs. Even though Carla provides an API that can be called remotely, we did not follow this approach due to latency errors and since it was out of scope for the simulated experiments.

4.3.1.1 Robustified cooperative awareness for mitigation and detection of cyberattacks

The deployed module was integrated and deployed in aforementioned Carla-ROS framework. Below we demonstrate examples inside Carla simulator before and the after spoofing initiation. From Figure 121, we see ego vehicle (bottom left), birds eye view (right) and cumulative distribution functions (CDF) of ego vehicle and cluster position error (top left). A text box appears in the bird's eye view window indicating the size of cluster, if spoofing has started, detection of spoofing and reduction of GPS error using R-CLL. Green texts appear also above vehicles in bird's eye view, highlight the benefits in meters in terms of position error using R-CLL and with respect to GPS. Since spoofing has not started yet, we see from CDFs of Figure 121 that baseline CLL achieves better positioning results than its robust alternative. Furthermore, detection alarm correctly indicates no spoofing. In any case, the reduction of standalone and cluster's GPS error using R-CLL reaches 22.56% and 41.26% in



(a)

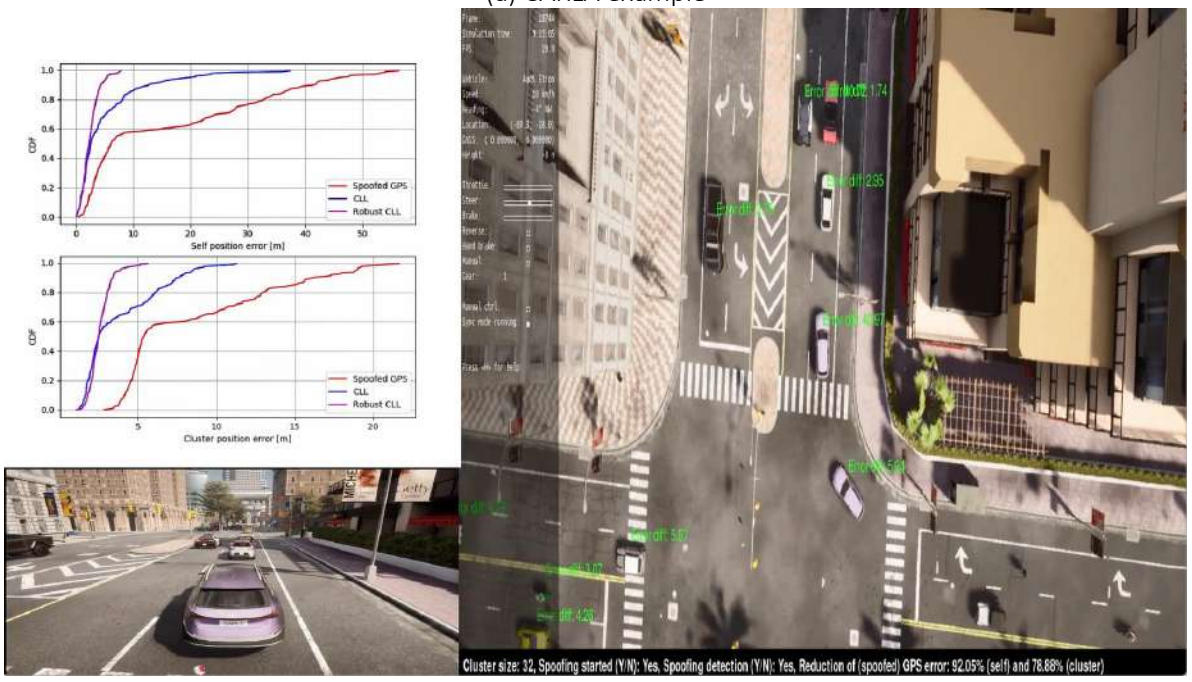


(b)

Figure 121: Realizations of collaborative defense mechanism inside CARLA simulator before spoofing

In Figure 122, spoofing has started targeting ego vehicle and 20% of the vehicles of the cluster. Therefore, both standalone and cluster's GPS error are significantly affected by the positioning outliers. For example, ego vehicle's GPS error exceeds 50m, and cluster's 17.5m. Detection alarm in both subfigures has been correctly raised, meaning that R-CLL successfully identified that ego vehicle has been compromised. Furthermore, from CDFs we conclude that baseline CLL is totally unable to mitigate the impact of spoofing, since although reduces (spoofed) GPS error, it does achieve maximum localization error slightly lower than 40m (for the case of ego vehicle). On the contrary, the proposed R-CLL module addressed spoofing very efficiently since it almost "cleared" GPS from spoofing biases. More specifically, R-CLL reduced GPS errors in two subfigures by 84.11% and 62.56%, and 92.05% and 78.88%, respectively.

(a) CARLA example



(b) CARLA example

Figure 122: Realizations of collaborative defense mechanism inside CARLA simulator after spoofing

4.3.2 Solutions Pipeline

4.3.2.1 AI Accelerated multimodal scene analysis and understanding

This scenario presents the application of Model Compression and Acceleration (MCA) techniques on high-performance Deep Neural Network (DNN) models used for scene understanding and understanding in automotive applications. To this end, we focus on two state-of-the-art weight sharing techniques, i.e., Vector Quantization (VQ) and Dictionary Learning (DL), and two modern DNN-based detectors, namely 2D (i.e., image based) SqueezeDet, and 3D (i.e., point-cloud based) Point-Pillars. Additionally, to make full use of the available information, that is, of the sensory inputs from the visual camera and LiDAR, we also present a simple late-fusion strategy that combines the outcome of the unimodal 2D and 3D detectors, for an enhanced detection performance. We apply weight sharing to both unimodal architectures and compare the performance of the fused output compared to each approach individually. The performance was measured in a complete dataset extracted by CARLA simulator.

A simple late fusion strategy is proposed for processing the detection outcomes of the two considered modalities, namely, 2D visual images and 3D point clouds. An illustration of the strategy is presented in Figure 123. The concept behind the fusion approach is to select the outcome of the detector with the highest confidence score, i.e., either the 2D detection based on SqueezeDet (lower branch in Figure 123) or the 3D detection based on Point-Pillars (upper branch in Figure 123). To this end, the well-known Non-Maximal Suppression (NMS) algorithm [11] is employed to process the detection outcomes of the two branches. Note that for the 3D detection branch, initially, the 3D bounding boxes are projected on the 2D image, and these projections are assigned the confidence score of the 3D detector. Thus, NMS receives a 2D image that contains bounding boxes from both modalities before fusing them.

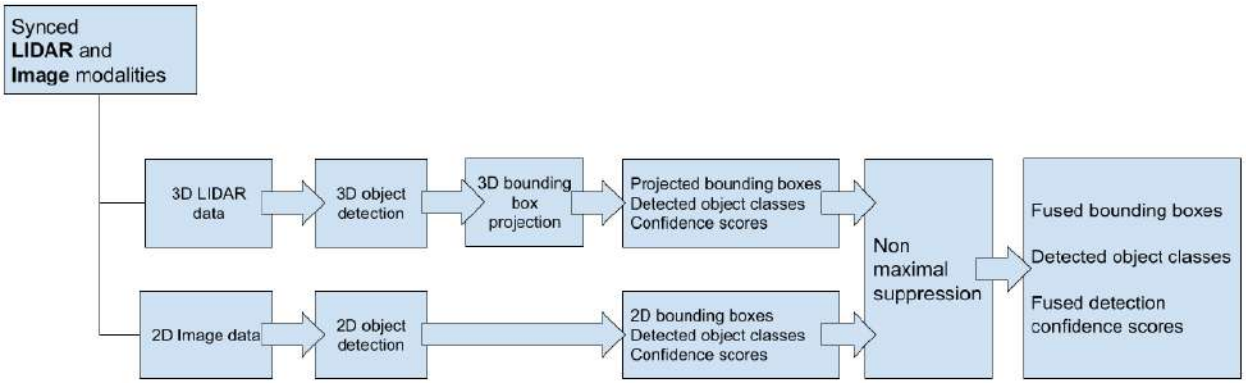


Figure 123: Architecture of late multi-modal fusion approach

See more details in Deliverable D3.1 and research papers [12][13].

4.3.2.2 Robustified cooperative awareness for cooperative localization

This module was developed in order to empower vehicles with distributed 4D situational awareness in the case of normal GPS operation. Apart from the approaches described in Deliverable D3.3, two more types of solutions have been integrated to the CARLA-ROS framework, i.e., local information diffusion [14] and collaborative odometry estimation [15]. We provide also a dataset [16] extracted from CARLA, containing driving parameters (ground truth position, velocity, acceleration, etc.) from many simulated vehicles in order to facilitate the evaluation of deployed cooperative localization solutions.

4.3.2.3 Local information diffusion

To boost the performance of Cooperative Awareness we propose a local diffusion-based approach, which addresses the limitations of global awareness solution (see D3.3) related to lacking knowledge about other networked vehicles and convergence speed. The proposed framework is linear with respect to its parameter, totally distributed, and doesn't depend on fixed communication topologies. The main properties of the deployed module could be summarized as follows: a) A local diffusion-based awareness scheme is proposed, in which every vehicle estimates its own target location vector. Least-Mean-Squares (LMS) algorithm has been used for the distributed solution of target cost function, b) Local graph Laplacian matrix was employed for the derivation of a linear measurement model comprised of multimodal measurement sources, c) The solution doesn't depend on fixed vehicular topologies, but rather addresses dynamic and sparse topologies, without all-to-all V2V communication, d) Realistic simulations carried out in CARLA simulator, verify the proposed analysis as well as its benefits.

4.3.2.4 Collaborative odometry estimation

The novelty of this module lies on the fact that the ego vehicle performs cooperative odometry estimation, i.e., multiple 3D pose estimation in absolute coordinates and over time horizon, so as to effectively be aware in a distributed manner of surrounding environment and not only of its own state. Our solution lends itself from the cooperative gradient descent algorithm [17] which is extended by multimodal measurement models and unknown target parameters. The main properties of this approach can be summarized as follows: a) A cost function using maximum likelihood criterion is formulated and minimized by the ego vehicle and comprised of four multimodal measurement models and the unknown 3 degrees of freedom (DOF) parameters of itself and closest neighbors, b) We derive a novel alternating gradient descent algorithm for effectively performing cooperative odometry, c) We verify that by treating yaw angles as parameters to be optimized, much more accurate localization can be achieved in the case of both self-position and awareness, d) The extensive evaluation study performed in the realistic traffic conditions of CARLA simulator has shown very promising results in terms of the accuracy of self and neighboring 3 DOF estimation.

4.3.2.5 Multimodal odometer solutions for autonomous vehicles

We build upon the deployed SLAM solutions described in deliverable D3.1 and design a more sophisticated multimodal fusion scheme based on Graph Laplacian Processing (see deliverable D3.3) in order to increase the accuracy of pose estimation. Its core idea is to couple the extracted pose from Visual (LIDAR based) approach with the landmarks detected by the LIDAR based (Visual) solution. These steps are performed using the Graph Laplacian Processing framework. The intuition behind our motive was that by effectively

combining poses and landmarks from different modalities, which refer to the same task, a more accurate pose of the vehicle could be produced. Therefore, as a first step, Graph Laplacian is used to estimate the new pose separately for the two groups of data. Afterwards, using the two estimated poses in the measurement models, we run two linear Kalman Filters (KFs) assuming the constant velocity motion model. The two KFs produce also the covariance matrices related to their (Gaussian) estimations. In order to fuse the two estimated poses in an optimal manner and avoid any handcrafted weights, we deduce to the technique described in [18]. Assume two groups of Gaussians means and covariances: (x_1, Σ_1), (x_2, Σ_2). Compute initially matrices: $P_1 = \Sigma_1^{-1}(\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$and $P_2 = \Sigma_2^{-1}(\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$. The optimal mean and covariance will be equal to: $x_{opt} = P_1 x_1 + P_2 x_2$ and $\Sigma_{opt} = P_1 \Sigma_1 P_1^T + P_2 \Sigma_2 P_2^T$. The same approach is followed in our framework using the estimations of the two KFs. To further increase the performance of the proposed solution, the measurement model at each KF is reformulated accordingly: Given the output pose of Graph Laplacian $\boldsymbol{x_{1,lapl}^t}$ at time instant t and the fused pose $\boldsymbol{x_{fused}^{t-1}}$ at $t-1$, the measurement vector of the first KF will be equal to: $\boldsymbol{z_1^t = 0.95 * \left(x_{fused}^{t-1} + v^t * dt\right) + 0.05 * x_{1,lapl}^t}$, where $\boldsymbol{v^t}$ is the velocity and $\boldsymbol{dt}$ the time interval, while $\boldsymbol{x_{fused}^{t-1} + v^t * dt}$ is actually a motion prediction of vehicle's position at the next time instant t. The same are followed for the second branch of our approach. Therefore, by infusing the previously estimated pose in the KFs, the solution can be further improved. The proposed system architecture is shown in Figure 124.

Additionally, we provide a dataset [19] et of testing different odometer solutions in diverse weather and traffic conditions using CARLA-ROS framework.

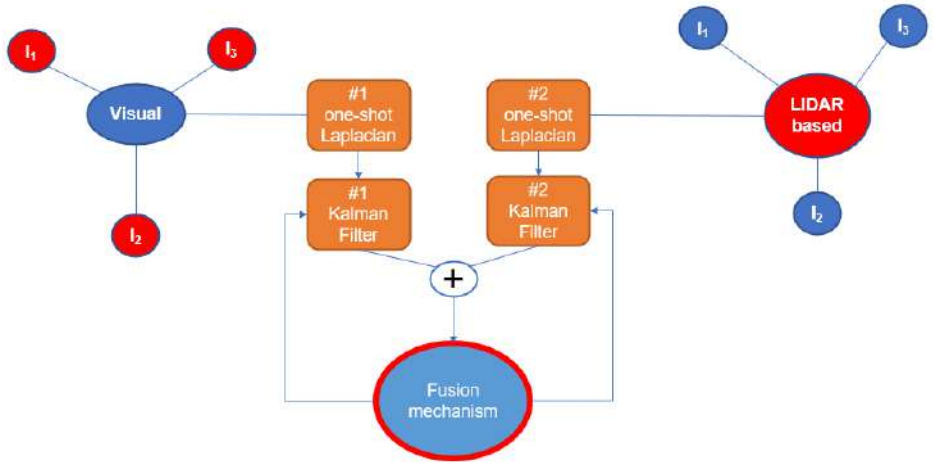


Figure 124: Proposed multimodal fusion of odometers approach

4.3.2.6 Simulation Based Evaluation

The same end-to-end testing framework, that is described in [7], was deployed for the evaluation of the following scenarios.

4.3.2.6.1 AI-Accelerated multimodal scene analysis and Understanding

A performance evaluation of the employed acceleration/ compression techniques, using state-of-the-art convolutional DNNs, is presented in this section, along with the effect of multi-modal fusion on the DNNs before and after the application of MCA. More specifically, in Experiment I, we evaluate the representation power of the various VQ and DL approximation schemes, by measuring the quantization error incurred by the techniques. This experiment helps us gain insight into the employed techniques and set optimal values for the required parameters. In Experiment II, the focus is on the application of multi-modal fusion of 2D-based and 3Dbased data for object detection in an automotive setting. It is shown that multi-modal fusion has a positive impact on the performance of object detection not only before but also after the application of MCA techniques.

