D4.5 – CPSOSAWARE AI FRAMEWORK AND MODEL BASED DESIGN

Authors	IBM, ISI, CTL, UoP
Work Package	WP4 – CPSoSaware System Layer Design and adaptation of dependable CP(H)SoS

Abstract

This document contains joint output of Task 4.5 "Cognitive System AI-assisted maintenance and CPSoS lifecycle Design Continuum Support" and Task 4.6 "Model-based Design and Redesign of CPSoS Functional blocks Realization" and describes the CPSoSaware AI framework and model-based design and re-design methodology. In the context of the AI framework, we performed multimodal odometer fusion evaluation studies and presented a vertical AI mechanism based on a data integration framework with application to automotive and manufacturing use-cases. Presented model-based design and re-design methodology validated on the model of driver state monitoring system.

Funded by the Horizon 2020 Framework Programme
of the European Union

Deliverable Information

Work Package	WP4 – CPSoSaware System Layer Design and adaptation of dependable CP(H)SoS
Task	T4.5 – Cognitive System AI-assisted maintenance and CPSoS lifecycle Design Continuum Support T4.6 - Model-based Design and Redesign of CPSoS Functional blocks Realization
Deliverable title	CPSOSAWARE AI FRAMEWORK AND MODEL BASED DESIGN
Dissemination Level	Public
Status	F: Final
Version Number	1.00
Due date	31/08/2022

Project Information

Project start and duration	01/01/2020 – 31/12/2022, 36 months
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	04/08/2022	*Document outline created*	IBM
0.2	11/08/2022	*CTL and ISI Contributions received*	CTL, ISI
0.3	18/08/2022	*CTL and ISI Contributions integrated*	IBM
0.4	25/08/2022	*IBM and UoP Contribution integrated*	IBM
0.5	01/09/2022	*Introduction added*	IBM
0.0		*Document Lost Due to computer failure*	
0.6	26/10/2022	*ToC and Introduction restored*	IBM
0.61	01/11/2022	*CTL and ISI Contributions integrated*	IBM
0.62	03/11/2022	*IBM and UoP Contribution restored*	IBM
0.63	06/11/2022	*ISI Contribution reviewed and errors fixed*	ISI, IBM
0.7	07/11/2022	*IBM and UoP Contribution integrated*	IBM
0.9	08/11/2022	*Document finalized*	IBM
1.0	09/11/2022	*Document reviewed and sent to coordinator*	IBM

	NAME
Prepared by	IBM
Reviewed by	-
Authorised by	IBM

DATE	RECIPIENT
08/11/2022	Project Consortium
10/11/2022	European Commission

Table of Contents

1 Executive Summary ... 6
1.1 Document structure ... 6
1.2 Definitions and Acronyms ... 7
2 AI Framework ... 9
2.1 Multimodal odometer fusion evaluation studies ... 9
2.1.1 Relocalization scheme ... 12
2.1.2 Implementation using CARLA-ROS framework ... 14
2.1.3 Experimental setup in CARLA-ROS and results ... 14
2.1.4 Evaluation studies with model compression acceleration ... 19
2.2 Semantic Data integration ... 27
2.2.1 Automotive Pillar ... 29
2.2.2 Manufacturing Pillar ... 35
3 Model-Based design and re-design ... 43
3.1 Model-based design and re-design methodology ... 43
3.2 DSM model ... 44
3.2.1 ERT models developed ... 45
3.2.2 Accuracy of the models ... 45
3.2.3 Latency and power consumption ... 46
3.3 Evaluation of model-based design and re-design methodology ... 47
3.3.1 System model and input data ... 47
3.3.2 Optimization model ... 50
3.3.3 Results ... 52
4 Conclusions ... 54

List of figures

Figure 1: ORB-SLAM architecture 10
Figure 2: LeGO-LOAM architecture 11
Figure 3: Multimodal re-localization scheme 13
Figure 4 Overview of the Carla ROS setup 14
Figure 5: Simulated test case 1: Normal weather conditions 15
Figure 6: Simulated test case 2: Hard rain at noon 16
Figure 7: Simulated test case 3: Hard rain at noon 17
Figure 8: Simulated test case 4: Medium rain at noon 18
Figure 9: Simulated test case 5: Medium rain at noon 19
Figure 10: SqueezeDet architecture. 20
Figure 11: Pointpillars network overview 20
Figure 12: PointPillars BEV Average Precision. The shown acceleration ratios of α = 10, 20, 30, and 40 on the targeted layers, correspond to a total acceleration of PointPillars by 5.6×, 7.6×, 8.6×, and 9.2×, respectively 21
Figure 13: PV-RCNN BEV Average Precision. The presented acceleration ratios of α = 10, 20, 30, and 40 on the targeted layers, correspond to a total acceleration of the BEV-Backbone block by 4.5×, 5.5×, 6.0×, and 6.3×, respectively 22
Figure 14: KITTI dataset examples. 22
Figure 15: Fusion pipeline 25
Figure 16: NMS algorithm 25
Figure 17: Detection example in highway scene 27
Figure 18: Overview of the process of semantic data integration. 28
Figure 19: Overview of the system architecture for the automotive pillar 29
Figure 20: Excerpt of the ATE and RPE observations 30
Figure 21: Representation of the sample observations in the KG 34
Figure 22: Performance comparison of LeGO vs DSO for a simulation session. 34
Figure 23: Graph indicating the risk levels during a driving session. 35
Figure 24: Workflow overview. 36
Figure 25: Simulated environment snapshot. 37
Figure 26: Core semantic model for the Manufacturing Use Case 38
Figure 27: Sample instantiations 38
Figure 28: HW-SW partitioning optimization approach 43
Figure 29: Concise model of DSM application - design 48
Figure 30: Concise model of DSM application – scenario table 49
Figure 31: Concise model of DSM application – hardware kernels catalog 49
Figure 32: Concise model of DSM application – optimization goals 50

List of tables

Table 1: Algorithm: Adaptive selection of involved landmarks 13
Table 2: KITTI difficulty levels 26
Table 3: Evaluation study results 26
Table 4: Set of rules for calculating the risk level. 35
Table 5: ERT parameters customized for the DSM application 44
Table 6: ERT models used, based on different ERT parameters 45
Table 7: Top-3 models with the highest accuracy in yawning measurement. 46
Table 8: Dynamic power consumption of the HW kernels of the models referenced in Table 7. 46
Table 9: Static power consumption of the HW kernels of the models referenced in Table 7 47
Table 10: Total power dissipation of the PL part 47
Table 11: Estimation of the single frame processing latency 47
Table 12: Normalized latency and power consumption 50
Table 13: Optimization results - Male-Dash Scenario 52
Table 14: Optimization results - Male-Mirror Scenario 52
Table 15: Optimization results - Female-Dash Scenario 52
Table 16: Optimization results - Female-Mirror Scenario 53
Table 17: Optimization results - Night-time Scenario 53