4.3.2.6.2 Experiment-I: Measuring the Quantization Error

Here, we focus our attention on the comparative performance of the employed weight-sharing techniques, the relation between the achieved acceleration and compression ratios, as well as the role of subspace grouping. Experiment I is based on individual, pre-trained layers from the widely used image classification CNNs ResNet50 [20] and SqueezeNet [21]. Note that the latter also constitutes the backbone network for the 2D object detector studied in Experiment II. After experimentation, the parameter values that yielded the best results were as follows: subspace dimension $N' = 8$, $c = 3$ (i.e., the DL codebook was 3 times larger than the VQ one), sparsity level (for DL) $\rho = 2$. Finally, to enable a direct comparison between the VQ and DL results, the two rivals yielded the same acceleration ratio.

In the first part of Experiment I, we present a performance evaluation concerning the two variants of the sparse coding step of the DL technique (as described in Section III), in comparison to the performance obtained by VQ, for a range of acceleration ratios. For this evaluation, we measure the mean squared error (MSE) between the original and approximate kernels of individual convolutional layers from the employed CNNs. A representative instance of this experiment, involving layers (a) res4f-branch2b of ResNet50 (256 kernels of size 3×3×256), and (b) fire8-expand3x3 of SqueezeNet (256 kernels of size $3 \times 3 \times 64$), is shown in the top row of Figure 125.

As it is apparent in this figure, the DL-based (both variants) techniques outperform their VQ rival leading to significantly lower MSE for the same acceleration, or, equivalently to a significantly higher acceleration ratio, for the same level of incurred error. Regarding the two variants of DL, it can be observed that the added flexibility enabled by the global sparsity constraint leads to (relative) acceleration gains of DL-GS vs DL-LS of up to approximately 10% in the shown examples, depending on the specific configuration and target acceleration.

In the second row of Figure 125, the performance of VQ vs DL, in terms of MSE, is depicted for different values of subspace grouping. As expected, increasing the number of subspaces per group has an impact on performance without changing the relative comparison between VQ and DL. On the other hand, in the third row of Figure 125, the MSE achieved by VQ and DL is depicted versus the achieved compression gain. Again, the advantage of DL vs VQ becomes readily apparent, namely, for the same level of incurred error, the employment of the DL technique results in considerably higher acceleration and compression ratio. Finally, we should notice that, as it is apparent from Figure 125, subspace grouping can be used to better control the achieved compression as a function of the acceleration ratio and the incurred quantization error, thus offering additional flexibility at the system design phase. Specifically, one can achieve higher compression

ratios by increasing the group size, sacrificing either the achieved acceleration (to keep the system performance constant) or the incurred MSE (to keep the acceleration constant).
Finally, we should notice that, as it is apparent from **Figure 125**, subspace grouping can be used to better control the achieved compression as a function of the acceleration ratio and the incurred quantization error, thus offering additional flexibility at the system design phase. Specifically, one can achieve higher compression ratios by increasing the group size, sacrificing either the achieved acceleration (to keep the system performance constant) or the incurred MSE (to keep the acceleration constant).

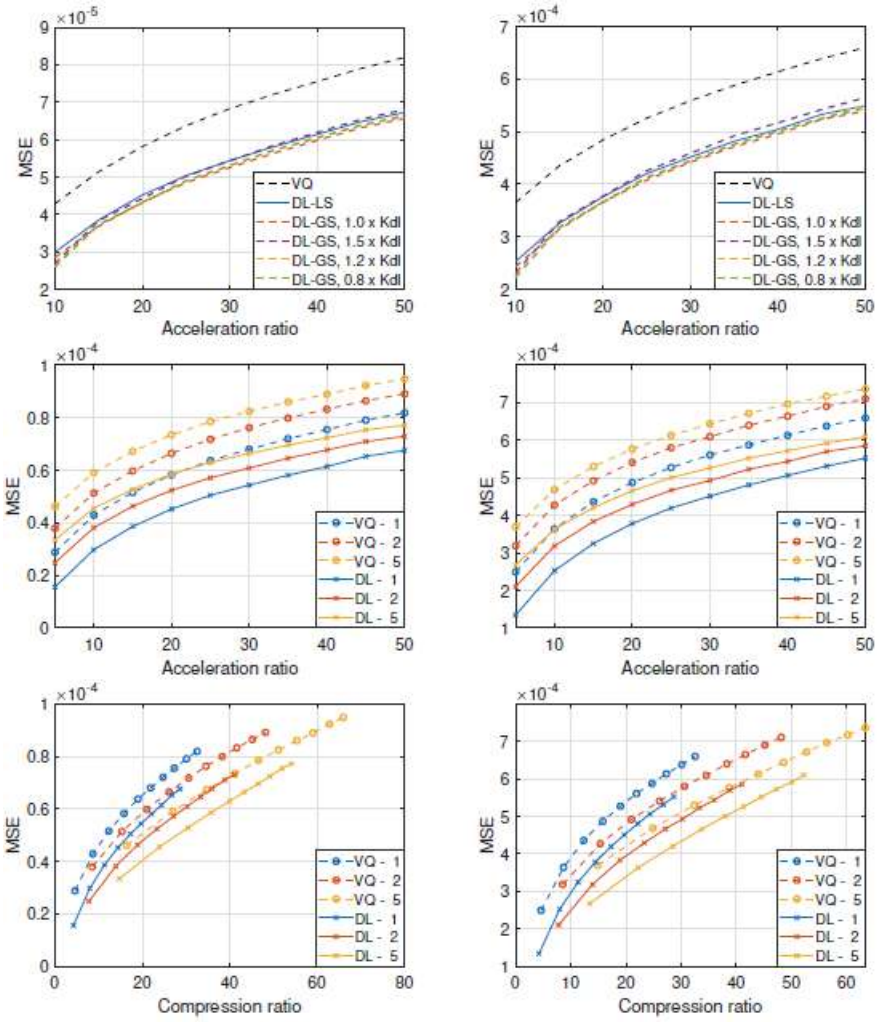


Figure 125: MSE as a function of acceleration and compression of DL vs VQ techniques for layers res4f-branch2b of ResNet50 (left), and fire8-expand3x3 of SqueezeNet (right), using subspace group sizes of 1 (all rows), 2, and 5 (bottom two rows).

4.3.2.6.3 Experiment-II: Application on Multi-modal Fusion driven Object Detection

In this experiment, we evaluate the performance of multimodal fusion when the presented weight-sharing MCA techniques are applied to demonstrate the robustness of the fusion of modalities. The late fusion

method has been implemented via modifying the well-known OpenPCDet suite1. The well-known SqueezeDet and Pointpillars deep models have been used for image based and LIDAR-based object detection, respectively.

SqueezeDet is a fully convolutional detection network, consisting of a feature-extraction part that extracts high dimensional feature maps for the input image, and ConvDet, a convolutional layer to locate objects and predict their class. For the derivation of the final detection, the output is filtered based on a confidence index also extracted by the ConvDet layer. Figure 126 presents the overall architecture of the deep networks, the convolutional volume kernel shapes and the feature tensor shapes. As it can be observed from Figure 126, the feature-extraction (convolutional) part of SqueezeDet is based on SqueezeNet, which is a fully convolutional neural network that employs a special architecture that drastically reduces its size while remaining within the state-of-the-art performance territory. Its building block is the "fire" module that consists of a "squeeze" 1×1 convolutional layer to reduce the number of input channels, followed by 1 × 1 and 3 × 3 "expand" convolutional layers that are connected in parallel to the "squeezed" output. SqueezeNet consists of 8 such modules connected in series.

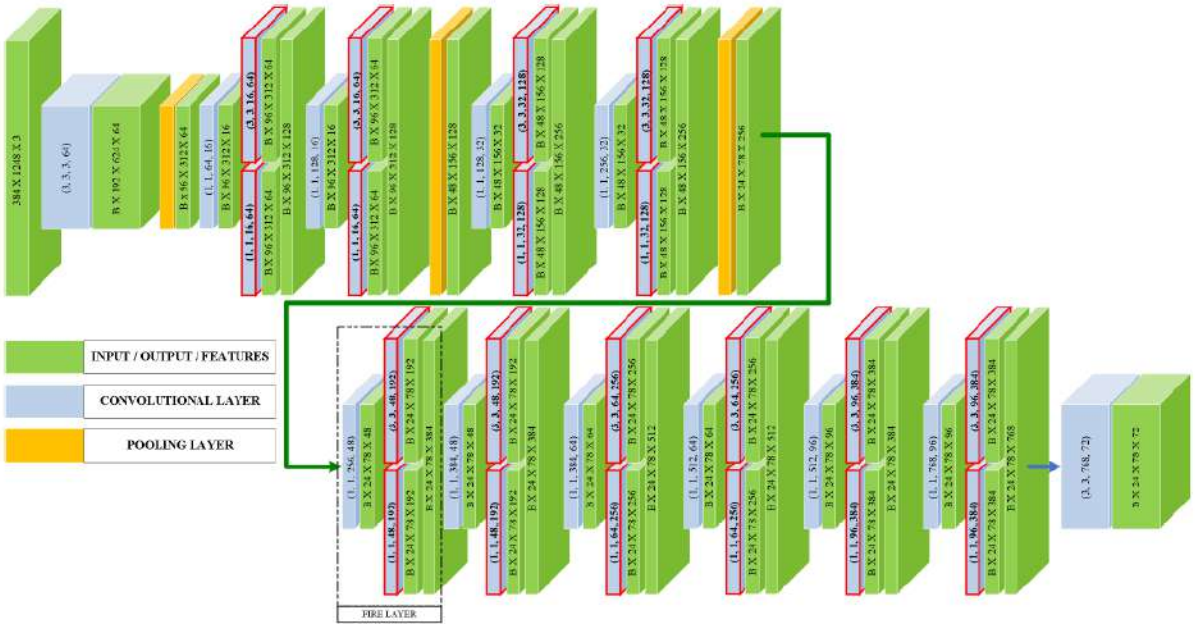


Figure 126: Architectures of the employed detector network. The convolutional layers highlighted by the red frames constitute the target layers in our acceleration experiments. $\boldsymbol{B}$ is the batch size, $\boldsymbol{H}$ is the height and $\boldsymbol{W}$ is the width of a volume kernel. $\boldsymbol{C_{L-1}}$ is the number of channels of the previous layer.

The PointPillars network is designed for 3D object detection using LiDAR point clouds. Its architecture (shown in Figure 127) consists of three main stages. More specifically, the first stage transforms the point cloud into a pseudo-image by grouping the points of the cloud into vertical columns, called pillars, that are positioned based on a partition of the x – y plane. The second stage consists of a feature extraction backbone network providing high-level feature-rich representations of the input. Finally, object detection

takes place in the third stage, which is responsible for producing 3D bounding boxes and confidence scores for the classes of interest.

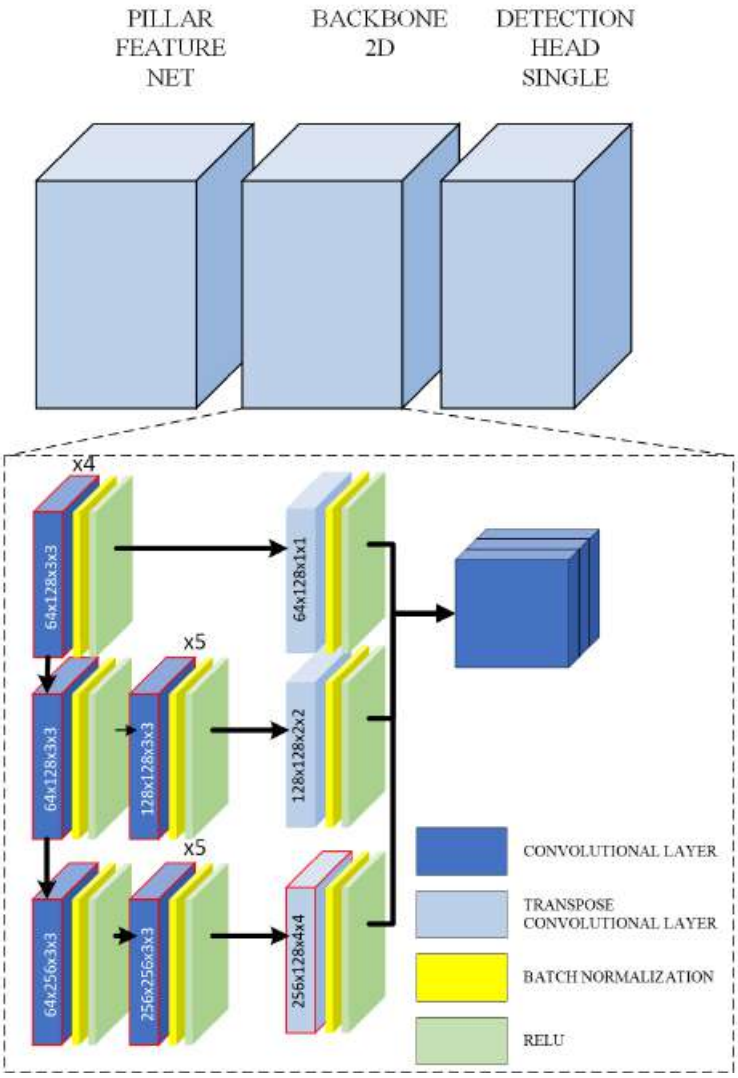


Figure 127: Architectures of the employed detector network Point-Pillars. The convolutional layers highlighted by the red frames constitute the target layers in our acceleration experiments.

In this experiment, we follow the acceleration strategy proposed in [22], whereby isolated parts of the network (e.g., individual layers) are quantized progressively, in stages, beginning at the original network. After each stage, the remaining original layers are retrained (or, fine-tuned).

Concerning SqueezeDet, the focus is on its feature extraction part, namely consists of 8 “fire” modules connected in series. SqueezeNet is responsible for roughly 83% of the total 5.3 × 109 MAC operations and 76% of the approximately 16 MB storage space required by SqueezeDet. Since it constitutes an already

efficient network, we only targeted SqueezeNet's "expand" layers in our experiments. Acceleration was performed in 8 acceleration stages (one "expand" module per stage), followed by fine-tuning.

Concerning PointPillars, its feature-extraction (backbone) stage is responsible for 97.7% of the total MAC operations required. In total, the Pointpillars network encompasses 4.835 × 106 parameters and requires 63.835 × 109 MACs. For a good balance between acceleration and accuracy loss, we only targeted the convolutional layers of the backbone network comprising the second stage of PointPillars. Specifically, the targeted 2D- and 4×4 transposed 2D- convolutional layers, are responsible for approximately 47% and 44.4% of the total required MACs, respectively. Acceleration was performed on 16 acceleration stages with each stage involving the quantization of a particular layer, followed by finetuning. Using the acceleration ratios α = 10, 20, 30, and 40 on the targeted layers leads to a reduction of the total required MACs by 82%, 86%, 88%, and 89%, or equivalently, to total model acceleration of PointPillars by 5.6×, 7.6×, 8.6×, and 9.2×, respectively.

Table 18: Acceleration and compression gains for the SqueezeDet and PointPillars networks under study, concerning both the Feature Extraction (FE) part and the Total model.

Model	**SqueezeDet**	
Gain	**MAC reduction (%)**	
Model Part	**FE**	**Total**
$a = 10$	74	60
$a = 20$	78	65
Model	**PointPillars**	
Gain	**MAC reduction (%)**	
Model Part	**FE**	**Total**
$a = 10$	84	82
$a = 20$	88	86
$a = 30$	91	88
$a = 40$	92	89

For each detection, the IOU score is computed as the ratio of the area of intersection to the area of union between the predicted and ground-truth bounding boxes. A true positive occurs when IOU > λ and the predicted class is the same as the ground-truth class, for some predefined threshold λ. A false positive occurs when IOU < λ or a different class is detected, meaning that unmatched bounding boxes are taken as false positives for a given class. Precision, recall and mean average precision (mAP) are subsequently calculated according to [23]. It is important to highlight that the performance of the image detector, the LIDAR detector and their fusion is measured using the 2D benchmark via projecting the bounding boxes to the 2D modality space.

the initial network architectures are compared with the accelerated ones via the Vector Quantization and Dictionary Learning approaches for an acceleration ratio $a = 10$ Table 19 presents the results for 2D, 3D and fusion-based object detection using the average precision (AP) per class and per difficulty, for all objects within the dataset. As the table reveals, in all cases, the fusion of modalities generates better results

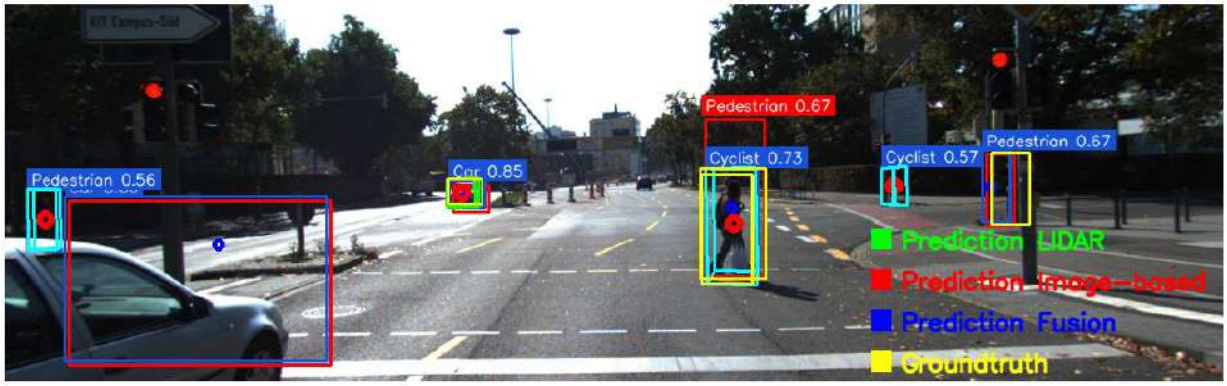
than each detector's ones, showcasing the acceptable performance of even a simplistic late fusion approach. The compression of the models in all cases, deteriorates the detection outcome of the individual

Table 19: Average precision for detection network and their respective fusion. Acceleration approaches $VQ_{a=10}$ and $DL_{\alpha=10}$ demonstrate the robustness of multi-modal fusion as an approach to combine the benefits of weak detectors.

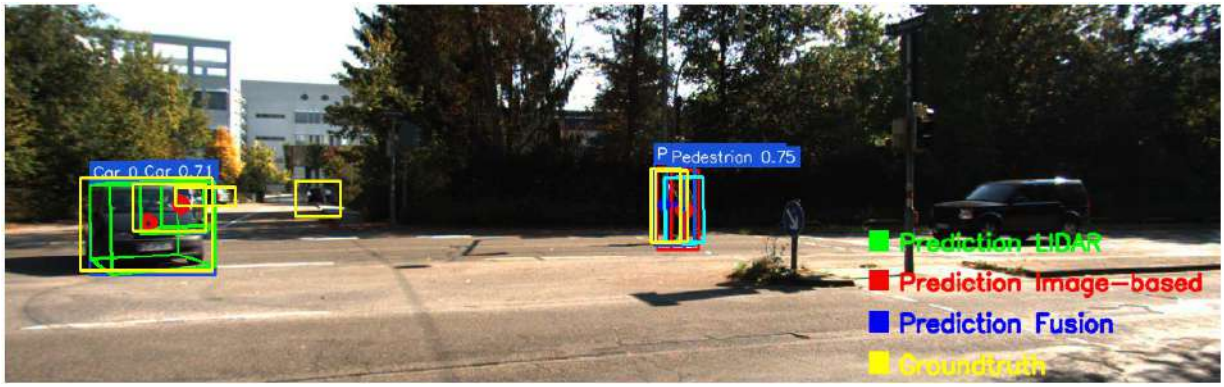
		Original			$VQ_{a=10}$						$DL_{a=10}$					
Class	*Difficulty*	*Fusion*	*Image*	*LIDAR*	*Fusion*		*Image*		*LIDAR*		*Fusion*		*Image*		*LIDAR*	
Car	Easy	83.8	63.7	78.0	83.3	**-0.5**	56.8	-6.9	77.8	-0.2	82.6	-1.2	58.9	-4.8	77.8	-0.2
	Moderate	85.4	56.4	79.0	83.7	-1.7	46.8	-9.6	78.9	-0.1	84.0	**-1.4**	53.4	-3.0	78.8	-0.2
	Hard	84.7	55.5	79.0	77.7	-7.0	45.1	-10.4	78.8	-0.1	77.7	-7.0	46.8	-8.6	78.7	-0.3
Pedestrian	Easy	86.2	76.4	78.3	80.1	-6.2	57.8	-18.6	75.5	-2.8	81.4	**-4.9**	62.8	-13.6	76.3	-2.0
	Moderate	86.1	76.4	68.1	64.4	-21.8	42.7	-33.7	64.4	-3.7	70.2	**-15.9**	50.9	-25.4	65.9	-2.2
	Hard	85.0	69.3	60.8	63.4	-21.6	41.8	-27.6	58.4	-2.5	69.3	**-15.8**	50.2	-19.1	59.4	-1.5
Cyclist	Easy	71.2	68.4	56.8	59.2	-12.0	49.2	-19.2	46.0	-10.9	65.3	**-5.9**	53.7	-14.7	47.8	-9.0
	Moderate	70.3	63.2	57.3	59.1	-11.2	44.8	-18.4	46.5	-10.9	61.4	**-8.9**	52.5	-10.8	48.3	-9.0
	Hard	68.3	61.4	50.2	58.1	-10.2	43.3	-18.0	46.7	-3.5	59.9	**-8.4**	46.6	-14.8	48.2	-2.0

detectors as the highlighted columns indicate. However, it is interesting to note that the late fusion approach improves the performance of the overall model even when the MCA techniques are applied, resulting in accelerations of about 2.5× and 6× for the 2D and 3D detectors, respectively, while the performance loss of the fused results ranging in most cases within single digits figures (as low as around 1% for the class "cars"). Comparing the performance of the utilized uni-modal detectors, it becomes readily apparent, that the 3D LiDAR based detector is much more resilient with respect to the incurred accuracy loss due to the application of acceleration/ compression. This comes as a direct consequence of the fact that SqueezeNet (i.e., the back-bone network of SqueezeDet) constitutes an already optimized lightweight network, as opposed to Pointpillars, whose architecture is much more "redundant" in the number of filters/parameters. Additionally, it can be observed that the performance of the DL-based weight-sharing MCA technique, is universally better than the one obtained via the VQ-based approach This indicates as expected that the gains in terms of weight approximation (i.e., quantization) error presented in Experiment I, are translated to analogous gains concerning the performance loss of the accelerated networks.

Finally, indicative detection results from KITTI examples and CARLA dataset is demonstrated in Figure 128:



(a): KITTI example



(b): KITTI example

(c): CARLA example

(d): CARLA example

Figure 128: Indicative detection results from AI Accelerated Multimodal Scene Analysis and Understanding module using KITTI and CARLA examples.

4.3.2.7 Robustified cooperative awareness for cooperative localization

4.3.2.7.1 Local information diffusion (see [9] for more details)

Impact of unknown multi-hop neighbors: Proposed solution Graph Laplacian Awareness using local diffusion LMS (GLA-LDL) aims to overcome the limitation of global awareness in the case of lacking location information of multi-hop neighbors at the initialization stage. Parameter p indicates the rate of these vehicles whose position is unknown by the ego vehicle. Note that for GLA-LDL and GLA p is always equal to 100%. The results of this study with p = 0, 10, 20% are depicted inFigure 129 and Table 20 with rc = 30m. The size of cluster which ego vehicle belongs to ranges between 2 and 97 vehicles, while the size of its neighbourhood is between 2 and 11. As Figure 129- (a) shows, our solution succeeds in hitting Average Location Awareness Error (ALAE) lower than 1.6m in less than 5 iterations, while GLLME (see D3.3) surpasses 1.8m and remains stable after the 10th iteration. This behaviour is also apparent in Table 20, where GLA-LDL reduced RT − LAE of standalone position source by 45%, with respect to 24% of GLA, 42% of GLLMS (see D3.3) and 43% of GLLME. It also proves the benefits of combining the overlapping estimations of vehicles following (10), since the without combination step GLA is much less accurate. Increasing p to 10% and 20%, Figure 129-(b), (c) prove that global awareness is totally unable to converge, significantly increasing ALAE beyond 20m. The proposed solution remains unaffected by this since it was explicitly developed in order to counter this issue. Therefore, we conclude that as p increases, so does GLLMS and GLLME remove away from convergence, while GLALDL achieves a significant reduction of RT −

LAE in less than 5 iterations. We verify that defining a dynamic target location vector for each one of the vehicles instead of a common across them, it benefits CA ability in CAVs.

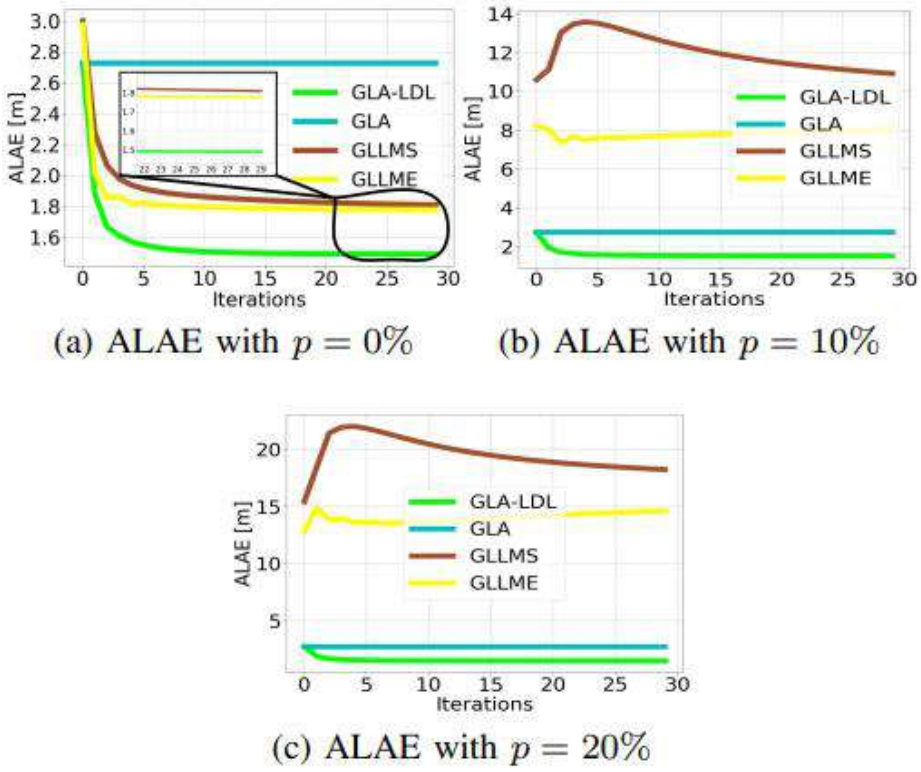


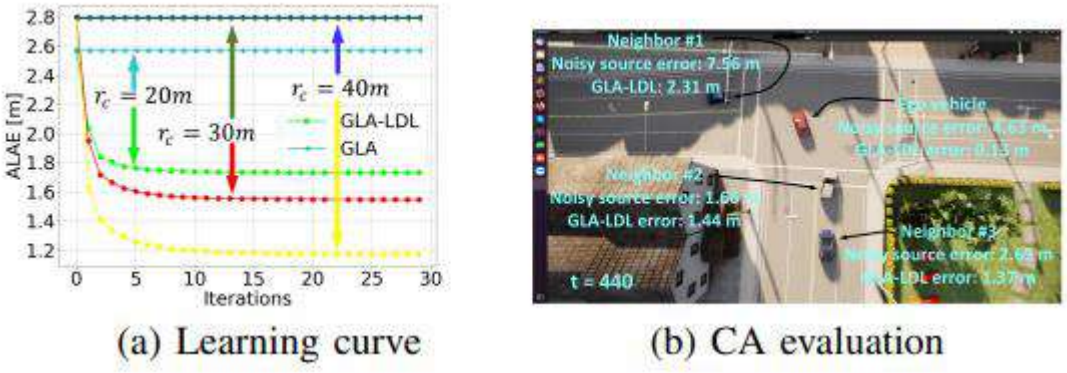
Figure 129: Convergence with different rates of unknown multi-hop neighbours

Table 20: Root mean square error over time horizon of location awareness error RT-LAE (m) VS number p of unknown multihop neighbors

p	$GLA - LDL$	GLA	$GLLMS$	$GLLME$	$Noisy\ source$
0%	2.1	2.9	2.2	2.16	3.91
10%	2.1	2.9	14.7	13.8	3.91
20%	2.1	2.9	22.16	22.16	3.91

Impact of connections: In order to evaluate the convergence ability and location accuracy of our scheme in the case of dynamic and sparse communication topologies, we studied the effect of communication range rc. The latter parameter indicates the number of vehicles which comprise the specific cluster, as well as how dynamic the topology of connections is. With rc = 40m cluster's size ranges between 2 and 145, while neighborhood's size between 2 and 13. Varying size of corresponding topologies, indicate dynamic connection links among vehicles. The results are summarized in Figure 130 and Table 21 We choose p = 0%, thus GLLMS and GLLME performances can be considered biased, though very close to GLA-LDL. Increasing r_c, RT – LAE of GLALDL reduces, reaching its minimum 1.89m at $r_c = 40m$, reducing by 50% the

error of noisy source. This performance should be correlated with the fact that RT − LAE of GLA slightly increases as r_c increases, verifying that with the neighborhood's size growing larger, CA task seems to be more challenging. Figure 130-(a) proves that considering larger amount of overlapping estimations through larger combination weights positively influences convergence speed. For example, ALAE at 10th iteration is equal to 1.8, 1.6 and 1.2m, respectively for $r_c = 20, 30, 40m$. Therefore, we can conclude that the combination step of the proposed GLA-LDL provides the scalability to modifying topologies, facilitating the efficient integration of neighbouring vehicles' estimations. The accuracy which ego vehicle estimates its neighbours' positions at time instant t = 440 and r_c = 30m is shown in Figure 130-(b).