1 Executive Summary

This document contains the joint output of Task 4.5 "Cognitive System AI-assisted maintenance and CPSoS lifecycle Design Continuum Support" and Task 4.6 "Model-based Design and Redesign of CPSoS Functional blocks Realization". Task 4.5 deals with the coordination of adaptation mechanisms within and across layers to ensure system-wide consistency and the incorporation of emerging machine learning techniques to address challenging situations. In the context of this task, we developed vertical AI schemes for cooperative fusion of different sensor agents. Sensor cooperation is necessary in order to maximize overall performance of vehicles under various environmental conditions, where particular sensors may fail or provide inaccurate data. To test our cooperative sensor fusion approach, we performed multimodal odometer fusion evaluation studies that based on realizes sensor cooperation approach that is important for improving two major tasks of autonomous driving: vehicle odometry or SLAM and scene analysis and understanding. Furthermore, we developed a vertical fusion strategy, which integrates the data from "selfish nodes", i.e., outputs of individual sensors into a combined estimation framework, and provides more accurate pose information as well as object detection. The vertical fusion of selfish sensor agents is extended by a vertical AI mechanism based on a novel semantic data integration framework for monitoring and safeguarding the ergonomics of human operators during a collaborative assembly task in an automotive manufacturing environment. This mechanism is a part of CPSoSaware Cognitive System AI Engine (CSAIE) component that analyses outputs of cooperative sensor measurements and feeds them to the ontology-based semantic Knowledge Graph (KG) (which defines the set of incentives) through the flexible semantic data integration framework. CSAIE component's primary purpose is to provide cognitive control and management mechanism of the CPSoS functional and non-functional goals as those are captured through the requirement KPIs of the system. The CSAIE collects inputs of different sensors and performs a broad analysis on the sensors data in combination with simulated training datasets from the SAT block using an AI engine. These allow us to collect data on functional and non-functional KPIs of different modules and parts of CPSoS under different environmental conditions, in order to identify the set of optimal model-based designs, where each particular design provides an optimal combination of values for the specific set of KPIs under specific environmental condition. The identification of such designs is performed in Task 4.6.

The main purpose of Task 4.6 is to develop a model-based design and re-design component which selects models/designs that are optimal with respect to different objectives under different environmental conditions and triggers re-design with respect to the most important situational objectives of the current environmental conditions. The choice of optimal models/designs introduces a trade-off between different objectives. To handle this trade-off, we considered a multi-objective multi-scenario optimization approach that produces a set of optimal designs. Each design corresponds to one or several scenarios and/or to one or several combinations of situational goals. This allows triggering the re-design process when environmental conditions change, ensuring that at that target SoS always has a design that is optimized with respect to situational goals. The re-design phase itself utilizes commissioning and decommissioning mechanism that developed in the scope of Tasks 4.2/4.3 and described in detail in the corresponding deliverables. In the context of Task 4.6 we describe model-based design and re-design methodology and present evaluation of this methodology to the driver monitoring system model (DSM).

1.1 Document structure

This document is structured into eight major sections:

- **Section 1** introduces the document, outlining its structure, and identifying terms and acronyms used across the document.
- **Section 2** describes CPSoSaware AI framework and shows multimodal odometer fusion evaluation study and presents vertical AI mechanism based on a data integration framework with application to automotive and manufacturing use-cases.
- **Section 3** presents CPSoSaware model-based design and re-design methodology and describes evaluation of this methodology on driver monitoring system model.

- **Section 4** concludes the document.

1.2 Definitions and Acronyms

Below are listed the most relevant acronyms used in the document and recurring definitions:

Acronym / Term	Definition
AI	Artifical Intelengences
AOS	Average Orientation Similarity
ATE	Absolute Trajectory Error
BEV	Bird Eye View
BRAM	Block Random Access Memory
CASPAR	CTL's proprietary semantic data integration framework
CL	Cooperative Localization
CPS	Cyber-Physical System
CPSoS	Cyber-Physical System of Systems
CR	Confidence Rate
CSAIE	Cognitive System AI Engine
DEST	Deformable Shape Tracking
DMS	Driver Monitoring System
DoA	Grant Agreement No. 871738 – CPSoSAware. Annex 1 Description of the Action
DSM	Driver Monitoring System
DSO	Direct Sparse Odometry
DSP	Digital Signal Processor
EAR	Eye Aspect Ratio
ERT	Ensemble of Regression Trees
HLS	High Level Synthesis
HW	Hardware
ILP	Integer Linear Programming
IMU	Inertial Measurement Unit
IOU	Intersection Over Union
JSON	JavaScript Object Notation
KF	Kalman Filter
KG	Knowledge Graph
KITTI	The KITTI Vision Benchmark Suite A project of Karlsruhe Institute of Technology and Toyota Technological Institute at Chicago
KPI	Key Performance Indicator
LeGO-LOAM	Light Weight and Ground Optimization LIDAR Odometry and Mapping
LIDAR	Light Detection and Ranging
LOD	Linked Open Data

MAC	Multiply ACcumulate
MAP	Maximum A Posteriori
mAP	Mean Average Precision
MMCM	Mixed-Mode Clock Manager
NMS	Non-Maximum Suppression
OF	Occupancy Factor
OFE	Occupancy Factor Estimation
OPL	Optimization Programming Language
PERCLOS	Percentage of Eye CloSure
PL	Programmable Logic
PS	Processing System
RAM	Random Access Memory
RDF	Resource Description Framework
RMSE	Root Mean Square Error
ROS	Robot Operating System
RPE	Relative Pose Error
RULA	Rapid Upper Limb Assessment
SAT	Simulation and Training
SGD	Stochastic Gradient Descent
SLAM	Simultaneous Localization And Mapping
SPARQL	Standard Query Language and Protocol for Linked Open Data
SySML	Systems Modeling Language
W3C	The World Wide Web Consortium
XRT	Xilinx Runtime Library

2 AI Framework

2.1 Multimodal odometer fusion evaluation studies

This Section is dedicated to the presentation of **vertical AI schemes for cooperative fusion** of different sensor agents, i.e., LIDAR and camera, for improving namely two major tasks of autonomous driving: vehicle odometry or SLAM and scene analysis and understanding. SLAM approaches rely on camera or LIDAR sensor, which are susceptible to failure in harsh conditions (e.g., extreme weather, featureless areas, sharp turns, etc.). **Sensor cooperation** is therefore necessary in order to **maximize overall performance of vehicle** and will be realized through Graph Laplacian Processing fusion technique. The same limitations are also apparent when either of the two sensors is used for scene analysis and understanding, thus LIDAR and image data fusion becomes imperative. For this case, a geometric approach is introduced which fuses the output of the two sensors from 3D-to-2D projection. More specifically, a **vertical fusion strategy** has been developed for each one of the associated tasks, which integrates in a combined estimation framework the data from **"selfish nodes"**, i.e., outputs of individual sensors, and provides more accurate pose information as well as object detection. In following, we will shortly review the developed multi-modal re-localization and scene analysis and understanding schemes. Afterwards, we will describe the details of its deployment in the CARLA-ROS framework.