(a) Learning curve (b) CA evaluation

Figure 130: Impact of neighbours' number and CARLA visualization

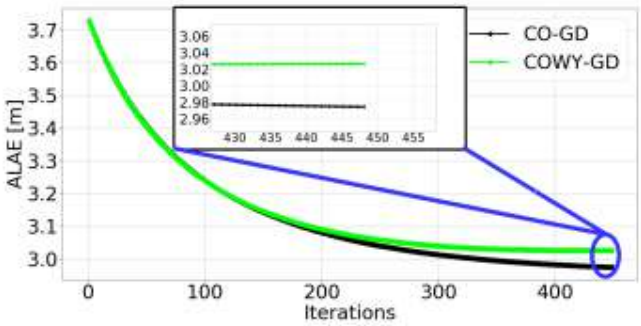
Table 21: RT-LAE (m) VS communication range r_c

r_c	*Size of neighborhood*	$GLA - LDL$	*GLA*	*GLLMS*	*GLLME*	*Noisy source*
20m	$2-9$	2.22	2.76	2.11	2.07	3.74
30m	$2-11$	2.1	2.9	2.2	2.16	3.91
40m	$2-13$	1.89	2.93	1.86	1.82	3.81

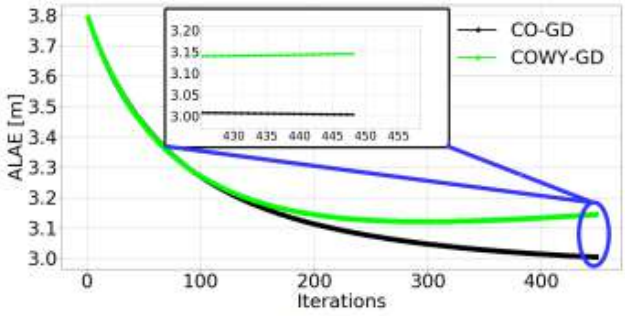
4.3.2.7.2 Collaborative odometry estimation

Convergence ability: The convergence performance of our algorithm Collaborative Odometry using Gradient Descent (CO-GD) is shown in Figure 131, under three different choices of communication range rc. The latter parameter dictates the number of connected neighbours to each ego vehicle i. More specifically, number of neighbours ranges between 1- 7, 1-9 and 1-11, for r_c = 20m, 30m, 40m, respectively. Figure 131 presents the ALAE of vehicle i, so as to depict the convergence behaviour in the mean sense. For L_{max} = 450 iterations, the proposed CO-GD succeeds in reducing the location awareness error at every iteration, reaching almost 3m at the end of optimization. In all three cases, it is apparent that our scheme successively minimizes location error. On the contrary, Collaborative Odometry without yaw using Gradient Descent (COWY-GD) which doesn't consider yaw as an unknown parameter, behaves much worse when the number of neighbours grows larger. During the optimization procedure the solution of COWY-GD gets away from the minimum, especially in the third case. This is due to the fact that the noise of angle measurement model caused by noisy yaw and relative measurement, which isn't addressed by the algorithm, seriously impacts on the accumulation of position estimation error. Therefore, we can conclude that larger number of neighbours don't influence negatively our scheme (instead of COWY-GD) in terms of

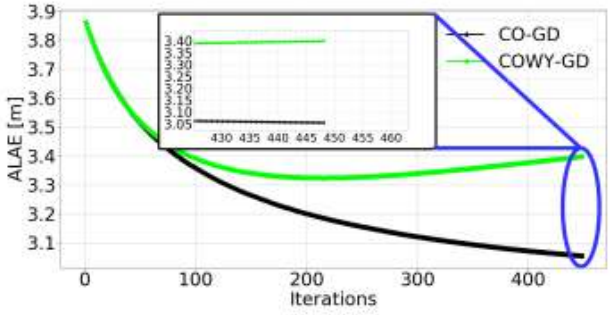
location awareness, while it is expected to be more accurate should the L_{max} increases. Scalability to the size of neighbourhood is attained in the case of self-position and yaw estimation, as pointed out next.



(a) ALAE with $r_c = 20m$



(b) ALAE with $r_c = 30m$



(c) ALAE with $r_c = 40m$

Figure 131: Convergence with different communication ranges

Time complexity: Averaged (over L_{max} and T) timing results for every iteration of both schemes are summarized in Table 22, for different communication ranges. Clearly, our method is more time consuming than COWY-GD, reaching 2.6 msec when $r_c = 40$, due to the two optimization problems which have to be

solved. Choice of Lmax in practical considerations is related to these timing outcomes, since optimization procedure has to be completed within the time interval dt.

Table 22: Averaged time results for every iteration (msec)

Communication range r_c	$CO-GD$	$COWY-GD$
$20m$	1.6	0.3
$30m$	2	0.5
$40m$	2.6	0.6

4.3.2.7.3 Multimodal odometer solutions for autonomous vehicles

The proposed multi-modal fusion approach has been evaluated in the CARLA-ROS framework. Three state-of-the-art SLAM solutions were employed (see details in D3.1): DSO, ORB-SLAM and Lego-LOAM. The first test scenario includes DSO and ORB-SLAM, while in the second scenario DSO and Lego-LOAM were fused, adopting also a novel strategy for selecting the useful landmarks. In all cases, we assume that an operational IMU provides 2D velocity readings which is degraded by Gaussian noise with zero mean and standard deviation equal to $\sigma_v = \alpha * actual\ velocity$. The latter value is chosen so as to simulate realistic accumulative sensor stressing. Note that ground truth velocity has been generated by CARLA. Unless mentioned otherwise, $\alpha = 0.1$.

Test scenario 1: The simulation horizon was set to $T = 3000$ time instances with $dt = 0.033\ sec.$ At each time instant, DSO and ORB-SLAM generate the vehicle's pose and the landmarks of the surrounding environment. The trajectory of vehicle is shown in Figure 132, where we can see that DSO is very close to ground truth. The localization error of DSO and ORB-SLAM versus time is also highlighted in Figure 133. Clearly, ORB-SLAM behaves much worser than DSO, reaching almost $20\ m$ error, instead of DSO which is constantly below $5\ m$. The results of our framework are shown in Figure 134, where DSO Lapl Tr, ORB Lapl Tr and Fusion, correspond to the first, second and final branch of our solution. We must mention that we found experimentally that reducing the number of involved landmarks, more accurate results can be generated. To do so, we employ a naïve strategy: choose landmarks whose distance with the corresponding pose is below a fixed threshold. For the first branch (DSO pose and ORB-SLAM landmarks) the threshold is equal to $thr_1 = 200\ m$, while for the second branch $thr_2 = 50\ m$. Note that Figure 134 depicts the corresponding Cumulative Distribution Function (CDF) of localization error. Root mean square errors (RMSEs) of translational absolute pose errors (APEs) are also shown in Table 24. The latter indicates that DSO Lapl Tr branch achieved the lowest RMSE, even lower (by ~ $1m$) than DSO, as well as that ORB Lapl Tr reduced the high error of ORB-SLAM by ~ $1.3m$. However, the proposed Fusion branch didn't achieve the lowest error, resulting between DSO Lapl Tr and ORB Lapl Tr. However, this initial testing scenario indicates the potential of our framework in reducing APE of the original solutions. In the following scenario, where the two original solutions are close enough in terms of APE and not exceed significantly one another (instead of the current scenario), the Fusion branch is proven that indeed enhances the performance of both solutions, not just one of them. As a last remark, we plotted the trajectory of Fusion in Figure 135. As we can see, the proposed solution has very "bad" tracking behavior since the individual localization errors are quite diverse. This is due to the fact that the poses of ORB-SLAM are significantly non-accurate. Despite this unattractive performance, we notice at $t = 1551$ where ORB-SLAM reaches its maximum error, Fusion achieves significantly lower APE, i.e., $2.1\ m$ instead of $19.6\ m$.

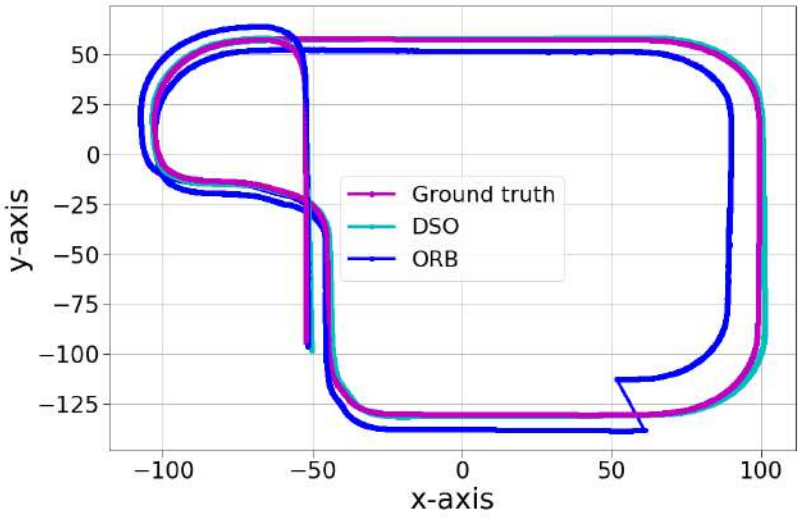


Figure 132: Ground truth, DSO and ORB-SLAM trajectories

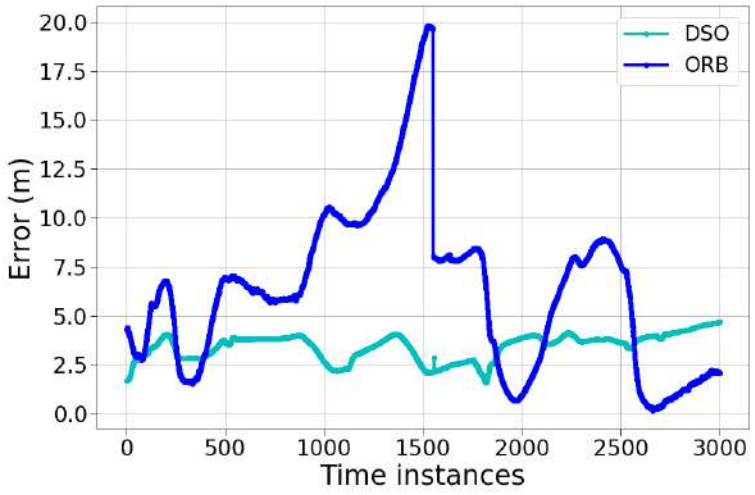


Figure 133: Localization error versus time horizon for DSO and ORB-SLAM

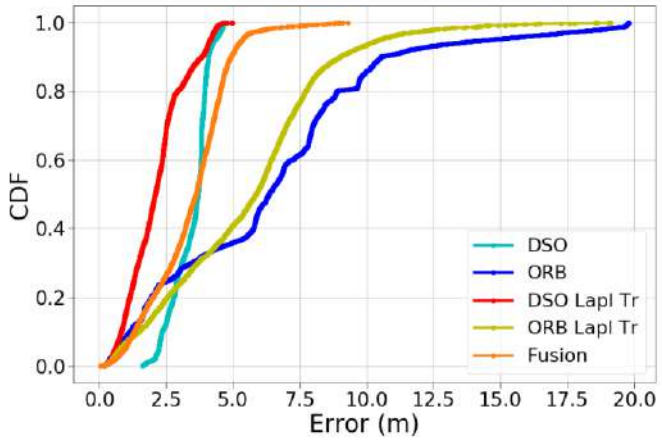


Figure 134: CDF of localization error using DSO and ORB-SLAM

Table 23: RMSE results for test scenario 1

	DSO	ORB-SLAM	DSO Lapl Tr	ORB Lapl Tr	Fusion
RMSE (m)	3.49	7.59	2.36	6.3	3.67

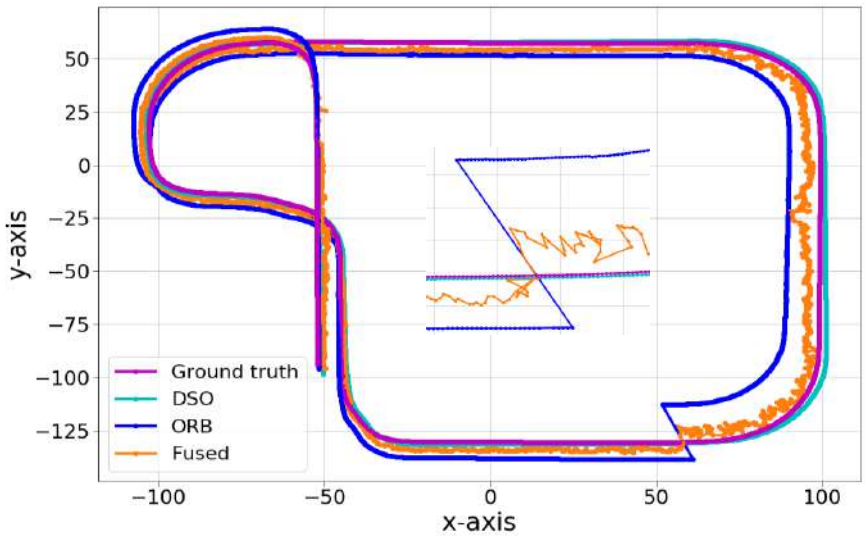


Figure 135: Fusion trajectory for test scenario 1

Test scenario 2: The simulation horizon was set to $T = 921$ time instances, while $dt = 0.098\ sec$. Figure 136 depicts ground truth, DSO and Lego-LOAM trajectories. Evidently, both odometry solutions are very close to the ground truth, which is also verified in the Figure 137 of localization error versus time horizon. More specifically, DSO achieves accuracy below $4\ m$, while Lego-LOAM below $2.5\ m$ for the entire simulation. Thus, both solutions have attractive performance. In order to use the proposed re-localization framework, we have to reduce the number of involved landmarks (as mentioned before). To do so, we measure the distances of one group of landmarks to the related pose, formulate the corresponding histogram, and keep only the first bin. Afterwards, the threshold is determined by the average value of the first 10% of the corresponding distances. In this adaptive way, a novel strategy for selecting landmarks at each time instant can be applied. The indicative CDF of DSO, Lego-LOAM and Fusion branch is shown in Figure 138. Apparently, Fusion has achieved lower localization error and outperforms both odometry solutions, proving the benefits of the proposed framework. More specifically, Table 23 indicates the performance of Fusion using the novel selection of landmarks, with respect to choosing fixed distance thresholds equal to $thr_1 = 1\ m$ for the first branch and $thr_2 = 10\ m$ for the second branch. Proposed Fusion with the adaptive threshold has reduced the RMSE of Lego-LOAM and DSO by 31% and 53%, respectively. Using the fixed thresholds (though determined by *trial-and-error* experimentation), the reduction was equal to 27% and 50%, slightly lower than before. Moreover, the performance of DSO Lapl Tr and Lego Lapl Tr using the adaptive strategy has also shown its potential. Both of them have achieved almost the same accuracy as the Fusion branch. It is evident that the adaptive strategy of selecting landmarks is more beneficial is terms of the acquired performance. Therefore, we conclude that the purpose of a more accurate re-localization system has been achieved, through the proposed multi-modal fusion framework. Fusion resulting trajectory is shown in

Figure 139, while the corresponding localization error versus time horizon is depicted in Figure 140. As a last remark, we measured the RMSE of translational relative pose error (RPE) for the Fusion branch, DSO and Lego-LOAM for different parameter values of velocity noise. More specifically, $\alpha = 0, 0.05, 0.1$. The results are shown in Table 25. Clearly, as σ_v grows larger so does the RPE of Fusion. This is something that should be expected since the noisy velocity readings from IMU are feeding the prediction step of Kalman, and negatively impact its performance. However, in all cases Fusion RPE is slightly larger than DSO's and lower than Lego-LOAM's.

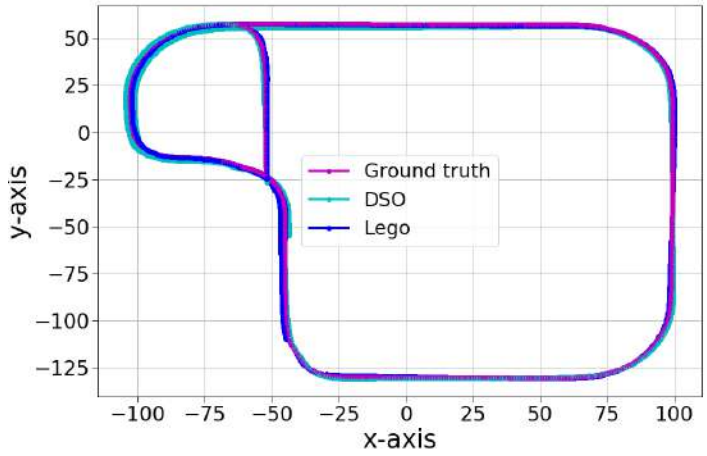


Figure 136: Ground truth, DSO and Lego-LOAM trajectories

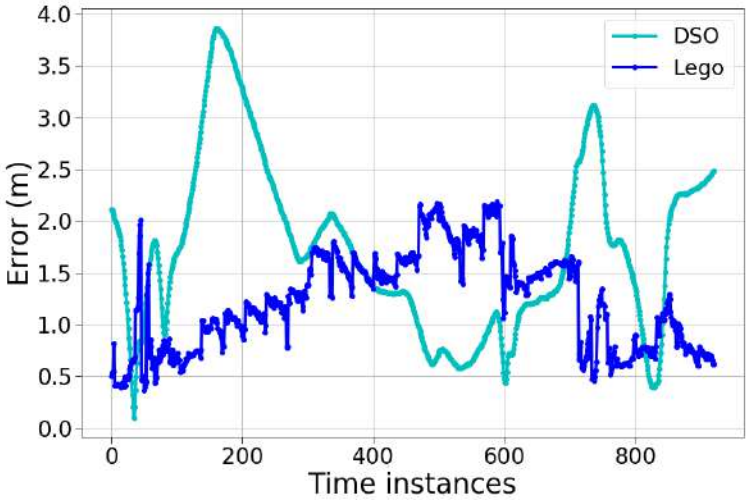


Figure 137: Localization error versus time horizon for DSO and Lego-LOAM

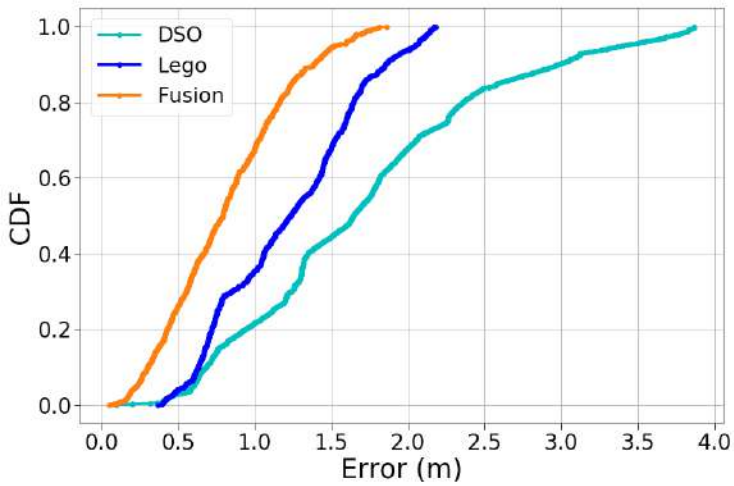


Figure 138: CDF of localization error using DSO, Lego-LOAM and Fusion

Table 24: RMSE results for test scenario 2

	DSO	Lego-LOAM	DSO Lapl Tr (adapt. threshold)	Lego Lapl Tr (adapt. threshold)	Fusion (adapt. threshold)	Fusion (fixed threshold)
RMSE (m)	1.91	1.3	0.91	0.9	0.9	0.94

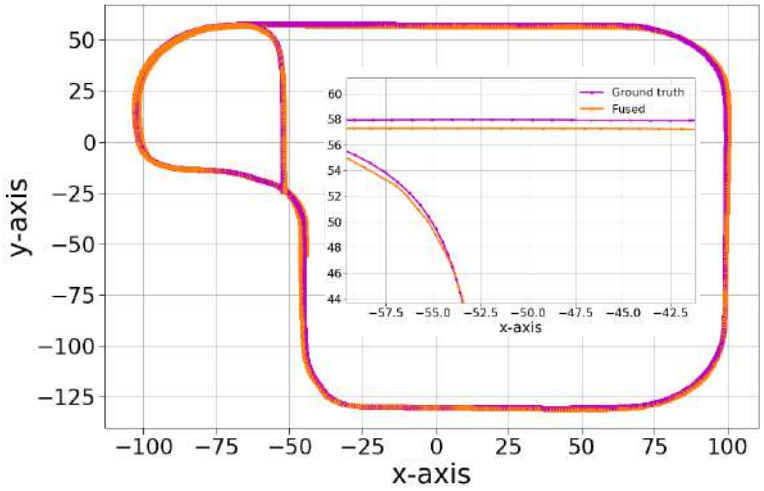


Figure 139: Fusion trajectory for test scenario 2

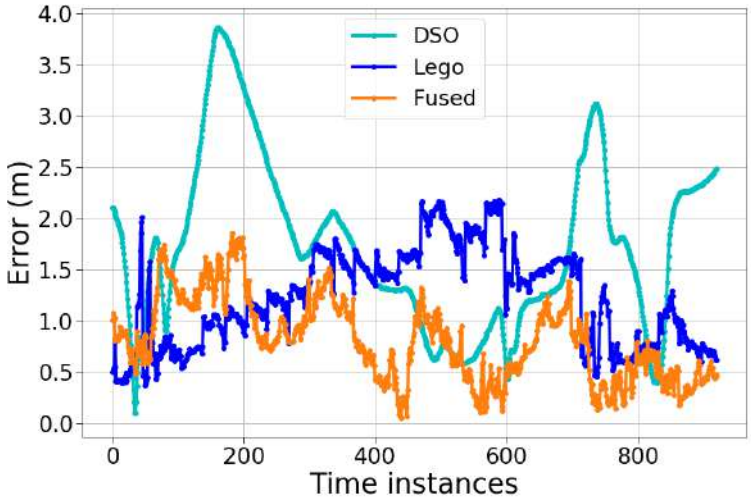


Figure 140: Fusion localization error versus time

Table 25: RMSE of RPE (m) for test scenario 2

a	DSO	Lego-LOAM	Fusion
0	0.05	0.14	**0.06**
0.05	0.05	0.14	**0.07**
0.1	0.05	0.14	**0.11**

Finally, an indicative example showing the combined integration of the three previous modules in the CARLA-ROS framework is demonstrated in Figure 141.

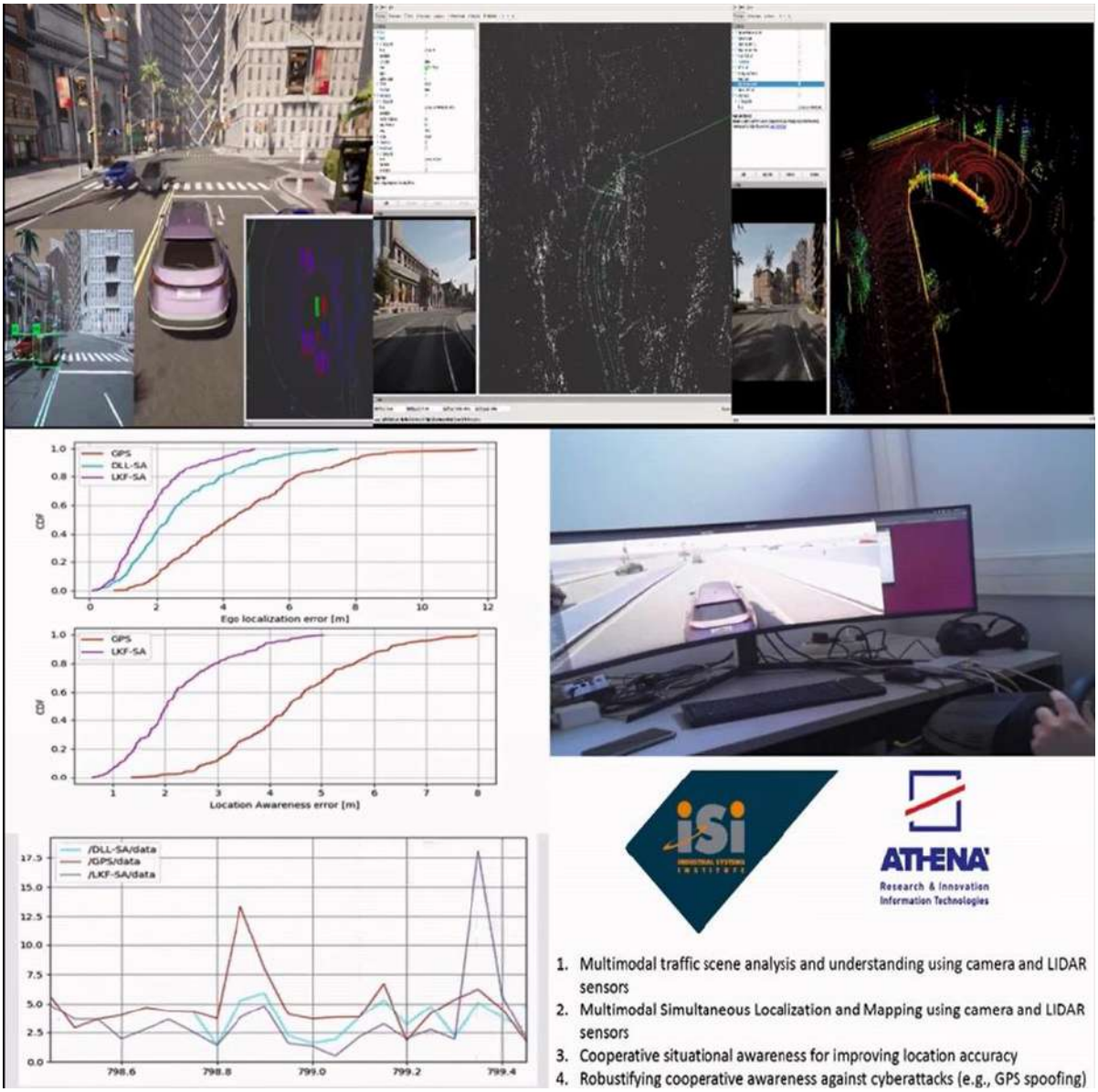


Figure 141: Combined modules of AI accelerated multimodal scene analysis and understanding (top left), robustified cooperative awareness (bottom left) and multimodal odometers (top right) deployed in the CARLA-ROS framework.

4.3.3 Co-Operative Awareness based on Static Agents

One of the major targets of Co-Operative Awareness is to establish Sparse MAp RegisTration (SMART) functionality, ideally defined as a bijective transformation between two versions of the map, coming from consecutive observations. This would mean that all the landmarks detected in one map would correspond to just one landmark in the second map. Using landmarks instead of points in a dense point cloud is the reason we use the word sparse in the naming of the module.

4.3.3.1 Problem to be Solved

The problem to be solved is using the detected landmarks in a way in which the outliers are ignored, and registration is performed with a minimal spatial error inflicted, so that the vehicle localization is in turn more accurate. To achieve this, we have to get enough corresponding landmarks from the Landmark Extraction module in subsequent iterations, so that registration is feasible.

4.3.3.2 Map based Registration

Our system architecture is designed such that a map-to-map landmark registration process will be needed. Therefore, in this section we will discuss this type of method. In [29], the matching process happens per class, with a cost function minimization criterion calculated for each of the landmark pairs. The criterion is different for each landmark class (facades, poles and road markings). This is a very interesting approach for us, since the idea of reusing our mature modules will probably lead to a heterogeneous landmark matching problem. Let's look at some details on that. The aim is to find associations between elements of the local map M_l and global map M_g: To do so, the solution contributed to CPSoSAWARE, introduced shifting the estimated vehicle pose p for best alignment of M_l and M_g: The quality of alignment is given then by the cost function given by:

$$\boldsymbol{C(M_l, M_g, p) = \sum_{f_d \in M_l} C_f(f_d, M_g, p)} \quad \text{(Equ. 1)},$$

where f_d denotes a detected feature.

The single feature association cost $C_f\,(f_d, M_g, p)$ is defined as following:

$$\boldsymbol{C_f(f_d, M_g, p) = \begin{cases} C_p(f_d, M_g, p), & if\ pole \\ w_{f,1} C_s(f_d, M_g, p), & if\ facade \\ w_{f;2} C_s(f_d, M_g, p), & if\ road\ marking \end{cases}} \quad \text{(Equ. 2)}$$

where $w_{f;1}$ and $w_{f;2}$ are weighting factors of the different landmark types. For reference, pole association costs $C_p\,(f_d; M_g;\ p)$ are defined as:

$$\boldsymbol{C_p(f_d, M_g, p) = \begin{cases} \frac{d_p(f_t, f_m, p)}{d_{p,max}} + w_p \frac{\Delta a_p(f_d, f_m, p)}{\Delta a_{p,max}} & if\ \exists f_m \\ 1 + w_p & else \end{cases}} \quad \text{(Equ. 3)}$$

A match $f_m \in M_g$ exists if it is similar in radius, inclination angle and position to the detection f_d. In case of multiple matches, the match with the lowest cost C_p is chosen. The pole matching cost is build-up of two terms ( ) first case). The first term penalizes a high distance between f_d and f_m and the second term penalizes a large angle difference. The $2D$ point-to-point distance d_p is normalized with the maximum distance $d_{p;max}$ of a matched pole pair $(f_d;\ f_m)$ The angle $\Delta\alpha_p$ between the poles is normalized with the maximum angle $\Delta\alpha_{p;max}$ of a match. If no matching pole could be found, the maximum cost is set for the pole f_d() second case).

The association costs for facades and road markings are computed the same way by using segment matching costs:

$$C_s(f_d, M_g, p) = \begin{cases} \frac{d_s(f_d,f_m,p)}{d_{s,max}} + w_{s,1}\frac{\Delta\alpha_s(f_d,f_m,p)}{\Delta\alpha_{s,max}} + w_{s,2}\frac{0_{out}(f_d,f_m,p)}{0_{in}(f_d,f_m,p)+0_{out}(f_d,f_m,p)}, & if\ \exists\ f_m \\ 1 + w_{s,1} + w_{s,2}, & else \end{cases} \quad \text{(Equ. 4)}$$

Two segments are a match if they meet these criteria:

- They belong to the same type (facade, dashed line, stop line, etc.)
- They are close distance-wise.
- They overlap substantially.

The segment matching cost (Eq. 5 first case) is built up of three terms. The first term penalizes the distance between the segments. More precisely, ds are the distance between the center point of f_d to the line segment f_m: This distance is independent of the difference in orientation and therefore is more suitable than real segment-to-segment distance. The second term penalizes differences in $2D$ orientation $\Delta\alpha_s$ The third term considers the overlap of the two segments. We define a measure for overlap which is independent of the difference in orientation. One segment is rotated around the center point so that the segments are parallel. Then O_{in} is the overlapping length and O_{out} is the non-overlapping length when projecting one segment onto the other. The parameters $d_{s;max}$ and $\Delta\alpha_{s,max}$are used for normalization and ws;1 and ws;2 are weighting factors. Figure **142** visualizes the three terms of segment matching costs. If no match was found, the maximum cost for the detection f_d is set ( ) second case). In [34] [18], the detected pole landmarks are projected on the $2D$ map with their center and width information and each new local map that overlaps with a previous one is checked for matching pole landmarks, which are merged to form a new one. In [34] [3], the matching process is two-fold; the road markings are matched on map space and then the location hypotheses generated are verified by querying the image in specific ROIs for the presence of road signs.

4.3.4 Iterative Closest Point (ICP)

The ICP algorithm aims to minimize the Euclidean distance between the input data (detections from the sensor) and reference data (global map). Let two sets of $2D$ points, source A (with n points) and target B (with m points) $\subset \mathbb{R}^2$.