For the purposes of this deliverable, three state-of-the-art SLAM solutions, i.e., Direct Sparse Odometry (DSO)[1], ORB-SLAM[2], and Light Weight and Ground Optimization LIDAR Odometry and Mapping (LeGO-LOAM)[3], have been integrated to CARLA-ROS framework and have been previously analyzed in D3.1

Those three algorithms enabled the design and development of the proposed multimodal re-localization odometry scheme. The details are discussed below:

DSO is a visual odometry method based on a novel, highly accurate sparse and direct structure and motion formulation. It exploits a probabilistic model (minimizing a photometric error) with consistent, joint optimization of all model parameters, including geometry-represented as inverse depth in a reference frame-and camera motion. This probabilistic model takes noisy measurements Y as input and computes an estimator X for the unknown, hidden model parameters (3D world model and camera motion), following a Maximum A Posteriori (MAP) approach. Due to the direct formulation of DSO, it directly uses the actual sensor values-light received from a certain direction over a certain time period-as measurements Y in the probabilistic model. Additionally, one of the main benefits of a direct formulation is that it does not require a point to be recognizable by itself, thereby allowing for a more finely grained geometry representation (pixelwise inverse depth). Furthermore, data from across the image can be sampled—including edges and weak intensity variations generating a more complete model and lending more robustness in sparsely textured environments. A sparse framework (these methods use and reconstruct only a selected set of independent points, traditionally corners) has been chosen during the optimization since the main drawback of adding a geometry prior, as dense methods do, is the

[1] J. Engel, V. Koltun and D. Cremers, "Direct Sparse Odometry," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 40, no. 3, pp. 611-625, 1 March 2018, doi: 10.1109/TPAMI.2017.2658577.

[2] R. Mur-Artal, J. M. M. Montiel and J. D. Tardós, "ORB-SLAM: A Versatile and Accurate Monocular SLAM System," in IEEE Transactions on Robotics, vol. 31, no. 5, pp. 1147-1163, Oct. 2015, doi: 10.1109/TRO.2015.2463671.

[3] T. Shan and B. Englot, "LeGO-LOAM: Lightweight and Ground-Optimized Lidar Odometry and Mapping on Variable Terrain," 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), 2018, pp. 4758-4765, doi: 10.1109/IROS.2018.8594299.

introduction of correlations between geometry parameters, which render a statistically consistent, joint optimization in real time infeasible. Backend optimization is performed in a sliding window, exploiting Gauss-Newton algorithm, where old camera poses as well as points that leave the field of view of the camera are marginalized. In contrast to existing approaches, this method further takes full advantage of photometric camera calibration, including lens attenuation, gamma correction, and known exposure times. This integrated photometric calibration further increases accuracy and robustness. DSO, apart from a geometric camera model which comprises the function that projects a 3D point onto the 2D image, considers also a photometric camera model, which comprises the function that maps real-world energy received by a pixel on the sensor (irradiance) to the respective intensity value. Two additionally important modules of DSO are Frame and Point Management. In terms of Frame Management, a window of up to N_f active keyframes ($N_f = 7$ is used) must be kept. Every new frame is initially tracked with respect to these reference frames (Step 1). It is then either discarded or used to create a new keyframe (Step 2). Once a new keyframe-and respective new point are created, the total photometric error is optimized. Afterwards, one or more frames are marginalized (Step 3). As far as the Point Management is concerned, a fixed number N_p of active points (we use $N_p = 2000$), equally distributed across space and active frames, is used in the optimization. As a first step, N_p candidate points are identified in each new keyframe (Step 1). Candidate points are not immediately added into the optimization, but instead are tracked individually in subsequent frames, generating a coarse depth value which will serve as initialization (Step 2). When new points need to be added to the optimization, a number of candidate points (from across all frames in the optimization window) is chosen to be activated, i.e., added into the optimization (Step 3). Note that DSO only keeps N_p active points across all active frames combined.

ORB-SLAM, is a feature-based monocular SLAM system that operates in real time, in small and large indoor and outdoor environments. The system is robust to severe motion clutter, allows wide baseline loop closing and relocalization, and includes fully automatic initialization. Building on excellent algorithms of recent years, a novel system that uses the same features for all SLAM tasks is designed: tracking, mapping, relocalization, and loop closing. ORB-SLAM system overview is shown in Figure 1. ORB-SLAM extends the mapping capability to large scale spaces by dividing the mapping thread into a local mapping thread and a global mapping thread. The local mapping thread maintains a set of keyframes and landmarks which the tracking thread is used for camera pose estimation. However, in ORB-SLAM the local map is bounded in its complexity by only including keyframes that visually overlap with the current keyframe. During the tracking process correspondences between the input frame and the local map are computed. As keyframes are added to the local map, starting with the correspondences detected by the tracking thread, a further correspondence search is carried out to compute the level of visual overlap (i.e., shared features) between other keyframes in the local map. This information is stored in a co-visibility graph, where edges are added between camera nodes that share features. The weight of an edge corresponds to the number of shared features between its associated cameras. During operation the local mapping thread only considers keyframes linked to the current reference frame via edges above a threshold weight. As the camera explores, new keyframes are added liberally to the local map to ensure

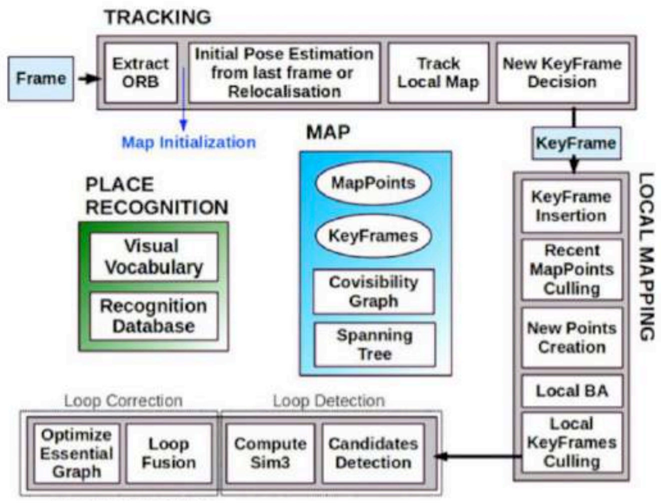


Figure 1: ORB-SLAM architecture

robust tracking, however in order to ensure complexity of the local mapping is bounded, keyframe and landmark culling strategies are also employed. Taken together, the co-visibility graph and the culling strategies decouple the complexity of the local mapping process from the scale of the global map. ORB-SLAM's large scale mapping capability is provided by a third thread. As each new keyframe is added to the co-visibility graph, the strongest associated edge is also added to a second essential graph. The strongest edge connects the new keyframe to the keyframe with the highest number of shared features. The essential graph therefore contains all of the keyframes from the co-visibility graph, but only a subset of the edges that form a spanning tree from the first keyframe, and any edges that are added due to loop closures. Therefore, as new frames are input from the camera, they are first processed by the tracking thread which execute the following stages: i) ORB feature extraction, ii) Pose estimation and relocalization, iii) Local map tracking, and iv) Keyframe generation. As new keyframes are inserted the local mapping thread culls map and keyframes, adds new map points, and performs local bundle adjustment on the local map. In more detail, it executes the following steps: i) Keyframe insertion, ii) Map-point culling, iii) New point creation, iv) Local Bundle Adjustment, v) Keyframe culling. Finally, as new keyframes are generated the loop closing thread attempts to detect loop closures as follows: i) Place recognition & loop detection, ii) 7DOF (translation, rotation and scale) constraint estimation, iii) Loop fusion and iv) Pose graph optimization.

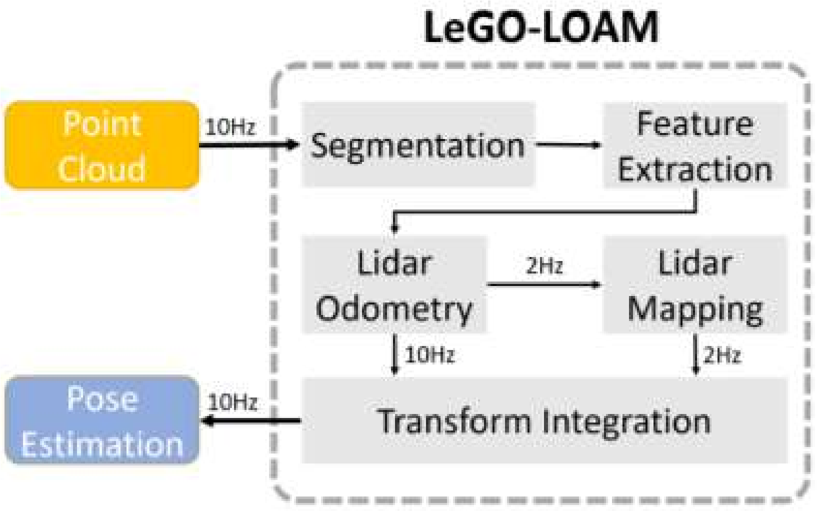


Figure 2: LeGO-LOAM architecture

LeGO-LOAM is a LIDAR odometry solution for pose estimation in complex environments with variable terrain. LeGO-LOAM is lightweight, as real-time pose estimation and mapping can be achieved on an embedded system. Point cloud segmentation is performed to discard points that may represent unreliable features after ground separation. LeGO-LOAM is also ground-optimized, as a two-step optimization for pose estimation is introduced. Planar features extracted from the ground are used to obtain z translation, roll and pitch during the first step. In the second step, the rest of the transformation (x, y translation and yaw) is obtained by matching edge features extracted from the segmented point cloud. The ability to perform loop closures to correct motion estimation drift is also introduced. Lego-LOAM system overview is shown in Figure 2. The overall system is divided into five modules. The first, segmentation, takes a single scan's point cloud and projects it onto a range image for segmentation. The segmented point cloud is then sent to the feature extraction module, which determines two types of features: edge and planar. Then, LIDAR Odometry uses features extracted from the previous module to find the transformation relating consecutive scans using the two-step Levenberg-Marquardt optimization. The features are further processed in LIDAR mapping, which registers them to a global point cloud map. At last, the transform integration module fuses the pose estimation results from lidar odometry and LIDAR mapping and outputs the final pose estimate. The proposed system seeks improved efficiency and accuracy for ground vehicles, with respect to the original, generalized LOAM framework. LeGO-LOAM doesn't perform any backend optimization as DSO does.

2.1.1 Relocalization scheme

The overview of the proposed multimodal re-localization scheme is shown in Figure 3. At the core of our approach lies the concept of coupling the pose from one modality (e.g., visual) with the landmarks of the other modality (e.g., LIDAR) through Graph Laplacian Processing technique, a graph signal processing tool. The intuition behind our motive was that by effectively combining poses and landmarks from different modalities, referring however 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 (x, y position and yaw angle) 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. We employ a technique[4] of fusing the two estimated poses in an optimal manner and avoiding any handcrafted weights.

Assume two groups of Gaussians means and covariances: (x_1, Σ_1), (x_2, Σ_2). Compute initially matrices:

$$P_1 = \Sigma_1^{-1}(\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$$

$$P_2 = \Sigma_2^{-1}(\Sigma_1^{-1} + \Sigma_2^{-1})^{-1}$$

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 (mean and covariance) of the two KFs. The motion model utilized by each KF is described by the following equations:

$$x^t = x^{t-1} + \tilde{u}_x^t \Delta t$$

$$y^t = y^{t-1} + \tilde{u}_y^t \Delta t$$

$$\theta^t = \theta^{t-1} + \widetilde{\omega}^t \Delta t$$

Note that $[x^{t-1}, y^{t-1}, \theta^{t-1}]$ correspond to the previous estimations of individual KF, not the outputs of fusion level. It is assumed that control parameters of linear velocities $\tilde{u}_x^t, \tilde{u}_y^t$ and yaw rate $\widetilde{\omega}^t$ are given by the Inertial Measurement Unit (IMU). In addition, all three motion parameters are degraded by Gaussian measurement noise:

$$\tilde{u}_x^t = u_x^t + n_{u_x}, n_{u_x} \sim \mathcal{N}(0, \alpha u_x^t)$$

$$\tilde{u}_y^t = u_y^t + n_{u_y}, n_{u_y} \sim \mathcal{N}\left(0, \alpha u_y^t\right)$$

$$\widetilde{\omega}^t = \omega^t + n_\omega, \qquad n_\omega \sim \mathcal{N}(0, \beta\ degrees\ per\ hour),$$

[4] P. Yang, D. Duan, C. Chen, X. Cheng and L. Yang, "Multi-Sensor Multi-Vehicle (MSMV) Localization and Mobility Tracking for Autonomous Driving," IEEE Transactions on Vehicular Technology, vol. 69, p. 14355–14364, 12 2020