We seek a transformation function $u : A \longrightarrow B$ that minimizes the mean squared distances (MSD) between A and B:

$$MSD(A, B, u) = \frac{1}{n}\sum_{a\in A, b\in B}\|b - u(a)\|^2 \quad \text{(Equ. 5)}$$

Incorporating the rotation (R) and translation (t) matrices into the MSD equation, the minimization problem becomes:

$$\min_{u:A\longrightarrow B}\sum_{i=1}^{n}\left\|b_i - R_{a_i} - t\right\|^2 \quad \text{(Equ. 6)}$$

Hence, the ICP tries in each iteration to minimize the $MSD(A,B,u)$ by switching between a matching and a transformation stage. Matching stage: In this first stage, the objective is to minimize the $MSDs\ MSD(A,B,u)$ by finding the best correspondence between a point $a_i \in A$ and $b_i \in B$: This step is in its most basic form executed by selecting the point $b_i \in B$ with the minimum Euclidean distance to the point $a_i \in A$: Note that in the first iteration, t and R is normally set to $[0;\ 0]^T$ and to the identity matrix respectively. Transformation stage: During the transformation stage, the objective is to compute the optimal R and t that minimize) using the correspondences computed in the previous stage. ICP uses a simple least square solver to find the optimal linear transformation matrix $(R|t)$ that minimizes). For this purpose, the algorithm starts by subtracting from the reference and sensor point clouds their respective centroid, as shown in the following equations:

$$\acute{\boldsymbol{a}}_{\boldsymbol{\iota}} \;\; = \boldsymbol{a_i} - \frac{\mathbf{1}}{\boldsymbol{n}} \sum_{\boldsymbol{i=0}}^{\boldsymbol{n}} \boldsymbol{a_i} \qquad \text{(Equ. 7)}$$

$$\acute{\boldsymbol{b}}_{\boldsymbol{\iota}} \;\; = \boldsymbol{b_i} - \frac{\mathbf{1}}{\boldsymbol{m}} \sum_{\boldsymbol{i=0}}^{\boldsymbol{m}} \boldsymbol{b_i} \qquad \text{(Equ. 8)}$$

This step helps simplifying the minimization problem before the cross-variance matrix is computed using the following equation with $\acute{A}$ and $\acute{B}$ being the set of points $a_i'$and b_i' respectively:

$$\boldsymbol{H} = \acute{\boldsymbol{A}}\acute{\boldsymbol{B}}^{\boldsymbol{T}} \qquad \text{(Equ. 9)}$$

Now, the rotation angle θ can be computed from:

$$\boldsymbol{\theta} = \boldsymbol{atan2}\left(\left(\boldsymbol{H}(\mathbf{0},\mathbf{1}) - \boldsymbol{H}(\mathbf{1},\mathbf{0})\right), \left(\boldsymbol{H}(\mathbf{0},\mathbf{0}) + \boldsymbol{H}(\mathbf{1},\mathbf{1})\right)\right) \qquad \text{(Equ. 10)}$$

It can be shown that, the optimal solution for R and t that minimizes the objective function is given by equations) and) below, where $\bar{a}$ and $\bar{b}$ are the points centroid computed in equations) and). In the end of the transformation stage, the source / sensor data is transformed using the estimated $(R|t)$ matrix and the algorithm goes back to the matching stage (with $(R|t)$ set to the identity matrix), unless a stopping criterion is verified (e.g. number of iterations, Euclidean error improvement between iterations, etc).

$$\boldsymbol{R} = \boldsymbol{m}\begin{bmatrix} \boldsymbol{\cos\theta} & \boldsymbol{-\sin\theta} \\ \boldsymbol{\sin\theta} & \boldsymbol{\cos\theta} \end{bmatrix} \qquad \text{(Equ. 11)}$$

$$\boldsymbol{t} = \bar{\boldsymbol{b}} - \boldsymbol{R}\bar{\boldsymbol{a}} \qquad \text{(Equ. 12)}$$

4.3.5 Software Structure

SmaRt functionalities are inside LME. When they will be used in a standalone module, the interfaces will change accordingly. The pose output produced by LME is better suited as an output here, while not showing the full functionality. For calling LME functions, please refer to the LME documentation. The part relevant to SmaRt, is the calls related to ICP, which is also a class and instantiated as a member variable of RoadClass, once per camera. Once the landmarks are added to the list of a RoadSide object, ICP can be called using first the function to add all points to the point cloud. This function allows for augmentation

of Parking Slot landmarks, by adding more "virtual" points on the line defining them (from entry point to back point):

1. void addAllPts2ICP(i32 numOfVirtPts);

We can also To have the trained point cloud to compare to, we have to somehow load it from memory. At the moment, for the Algo C implementation this is done by dumping the detected landmarks at the phase of training to several text _les, then loading them in the beginning of the testing phase. For the Proof of Concept stage, saving is done once, when the car is stopped at the drop-o_ zone and loading is done in the beginning of the test drive. The functions to do this, are also in LME:

2. void saveDropO_(); void loadDropO_();

Then, 2d ICP based registration can take place using:

3. void calcPose(const RoadSide& trainedRS, bool smooth = false);

This function performs ICP registration and pose estimation at once, allowing the user to smooth the results so that the median value of past estimations is used, avoiding jitter in the output. The core function of PMD:

4. bool run (Interface::Map const& interfaces, KeyboardFlags const& keyboardFlag s) override;

Update and get the output to the interface of LME, that carries the :

5. void updateInterface (TimeStamp time);
6. pmd::interfaces::PMD getInterfaceObject(void);

4.3.6 Module Description

The subsequent sections will define the functionality developed in this module at various levels of abstraction ranging from presenting the overall solution pipeline 4.3.6.2 to the explanation of the approaches followed by each module of the pipeline 4.3.6.3. Further explanation on integration strategies, software dependencies etc. are analyzed in D5.4.

4.3.6.1 Main Idea

Our work follows a single camera approach, which allows for redundancy in cases where one of the cameras is defective, or soiled. It also allows late fusion of the results to create an optimized global map. The main process is based on an Iterative Closest Point implementation running on the trained vs. the testing point clouds. These two-point clouds are enhanced versions of the most prominent point on the detected landmarks, which now are only PMD related (closest vertex for lines, entry point for slots). The most important landmarks are the slots in our case, so they can be augmented by additional points on their defining midline.

4.3.6.2 Block Diagram

The general block diagram of SmaRt and its interaction with LME in its intended form as in Figure 142

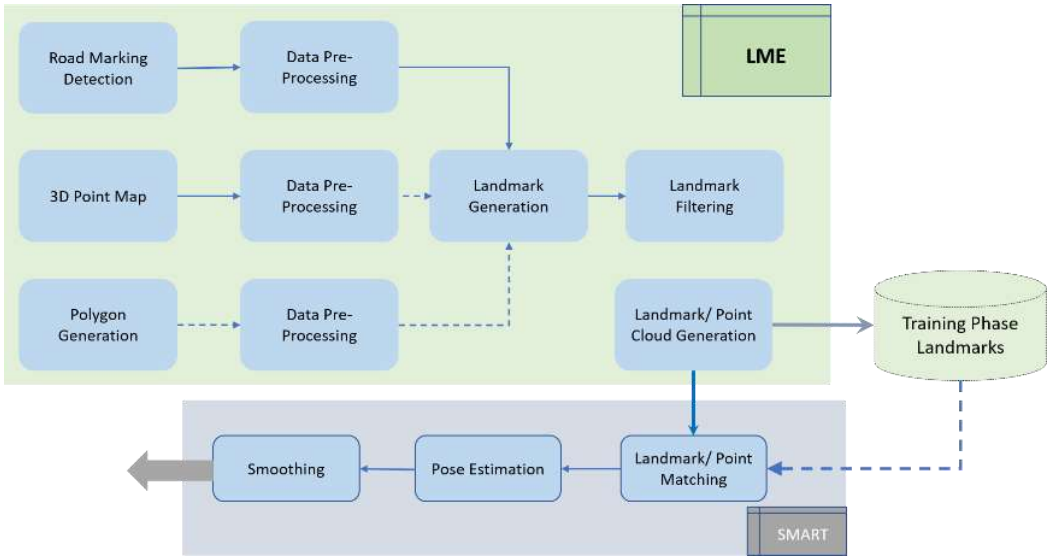


Figure 142: The block diagram of LME and SmaRt

4.3.6.3 *Experiments & Results*

Once we have loaded the trained point cloud and detected the new one on the y, ICP can be called to do the heavy lifting for point registration and pose estimation. This is shown in Figure 143.

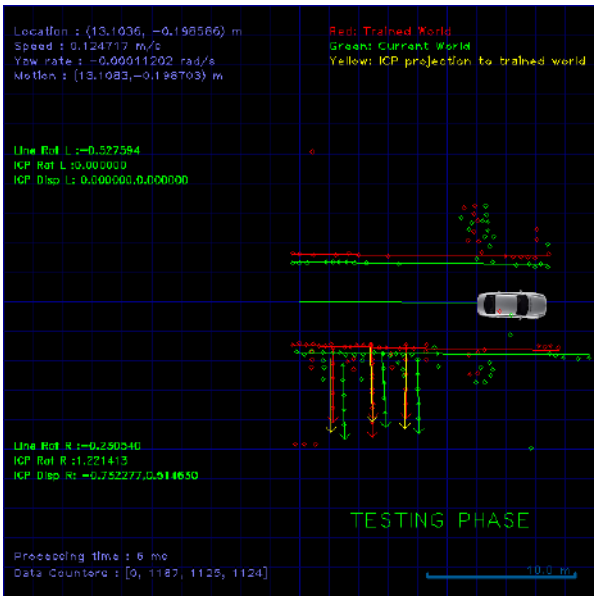


Figure 143: **A basic visualization of the module. Only Parking Slot matchings are shown, with arrows. Red for trained, green for tested, yellow for tested that are projected to the trained world. Red and green circles are the two-point clouds used**

In Figure 144 we can see more details of the ICP results. Additionally, to the elements show in Figure 143 we can also observe the current (testing) trajectory to the trained world and the ICP projections of all points

of the point cloud (yellow lines connecting projected points). The results stemming from ICP can be further enhanced with a fusion with the roadside lines fitted by LME. This way, there are a lot of outliers to be rejected and the results of ICP can be further improved.

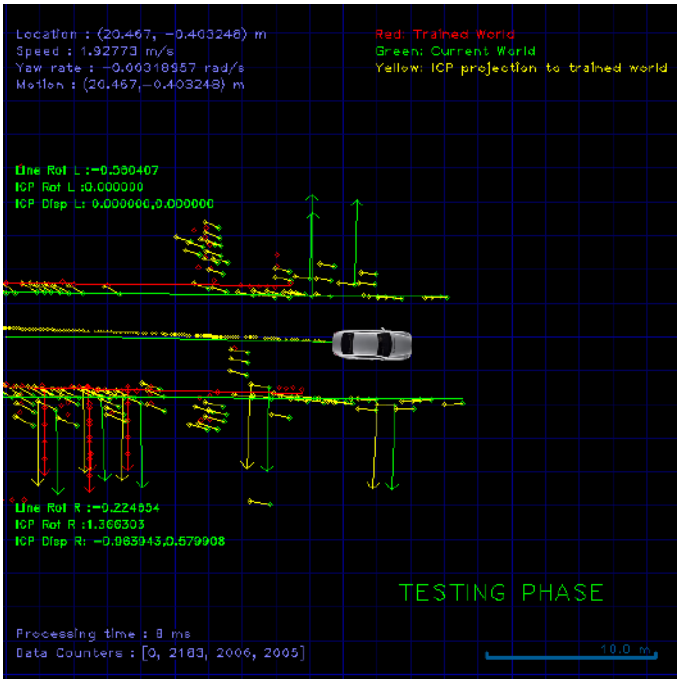


Figure 144: A more detailed visualization of the module. Trajectory projection and projected point cloud also shown.

During the Demonstration phase of CPSoSAWARE project, which took place in Frankfurt, the co-operative awareness demo was set as it is illustrated in Figure 146 from the vehicle interior and Figure 147 from the vehicle exterior. To be technically able to apply motion planning in the co-operative localization demo, the basic idea of planning in traditional robotics engineering needs to be adapted to the automotive industry requirements. Due to the huge size of required maps, long driving lengths, security and safety concerns and dynamics limitations of the automotive section, the motion planning algorithms which are being employed in the automotive industry should satisfy these concerns. To satisfy the requirements of Co-operative Localization, motion planning in autonomous vehicles has been divided in three main sections: global, local planning and collision avoidance:

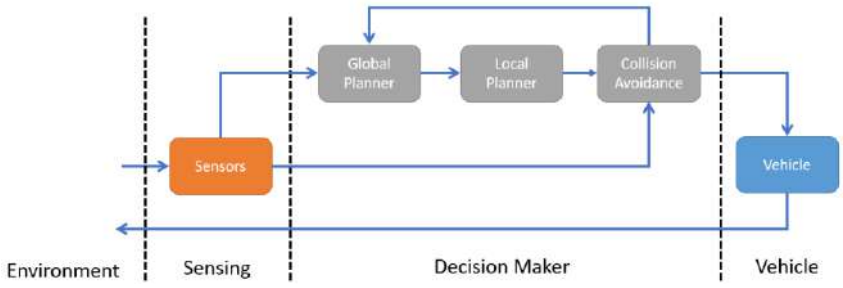


Figure 145: The framework of an autonomous system.

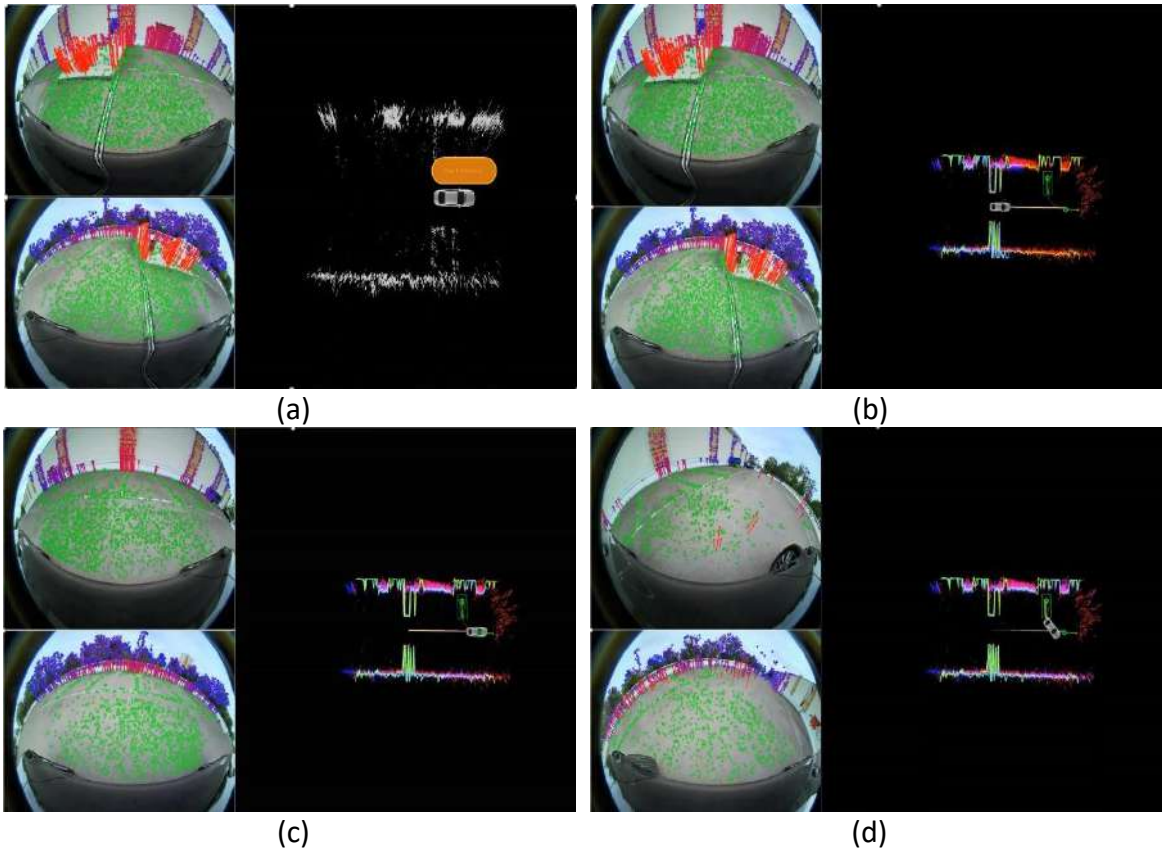
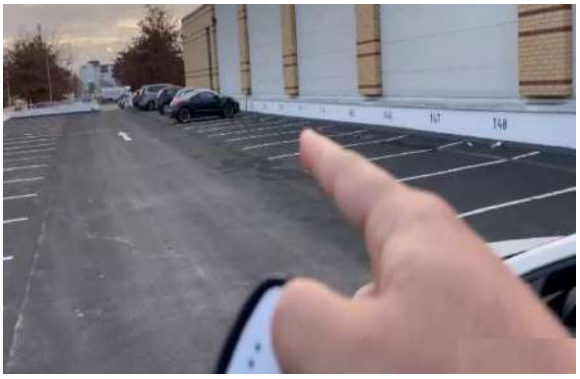


Figure 146: Co-Operative Localization Demo as it is visualized in the vehicle interior. (a) The testing phase starts, (b)-(d) Landmark registration between the prior map (yellow landmarks) and the landmarks derived from the current scan of the environment.