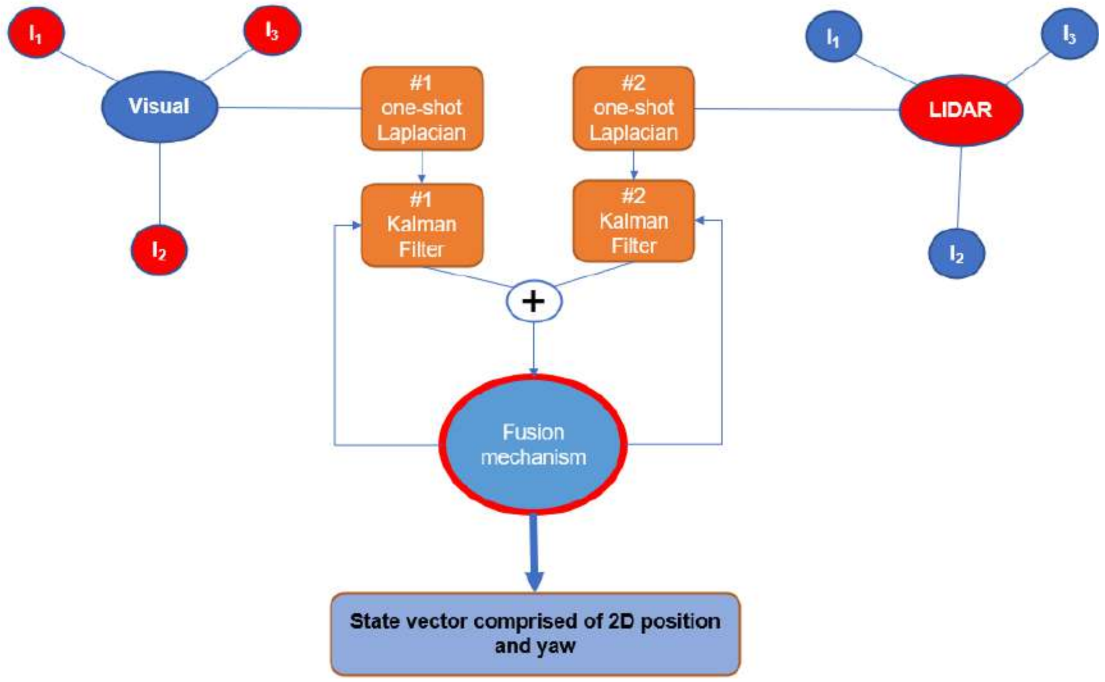


Figure 3: Multimodal re-localization scheme

where u_x^t, u_y^t, ω^t are the actual velocities and yaw rate. Velocity noise is modelled so as to simulate the accumulative or drift error of IMU, while yaw rate measurement noise is known as angle random walk.

In terms of KF's measurement models, the latter will be feeded by the output of fusion level. To be more specific and considering the left branch of Figure 3, assume that 2D position and yaw measured by the Laplacian Processing will be equal to $\left[x_{lapl}^t, y_{lapl}^t, \theta_{lapl}^t\right]$. The previously estimated state by the fusion mechanism is equal to $\left[x_f^{t-1}, y_f^{t-1}, \theta_f^{t-1}\right]$. The measurement vector $\boldsymbol{z^t}$ shall be comprised of a weighted version of Laplacian Processing and motion-updated fused outputs:

$$\boldsymbol{z^t} = \left[w_1 * \left(x_f^{t-1} + u_x^t \Delta t\right) + w_2 * x_{lapl}^t, w_1 * \left(y_f^{t-1} + u_y^t \Delta t\right) + w_2 * y_{lapl}^t,\ w_3 * \left(\theta_f^{t-1} + \omega^t \Delta T\right) + w_4 * \theta_{lapl}^t\right]$$

The weights w_1, w_2, w_3, w_4 are used to balance between the motion-updated fusion and Graph Laplacian Processing.

Furthermore, instead of using the whole number of landmarks detected by visual or LIDAR SLAM, we choose to discard those who are far away than the corresponding pose. To do so, we measure all the relevant distances between individual landmarks and pose. Afterwards, we construct the histogram of the distances and choose the first n centroids out of the total bins of the histogram. Then we take the sum of these n centroids and discard the distances which exceed this sum. Finally, the distance threshold is determined by taking the average distance of the remaining landmarks. The steps for this novel strategy are summarized in Table 1.

Table 1: Algorithm: Adaptive selection of involved landmarks

For every time instant t, repeat: *Step 1:* Measure distances between individual landmarks and pose

Step 2: Construct the histogram of distances and keep the centroids of first n out of total N bins

Step 3: Take the sum of n centroids and discard the landmarks whose distance from pose exceed this sum

Step 4: Set the dynamic threshold equal to the average of the remaining distances

2.1.2 Implementation using CARLA-ROS framework

The integration to the Carla-ROS framework is the development of the appropriate ROS nodes that will implement a specific algorithmic behavior. These nodes either contain all the resources necessary for executing the algorithm or the call an appropriate library. Every ROS node consumes and published data in the context of the ROS framework, under ROS topics. The type of these messages is either predefined by the ROS framework or custom types can be created and used. The synchronization of all the nodes is provided by the ROS framework. Concerning the programming languages, the nodes are either implemented in Python or in C++.

All the nodes consume data generated in the Carla simulation environment. The data can be either generated asynchronously (rosbags) or synchronously. In the latter case, the user of the Carla-ROS -bridge is necessary for the establishment of the bi-directional communication between Carla and the ROS framework.

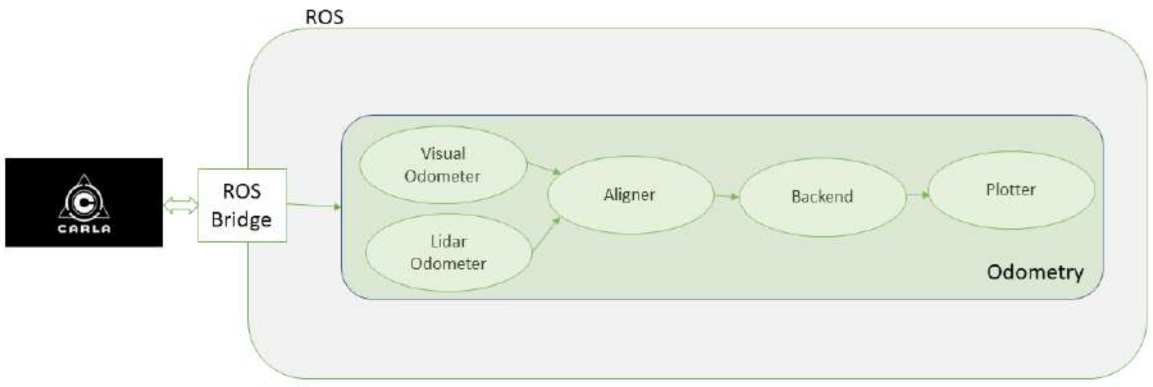
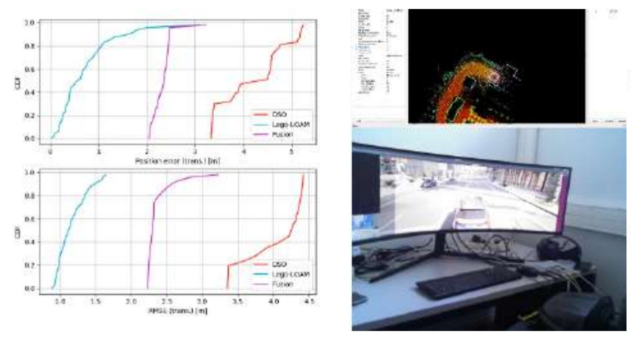