(a)

(b)

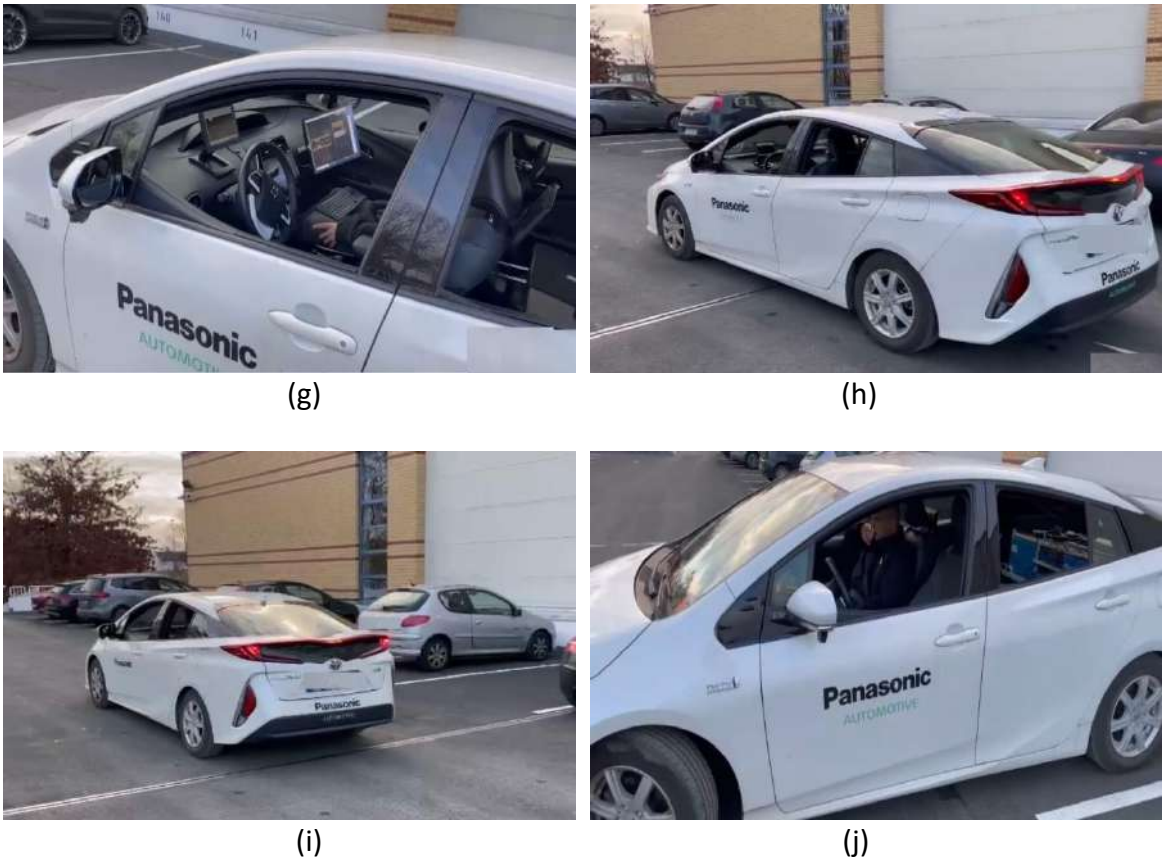


(c)

(d)

(e)

(f)



(g) (h)

(i) (j)

Figure 147: The Co-operative Localization Demonstration as it is illustrated from the exterior of the vehicle. (a)-(j) correspond to the stages depicted in Figure 146.

5 Conclusions & Further Work

The Demonstrated pilots of CPSoSAWARE identifies two key areas where innovations are introduced namely (a) Industrial Pilot and (b) Automotive Pilot. As a result, the three areas have been identified to be: (1) Human Driver/operator Monitoring, (2) Cyber-Security and (3) Co-operative awareness, upon which the penetration testing was performed.

This document describes the activities that were performed in the context of Task 6.4 '*Small Trial Demonstration and Validation*' of the CPSoSAWARE project. The document details the testing scenarios, protocols and results that were followed by the Trials to assess the CPSoSAWARE solution with its components that follow the specifications of discussed in D6.2 and D6.4. The current deliverable describes the results and the specific attacks for the three main pillars of CPSoSAWARE.

6 References

[1] EN ISO 10218-1:2011 Safety requirements for industrial robots. Part 1: Robots.

[2] EN ISO 10218-2:2011 Safety requirements for industrial robots. Part 2: Robot Systems and integration

[3] ISO/TS 15066:2016 Robots and robotic devices - Collaborative robots

[4] https://www.vipm.io/package/ros_for_labview/

[5] https://encyclopedia.pub/entry/20326

[6] https://www.baeldung.com/cs/direct-linear-transform#camera-calibration

[7] https://ergo-plus.com/wp-content/uploads/RULA-A-Step-by-Step-Guide1.pdf

[8] https://www.microsoft.com/en-us/hololens

[9] N. Piperigkos, A. S. Lalos, K. Berberidis and C. Anagnostopoulos, "Cooperative Multi-Modal Localization in Connected and Autonomous Vehicles," 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS), 2020, pp. 1-5, doi: 10.1109/CAVS51000.2020.9334558.

[10] 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), 2020, pp. 1-6, doi: 10.1109/CAMAD50429.2020.9209312

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik, “Rich feature hierarchies for accurate object detection and semantic segmentation,” in Proceedings of the IEEE conference on computer vision and pattern recognition, 2014, pp. 580–587.

[12] E.-V. Pikoulis, C. Mavrokefalidis, and A. S. Lalos, “A new clustering-based technique for the acceleration of deep convolutional networks,” in Proceedings of the IEEE International Conference on Machine Learning and Applications (ICMLA), 2020.

[13] Pikoulis, E.V., Mavrokefalidis, C., Nousias, S., Lalos, A.S. (2022). A New Clustering-Based Technique for the Acceleration of Deep Convolutional Networks. In: Wani, M.A., Raj, B., Luo, F., Dou, D. (eds) Deep Learning Applications, Volume 3. Advances in Intelligent Systems and Computing, vol 1395. Springer, Singapore. https://doi.org/10.1007/978-981-16-3357-7_5.

[14] N. Piperigkos, A. S. Lalos and K. Berberidis, "Robustifying cooperative awareness in autonomous vehicles through local information diffusion," 2022 IEEE 20th International Conference on Industrial Informatics (INDIN), 2022, pp. 442-447, doi: 10.1109/INDIN51773.2022.9976168.

[15] N. Piperigkos, A. S. Lalos and K. Berberidis, "Alternating optimization for multimodal collaborating odometry estimation in CAVs," 2022 30th Mediterranean Conference on Control and Automation (MED), 2022, pp. 670-675, doi: 10.1109/MED54222.2022.9837156.

[16] https://ieee-dataport.org/documents/realistic-vehicle-trajectories-and-driving-parameters-carla-autonomous-driving-simulator

[17] H. Wymeersch, J. Lien, and M. Z. Win, "Cooperative localization in wireless networks," Proceedings of the IEEE, vol. 97, no. 2, pp. 427– 450, 2009.

[18] P. Yang, D. Duan, C. Chen, X. Cheng and L. Yang, "Multi-Sensor Multi-Vehicle (MSMV) Localization and Mobility Tracking for Autonomous Driving," in IEEE Transactions on Vehicular Technology, vol. 69, no. 12, pp. 14355-14364, Dec. 2020, doi: 10.1109/TVT.2020.3031900.

[19] https://github.com/CarlaScenes/CarlaSence

[20] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in Proceedings of the IEEE conference on computer vision and pattern recognition, 2016, pp. 770–778.

[21] F. N. Iandola et al., "Squeezenet: Alexnet-level accuracy with 50x fewer parameters and< 0.5 mb model size," arXiv:1602.07360, 2016

[22] J. Cheng, J. Wu, C. Leng, Y. Wang, and Q. Hu, "Quantized cnn: A unified approach to accelerate and compress convolutional networks," IEEE Transactions on Neural Networks and Learning Systems, vol. 29, no. 10, pp. 4730–4743, 2018

[23] D. M. Powers, "Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation," arXiv preprint arXiv:2010.16061, 2020.

[24] N. Piperigkos, A. S. Lalos, and K. Berberidis, "Multi-modal cooperative awareness of connected and automated vehicles in smart cities," in 2021 IEEE International Conference on Smart Internet of Things (SmartIoT), 2021, pp. 377–382

[25] M Ring, J Dürrwang, F Sommer, R Kriesten «Survey on vehicular attacks-building a vulnerability database», In IEEE International Conference on Vehicular Electronics and Safety (ICVES), pages: 208-212.

[26] *Dimc, F.; Balžec, M.; Borio, D.; Gioia, C.; Baldini, G.; Basso, M.* «An experimental evaluation of low-cost GNSS jamming sensors», NAVIGATION J. Inst. Navig. 2017, 64, 93–109.

[27] *Wang, B.; Yu, L.; Deng, Z.; Fu, M.* «A particle filter-based matching algorithm with gravity sample vector for underwater gravity aided navigation», IEEE/ASME Trans. Mechatron. 2016, 21, 1399–1408.

[28] *Yang, C.; Strader, J.; Gu, Y.; Hypes, A.; Canciani, A.; Brink, K.* «Cooperative UAV Navigation using Inter-Vehicle Ranging and Magnetic Anomaly Measurements », In Proceedings of the Guidance, Navigation, and Control Conference, AIAA, Kissimmee, FL, USA, 8–12 January 2018; p. 1595.

[29] *Canciani, A.; Raquet, J.* « Absolute positioning using the Earth's magnetic anomaly field», Navig. J. Inst. Navig. 2016, 63, 111–126.

[30] Teixeira, F.C.; Quintas, J.; Maurya, P.; Pascoal, A. Robust particle filter formulations with application to terrain-aided navigation. Int. J. Adapt. Control Signal Process. 2017, 31, 608–651

[31] Marques, L.; Nunes, U.; de Almeida, A.T. Olfaction-based mobile robot navigation. Thin Solid Film. 2002, 418, 51–58.

[32] Rock, S.M.; Hobson, B.; Houts, S.E. Return-to-site of an AUV using terrain-relative navigation: Field trials. In Proceedings of the 2014 IEEE/OES Autonomous Underwater Vehicles (AUV), Oxford, MS, USA, 6–9 October 2014; pp. 1–8

[33] *Julius Kümmerle, Marc Sons, Fabian Poggenhans, Tilman Kühner, Martin Lauer, and Christoph Stiller*, «Accurate and efficient self-localization on roads using basic geometric primitives», In IEEE International Conference on Robotics and Automation (ICRA 2019), pages 5965-5971.

[34] *Alexander Schaefer, Daniel Büscher, Johan Vertens, Lukas Luft, and Wolfram Burgard*, «Long-term urban vehicle localization using pole landmarks extracted from 3-D lidar scans», In IEEE European Conference on Mobile Robots (ECMR 2019), pages: 1-7.

[35] *Kyoungtaek Choi, Jae Kyu Suhr, and Ho Gi Jung*. «Map-Matching-Based Cascade Landmark Detection and Vehicle Localization», IEEE Access, 7:127874-127894, 2019

7 Appendixes

Appendix 1 - List of sequences for the DMS Application testing

Long Distraction. Duration of the sequence: about 5 seconds each.

No.	Sequence type	Euro NCAP distraction

1.	LD Gaze forward	No
2.	LD Gaze upward	No
3.	LD Gaze downward	No
4.	LD Rearview mirror gaze	Yes
5.	LD Left mirror gaze	Yes
6.	LD Right mirror gaze	No
7.	LD Instrument cluster gaze	Yes
8.	LD Infotainment gaze	Yes
9.	LD Gearbox gaze	No
10.	LD Glovebox	Yes
11.	LD Gaze upward	No
12.	LD Gaze downward	No
13.	LD Rearview mirror gaze	Yes
14.	LD Left mirror gaze	Yes
15.	LD Right mirror gaze	Yes
16.	LD Instrument cluster gaze	No
17.	LD Infotainment gaze	Yes

18.	LD Gearbox gaze	No
19.	LD Driver side window	Yes
20.	LD Passenger side window	Yes
21.	LD Passenger footwell	Yes
22.	LD Passenger face	Yes
23.	LD Body lean toward passenger footwell	Yes
24.	LD Body lean toward rear passenger	Yes

Other Sequences (OS). Duration of the sequence: 5 seconds each.

No.	Sequence type	EuroNCAP distraction
	OS Head rolled to the left (eyes opened)	No
	OS Head rolled to the right (eyes opened)	No
	OS Head rolled to the left (eyes closed)	No
	OS Head rolled to the right (eyes closed)	No
	OS Head pitched up (eyes closed)	No
	OS Head pitched down (eyes closed)	No

Short Distraction (SD-VATS). Duration of the sequence: 10 seconds of shifting between the location and the forward view (on the road).

No.	Sequence type	EuroNCAP distraction
25.	VATS Gaze upward	No
26.	VATS Gaze downward	No
27.	VATS Rearview mirror gaze	Yes
28.	VATS Left mirror gaze	Yes
29.	VATS Right mirror gaze	No
30.	VATS Instrument cluster gaze	Yes
31.	VATS Infotainment gaze	Yes
32.	VATS Gearbox gaze	No
33.	VATS Glovebox	No
34.	VATS Passenger footwell	Yes
35.	VATS Driver side window	Yes
36.	VATS Combination of non-driving locations	Yes
37.	VATS Gaze upward	No
38.	VATS Gaze downward	No

39.	VATS Rearview mirror gaze	Yes
40.	VATS Left mirror gaze	Yes
41.	VATS Right mirror gaze	Yes
42.	VATS Instrument cluster gaze	No
43.	VATS Infotainment gaze	Yes
44.	VATS Gearbox gaze	No
45.	VATS Driver side window	No
46.	VATS Passenger side window	Yes
47.	VATS Passenger footwell	Yes
48.	VATS Passenger face	No

Phone Use (PU-VATS). Duration of the sequence: 10 seconds of shifting between the phone and the forward view (on the road).