Figure 4 Overview of the Carla ROS setup

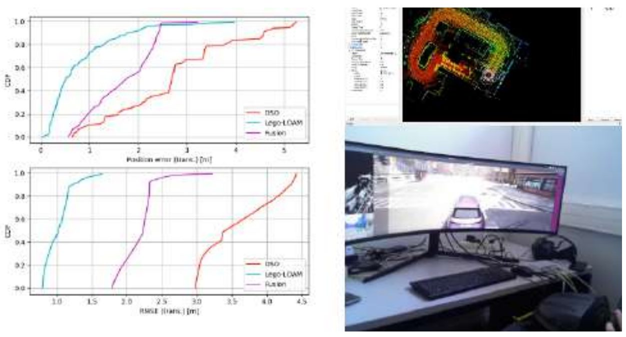
2.1.3 Experimental setup in CARLA-ROS and results

In this Section, we demonstrate the performance of the proposed scheme using the CARLA simulator and ROS framework. DSO, ORB-SLAM and LeGO-LOAM algorithms have been employed to generate the pose of vehicle and landmarks of the map. The evaluation study has been conducted considering the weather conditions of the specific testing setup, while we drive the car (autonomous pilot has been deactivated).

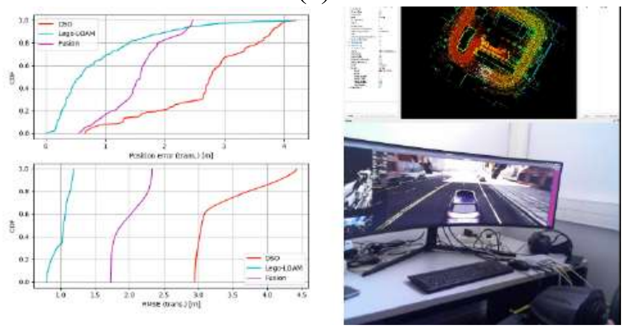
In the first simulated test case of Figure 5, the output at each one of the four time instances contains the instantaneous absolute translational error (top left), RMSE of absolute translational error over the entire trajectory up to the current time instant (bottom left), map generated by LeGO-LOAM (top right) and the testing setup (bottom right). RMSE of translational error over the entire trajectory is a more indicative metric of the performance since it demonstrates the overall ability of algorithms to provide localization information. Due to the sunny weather and normal driving style (avoiding sharp turns or sudden breakings) the proposed scheme is between LeGO-LOAM and DSO. Actually, LeGO-LOAM is shown to be superior to the other approaches. More specifically, LeGO-LOAM, proposed Fusion and DSO achieved maximum (instantaneous and overall) translational errors, respectively, equal to: 3.1m and 1.6m, 3.1m and 3.2m, and 5.2m and 4.4m in Figure 5-(a), 4m and 1.6m, 3.1m and 3.2m, and 5.2m and 4.4m in Figure 5-(b), 4.1m and 1.2m, 2.4m and 2.4m, and 4.2m and 4.4m in Figure 5-(c), and finally 3.1m and 1.6m, 3.1m and 3.2m, and 5.2m and 4.4m in Figure 5-(d). Although in some cases instantaneous position error with the proposed Fusion mechanism is smaller than LeGO-LOAM's, the latter achieved in all cases much more accurate overall positioning error, demonstrating higher localization ability. Therefore, we conclude that in normal driving conditions, LeGO-LOAM is more credible for the SLAM task.



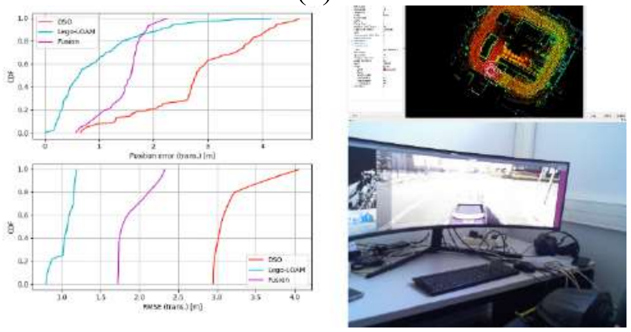
(a)



(b)



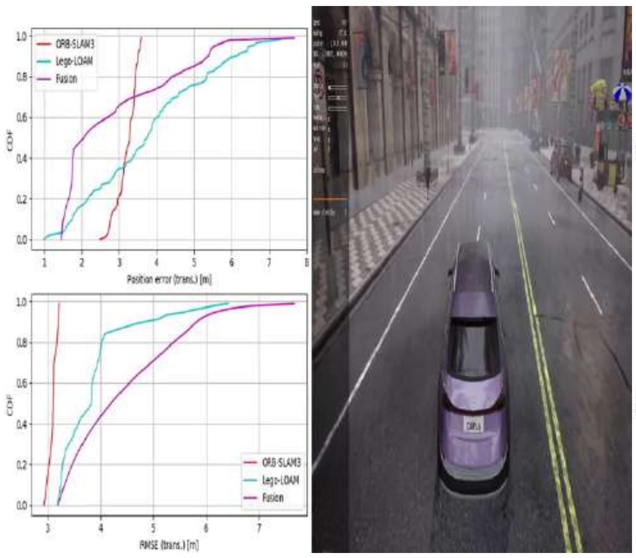
(c)



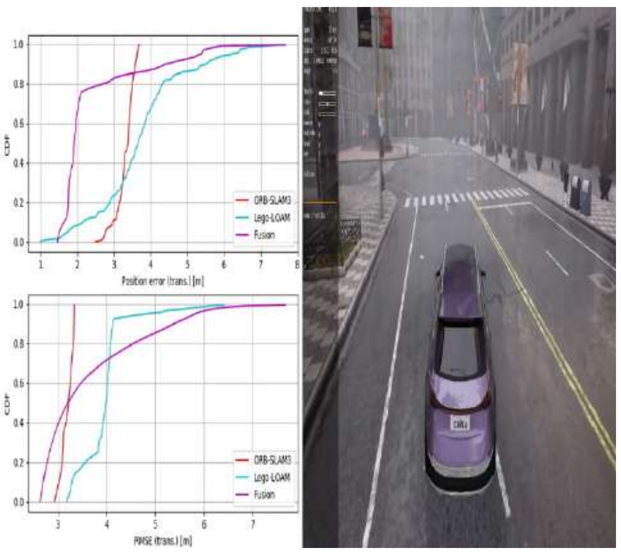
(d)

Figure 5: Simulated test case 1: Normal weather conditions

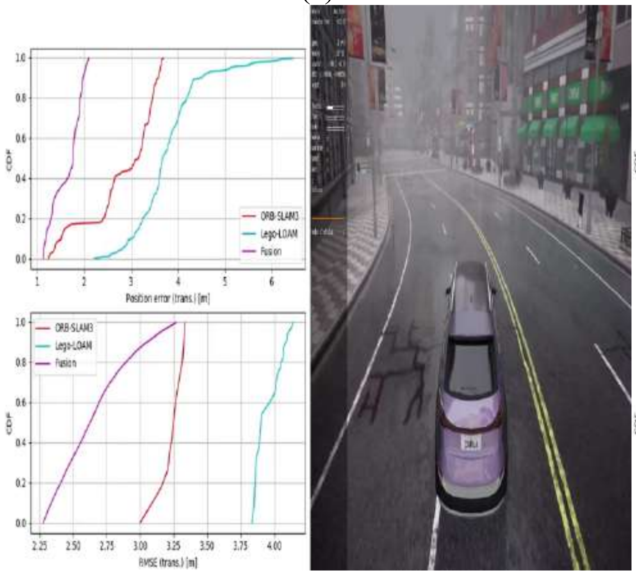
In the second simulated test case of Figure 6, we see that the performance of LeGO-LOAM has been degraded as a result of changing weather conditions to hard rain at noon, along with sharp turns during driving. Additionally, instead of DSO we employed ORB-SLAM (with ORB-SLAM3 implementation edition). More specifically, LeGO-LOAM, proposed Fusion and ORB-SLAM achieved maximum (instantaneous and overall) translational errors, respectively, equal to: 7.5m and 6.3m, 7.5m and 7.8m, and 3.5m and 3.2m in Figure 6-(a), 7.5m and 6.3m, 7.5m and 7.8m, and 3.8m and 3.3m in Figure 6-(b), 6.5m and 4.5m, 2.1m and 3.2m, and 3.7m and 3.2m in Figure 6-(c), and finally 6.1m and 4m, 2.1m and 2.2m, and 3.2m and 3.1m in Figure 6-(d). From Figure 6-(c) and (d), we see that the proposed Fusion reduced significantly not only the maximum errors, but constantly achieved lower translational errors than the other schemes. Therefore, when harsh conditions exist and performance of original SLAM algorithms is degraded, the proposed mechanism is able to significantly reduce position error.



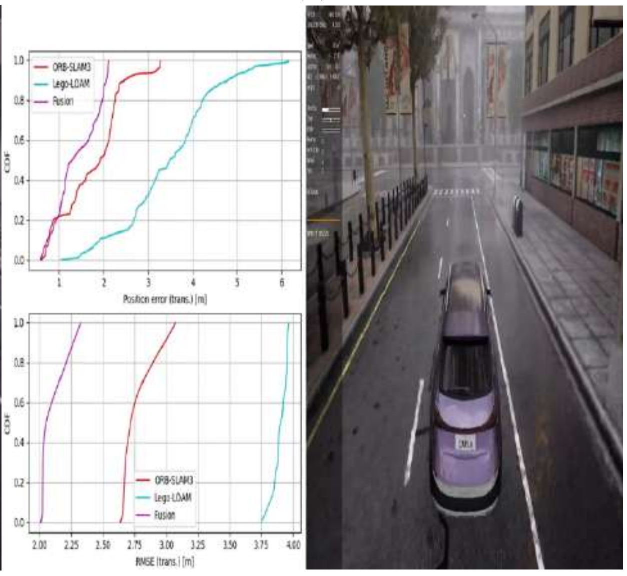
(a)



(b)



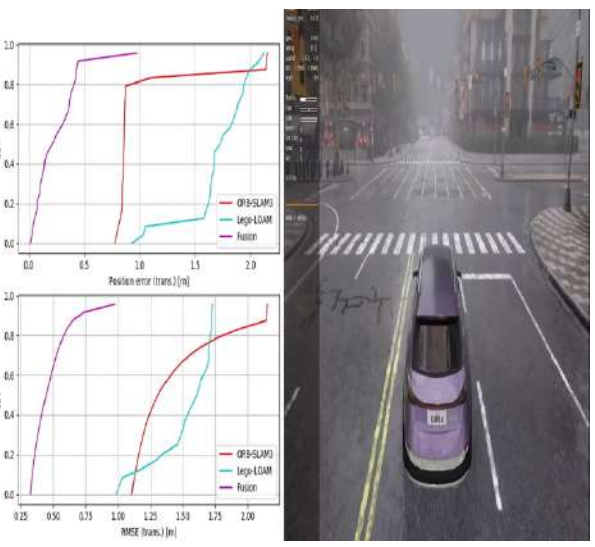
(c)

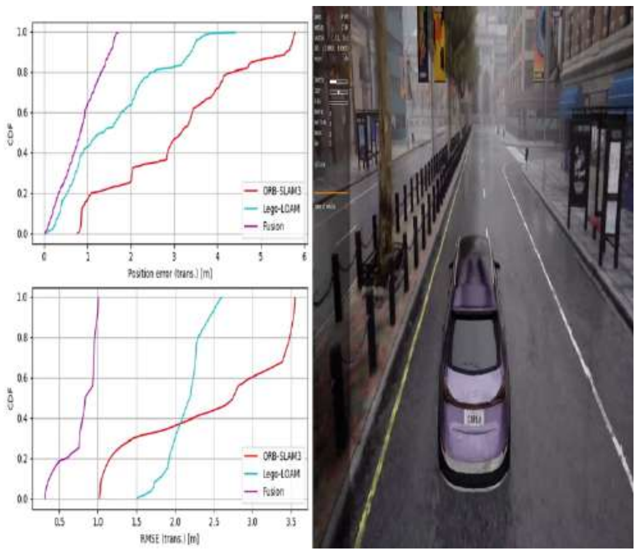


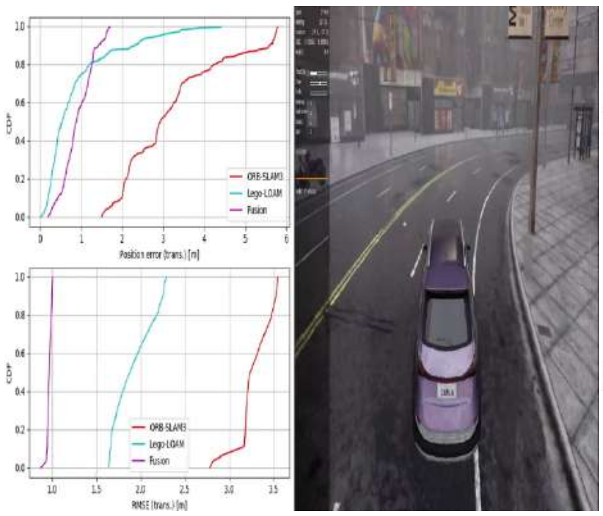
(d)