No.	Sequence type	EuroNCAP distraction
49.	PU Holding phone in 9-11 o'clock position of the SW	Yes
50.	PU Holding phone in center of the SW	Yes
51.	PU Gaze at phone at infotainment position	Yes

52.	PU Left driver knee	Yes
53.	PU Right driver knee	Yes
54.	PU Driver lap	Yes
55.	PU Left side of the dashboard	Yes
56.	PU Phone held in view of windscreen	Yes
57.	PU Phone held in view of instrument cluster	Yes
58.	PU Phone mounted in forward view of windscreen	Yes
59.	PU Left driver knee	Yes
60.	PU Right driver knee	Yes
61.	PU Gaze at phone at infotainment position	Yes
62.	PU Left side of the dashboard	Yes
63.	PU Driver lap	Yes

Appendix 2

This appendix contains descriptions of tests conducted to assess the application's functionality. Each table is a description of a separate use case test.

Table 3.1. Description of a use case test: installation.

#	1	Test case name	Application installation

Hardware	Samsung A13, serial: R58T503KNFN
System version	Android 12, One UI Core 4.1 & Android 9, EMUI 9.1.0
Purpose	Checking the process of installation from User's point of view.
Description	The installation process was carried out in accordance with the instructions attached to the application.
Steps	1. Downloading *cpsosAware_application.apk* file to the device from Google Drive 2. Installing application from the file 3. Launching *dsm_cpsos_mikit* application on the smartphone
Assessment criteria	Checking whether the process of instalation is successful on phones with different operating systems.
Outcome	Installation successful.

Table 3.2. Description of a use case test: configuration of the application on the phone with the smartwatch.

#	1.1	Test case name	Galaxy Watch configuration
Hardware		Samsung Galaxy Watch 4, serial: RFAR72TZZWZ	
System version		System version 11, One UI 4.0, Wear OS 3.2	
Purpose		Checking the process of Galaxy Watch configuration from User's point of view.	
Description		The configuration process was carried out in accordance with the instructions attached to the application.	

Steps	1. Downloading *Galaxy Wearable (Samsung Gear)* to the smartphone from Google Play Store 2. Connecting with the watch by Bluetooth. 3. Adding Galaxy Watch4 (ZZWZ) to the Galaxy Wearable device list 4. Turning on *Developer mode* on Galaxy Watch 5. Enabling the toggle for *ADB debugging* and *Debug over Wi-Fi* in *Developer options* 6. Note IP address: 192.168.0.181 7. Downloading cpsosAware_*wearable.apk* file to the device from Google Drive 8. Downloading *Easy Fire Tools* to the smartphone from Google Play Store 9. Setting IP compatible with Galaxy Watch 10. Connecting with the watch 11. Allowing the debugging request 12. Installing *cpsosAware_wearable.apk* on the watch using *Easy Fire Tools* 13. Lunching installed application on Galaxy Watch
Assessment criteria	The only criteria in this test case were the success or failure of the installation process of the two applications in the Galaxy Watch and the success or failure of the configuration of the application on the phone and the application on the Samsung Galaxy watch.
Outcome	Configuration successful with some difficulties.

Table 3.3. Description of a use case test: configuration of the application on the phone.

#	2	Test case name	Starting the application and compatibility testing
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when the phone is blocked – for example not to disturb the driver with the landmarks during the drive, but still wake the drive up if he/she falls asleep.	
Description		The application was turned on and then, after approximately 30 seconds of working phone was blocked. It was blocked for 3 minutes to come, then unlocked.	
Steps		1. Starting the driving session on the tested DMS application 2. Leaving the phone for ~30 seconds 3. Blocking the phone 4. Unlocking the phone 5. Ending driving session 6. Checking the output JSON file	
Assessment criteria		Checking whether data from the time when the phone was blocked would have been gathered anyway and would have been in the output JSON file (success or failure in data gathering during the tested event).	
Outcome		Application does not work while the phone is blocked. In the JSON file there is a time gap for when the phone was blocked.	
Outcome		Configuration successful with some difficulties.	

Table 3.4. Description of a use case test: starting and ending a session.

#	3	Test case name	Starting session & ending session
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1 & Android 9, EMUI 9.1.0	
Purpose		Checking the basic functionality of the application.	
Description		Application has a limited number of functions, and starting and ending a session are included in those basic functions – that is why it should be tested at various times, at different levels of the phone's battery, and with varying amounts of RAM in use.	
Steps		1. Staring session on 100% of the phone's battery 2. Ending session after 2 minutes 3. Staring session on 10% of the phone's battery 4. Ending session after 2 minutes 5. Staring session with 5 other applications working in the background 6. Ending session after 2 minutes	
Assessment criteria		Whether any crashes or major lags occur while the application is running (success or failure in data gathering and application performance during the tested event).	
Outcome		No crashes or lags bigger than ones occurring during optimal use of the application. The basic functionality of the application is maintained in various conditions. NOTE: It is lagging more on the phone with older Android and EMUI operating system (should be inspected with different Android versions).	

Table 3.5. Description of a use case test: using the phone while the application is on (answering/making a call).

#	4	Test case name	Using the phone while the application is on (answering/ making a call)
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when the driver answers a call.	
Description		While the application was on for approximately 40 seconds, researcher took a scheduled call without ending the session. The call lasted for about 3 minutes. Then the session was terminated and the JSON file was checked.	
Steps		1. Starting session 2. ~40 seconds of uninterrupted application performance 3. Taking a call 4. Finishing the call 5. Ending session	
Assessment criteria		Checking whether data from the time when the researcher took a call would have been gathered anyway and would have been in the output JSON file (success or failure in data gathering during the tested event).	
Outcome		The application did not gather any data during the phone call. In the JSON file, there is a time gap for when the phone call was occurring (*Figure 3.1*).	

```
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 296,
  "frame_timestamp": "2022-11-14T09:11:05.301Z",
  "frame_uuid": "adee2191-552f-4f87-af7d-1a596d655ba3",
  "heartRate": "No HR Available Yet",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 297,
  "frame_timestamp": "2022-11-14T09:14:36.772Z",
  "frame_uuid": "a645c9d3-bfee-49ab-8184-b494f95bf181",
  "heartRate": "No HR Available Yet",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
```


Figure 3.1. A fragment of a JSON file showing the time gap in the *frame_timestamp* variable for when the call was answered while application was on.

Table 3.6. Description of a use case test: using the phone while the application is on (running in the background).

#	5	Test case name	Using the phone while the application is on- opening google maps (running in the background check)
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when it is running in the background – for example with the main focus being on Google Maps (drivers often use their phones as navigation in the car).	

Description	While the application was on for a few seconds researcher changed the focus to the Google Maps application. Then the session was terminated, and the JSON file was checked.
Steps	1. Starting session 2. A few seconds of uninterrupted application performance 3. Opening Google Maps 4. Going back to the application 5. Ending session
Assessment criteria	Checking whether data from the time when the researcher was looking at Google Maps would have been gathered anyway and would have been in the output JSON file (success or failure in data gathering during the tested event).
Outcome	The application did not gather any data during the time when Google Maps was used. In the JSON file, there is a time gap for when the phone call was occurring (*Figure 3.2*).

Figure 3.2. A fragment of a JSON file showing the time gap in the *frame_timestamp* variable for when the main focus was on Google Maps application.

Table 3.7. Description of a use case test: blocking the phone while the application is working.

#	6	Test case name	Blocking the phone while application is on
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when the phone is blocked – for example not to disturb the driver with the landmarks during the drive, but still wake the drive up if he/she falls asleep.	

Description	The application was turned on and then, after approximately 30 seconds of working phone was blocked. It was blocked for 3 minutes to come, then unlocked.
Steps	1. Starting the driving session on the tested DMS application 2. Leaving the phone for ~30 seconds 3. Blocking the phone 4. Unlocking the phone 5. Ending driving session 6. Checking the output JSON file
Assessment criteria	Checking whether data from the time when the phone was blocked would have been gathered anyway and would have been in the output JSON file (success or failure in data gathering during the tested event).
Outcome	Application does not work while the phone is blocked. In the JSON file there is a time gap for when the phone was blocked (*Figure 3.3*).

```
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 144,
  "frame_timestamp": "2022-11-09T09:32:42.594120Z",
  "frame_uuid": "d26e0c42-37cd-44b9-8613-02037bb64ca9",
  "heartRate": "0,0",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
{
  "alert_is_fired": "false",
  "distractionLeft": "false",
  "distractionRight": "false",
  "eyes_closed": "false",
  "frame_number": 145,
  "frame_timestamp": "2022-11-09T09:35:54.360963Z",
  "frame_uuid": "9047dec8-7620-4db4-9a85-66c08cccfd2d",
  "heartRate": "0,0",
  "number_of_detected_faces": "1",
  "yawning": "false"
},
```


Figure 3.3. A fragment of a JSON file showing the time gap in the *frame_timestamp* variable for when the phone was blocked.

Table 3.8. Description of a use case test: plugging the phone to the charger while the application is working.

#	7	Test case name	Plugging phone in while application is on
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when power supply status changes charging off → charging on).	
Description		Checking how the application will perform when the phone gets plugged into the charger while the application is on.	

Steps	1. Starting the session 2. Plugging the phone to the charger 3. Waiting for any reaction from the application 4. Ending session 5. Checking the output JSON file
Assessment criteria	Whether the application will in any way react to the power supply status change (success or failure in data gathering and application performance during the tested event).
Outcome	The application does not respond to plugging the phone into the charger. This is interpreted as the correct response.

Table 3.9. Description of a use case test: unplugging the phone from the charger while the application is working.

#	8	Test case name	Unplugging phone while application is on
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when the power supply status changes charging on → charging off).	
Description		Checking how the application will perform when the phone gets unplugged from the charger while the application is on.	

Steps	1. Starting the session 2. Unplugging the phone from the charger 3. Waiting for any reaction from the application 4. Ending session 5. Checking the output JSON file
Assessment criteria	Whether the application will in any way react to the power supply status change (success or failure in data gathering and application performance during the tested event).
Outcome	The application does not respond to disconnecting the phone from charging. This is interpreted as the correct response.

Table 3.10. Description of a use case test: turning the phone off while the application is working.

#	9	Test case name	Turning the phone off while the application is running
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		The purpose of this test was to check whether the application will save already gathered data or remove everything since the session does not have a formal ending. This was tested to check the performance of the application in cases such as a dying phone battery.	
Description		While the app was running, the phone was turned off to make it look like it was completely discharged.	

Steps	1. Starting the session 2. Leaving the application to run uninterrupted for ~30 seconds 3. Turning the phone off 4. Turning the phone on 5. Checking the JSON file
Assessment criteria	Checking whether data from the time before the phone got turned off would have been gathered and saved anyway and would have been in the output JSON file (success or failure in data gathering and saving before the tested event).
Outcome	The application did not save the data gathered up until the point of the phone being turned off.

Table 3.11. Description of a use case test: face recognition with *N = 1.*

#	10	Test case name	Face recognition of 1 face
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking the basic functionality of the application.	
Description		The effectiveness of detecting one human face in view was tested in a controlled environment. This functionality was tested by pointing the phone's camera at the human agent from different angles.	

Steps	1. Staring the session 2. Placing the camera directly in front of human agent 3. Placing camera approximately 45 degrees to the right from the human's face center 4. Placing camera approximately 45 degrees to the left from the human's face center 5. Ending the session
Assessment criteria	Checking whether the application would detect a human agent's presence with different phone placements (success or failure in face detection and recognition).
Outcome	The application flawlessly detects that there is a human agent in the view.

Table 3.12. Description of a use case test: face recognition with *N = 2*.

#	11	Test case name	Face recognition of 2 faces
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Verify that the application can handle more than one detectable face and correctly recognize events for both of them/ at least the driver.	
Description		In a controlled environment, two human agents were set in the application's view.	

Steps	1. Starting the application 2. Yawning of the agent no. 2 (fake rear passenger) 3. Yawning of the agent no. 1 (fake driver) 4. Ending the session
Assessment criteria	Checking if the application would detect the presence of both human agents and see which one the application would prioritize (success or failure in two faces detection and events recognition).
Outcome	The application flawlessly detects that there are two human agents in the lens, but recognizes events (such as yawning) only for the one that is closer to the camera. This is interpreted as the correct response.

Table 3.13. Description of a use case test: recognition and classification of a human-like object.

#	12	Test case name	Face recognition of human-like objects
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking whether the application can be deceived by human-like objects and hence omit the driver's face and events detection for the true human agent.	
Description		In a controlled environment, a human-like object (schematic drawing of the human's face) was set in the application's view.	

Steps	1. Starting the session 2. Directing the camera on a human agent (as driver) and human-like object (as rear passenger) 3. Directing the camera on a human-like object as a driver without human agent in the view 4. Ending the application
Assessment criteria	Checking if the application would detect the presence of a human-like object and classify it as a human (success or failure in human-like object detection).
Outcome	The application classified a schematic representation of a human face drawn with a pen on a checkered sheet of paper as a human agent only when it was in a position of a rear passenger. When no human agent was visible in the camera lens, the schematic drawing was not classified as human, but when at least one human agent was in the camera facial features landmarks appeared on the schematic face image. NOTE: Since the application does not stop responding to the behavior of the human agent this is not considered a serious flaw in the application's performance, but it is not known how the application would react to a more accurate representation of the human face and apparel.

Table 3.14. Description of a use case test: face recognition with *N = 0.*

#	13	Test case name	Face recognition of 0 faces
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking the reaction of the application to the lack of any human agent in the view.	

Description	The application got simply turned on without a human agent in view; the application was turned on with a human agent in view, which then got out of view.
Steps	1. Starting session without human agent in view 2. Checking if there are any notifications/ alarms 3. Ending the session 4. Starting session with human agent in view 5. Human agent retreats from the view 6. Checking if there are any notifications/ alarms 7. Ending the session
Assessment criteria	Checking if the application would react to the lack or retreat of a human agent from the view (success or failure in lack of a human agent detection).
Outcome	When the application is turned on and there is no visible human being in the camera lens, the app simply does not respond. The same lack of reaction occurs when the human agent retreats from view. This is interpreted as the correct response to the absence of a human agent.

Table 3.15. Description of a use case test: subject's eyes closing.

#	14	Test case name	Subject's eyes closed/open
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when eye closure or opening occurs in a controlled environment without extra distractions or movements.	

Description	Human agent performed the action of closing and opening their eyes several times.
Steps	1. Starting the session when the human agent has their eyes closed 2. Human agent performs eyes half-closing and closing 3. Ending session 4. Starting the session when the human agent has their eyes opened 5. Human agent performs eyes opening 6. Human agent performs eyes half-closing and closing 7. Ending session
Assessment criteria	Success or failure in detecting the occurrence of a tested event (closed/ opened eyes) and giving correct feedback about eye-closure values.
Outcome	The application in a controlled environment detects that the eyes are open/ closed, but the reported values for eye-closure may not be infallible or even very accurate.

Table 3.16. Description of a use case test: yawning.

#	15	Test case name	Yawning
Hardware		Samsung A13, serial: R58T503KNFN	
System version		Android 12, One UI Core 4.1	
Purpose		Checking how the application will perform when yawning occurs in the controlled environment without extra distractions or movements.	

Description	Human agent performed the action of yawning several times with different levels of eye closure.
Steps	1. Starting the session 2. Human agent performs yawning several times with different eyes openings 3. Ending session
Assessment criteria	Success or failure in detecting the occurrence of a tested event (yawning).
Outcome	The application detects yawning almost flawlessly in a controlled environment. Some flickering of the 'Wake up' alarm was noticed and is presented in the recording *TestCase_14_Yawning*.