Figure 6: Simulated test case 2: Hard rain at noon

In the third simulated test case of Figure 7, we see that the performance of Fusion is now better than the other algorithms is all four cases, especially for the overall translational error of trajectory. Moreover, in Figure 7-(d), when ORB-SLAM and LeGO-LOAM are unable to produce accurate localization information due to sharp turn, proposed scheme is able to keep instantaneous error below 7.5m. More specifically, LeGO-LOAM, proposed Fusion and ORB-SLAM achieved maximum (instantaneous and overall) translational errors, respectively, equal to: 2.4m and 1.75m, 1m and 1m, and 2.4m and 2.5m in Figure 7-(a), 4.4m and 2.5m, 1.8m and 1m, and 5.8m and 3.5m in Figure 7-(b), 4.4m and 2.25m, 1.8m and 1m, and 5.8m and 3.5m in Figure 7-(c), and finally 20m and 7.5m, 6.25m and 2.5m, and 12.5m and 4.4m in Figure 6-(d). Therefore, with hard rain at noon, the quality of visual data is degraded and as a matter of fact the accuracy of SLAM algorithms is reduced. In those cases, the proposed Fusion mechanism is able to increase positioning accuracy.


(a)


(b)


(c)

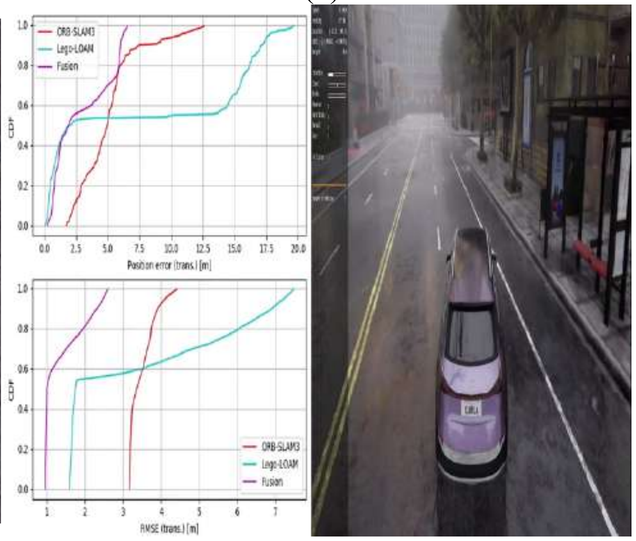
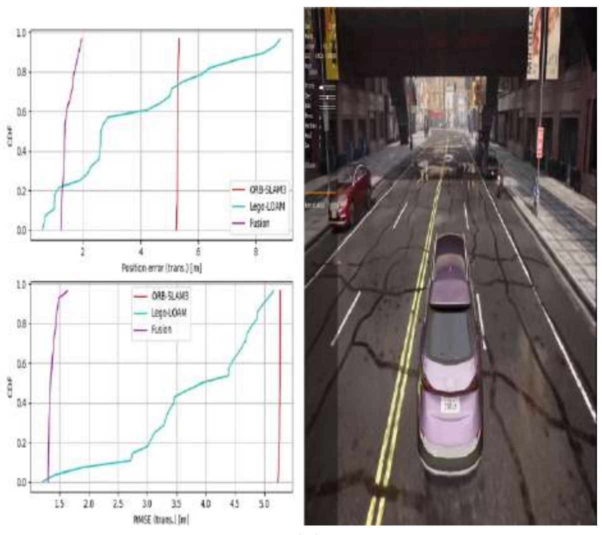
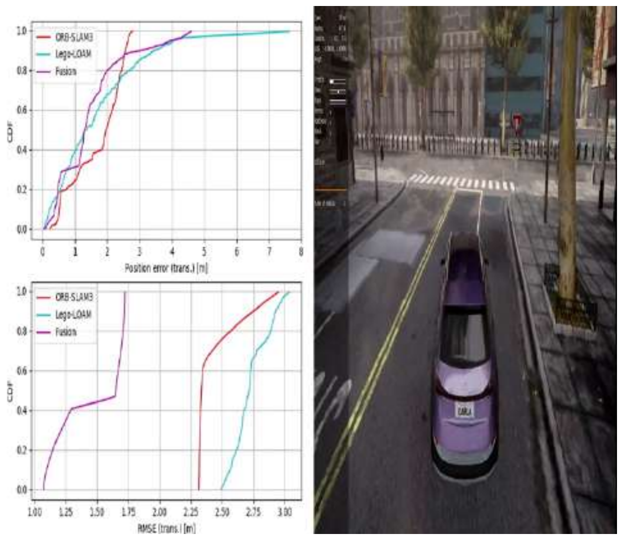

(d)

Figure 7: Simulated test case 3: Hard rain at noon

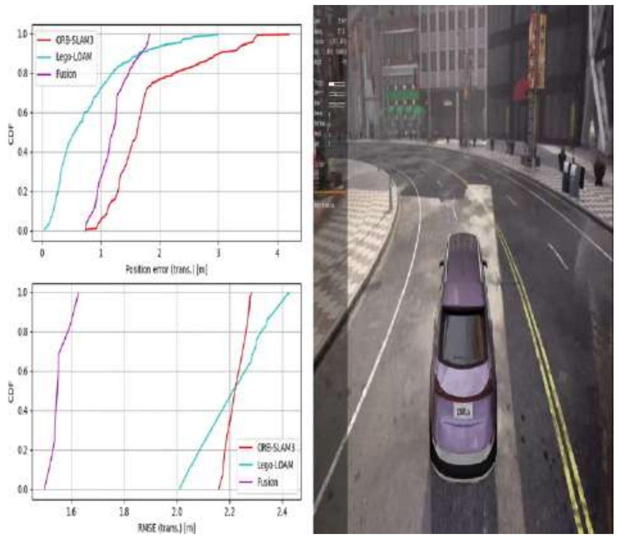
In the following scenarios, the configuration of weather has changed to medium weather at noon. Once again, we see from Figure 8 that the Fusion mechanism achieved the greatest accuracy, mainly in terms of overall translational error. More specifically, LeGO-LOAM, proposed Fusion and ORB-SLAM achieved maximum (instantaneous and overall) translational errors, respectively, equal to: 8.6m and 5.3m, 2m and 1.6m, and 5.2m and 5.4m in Figure 8-(a), 7.8m and 3m, 4.5m and 1.75m, and 2.8m and 3m in Figure 8-(b), 3m and 2.4m, 1.8m and 1.6m, and 4.2m and 2.3m in Figure 8-(c), and finally 7m and 1.8m, 2.4m and 1.8m, and 6m and 2.5m in Figure 8-(d).



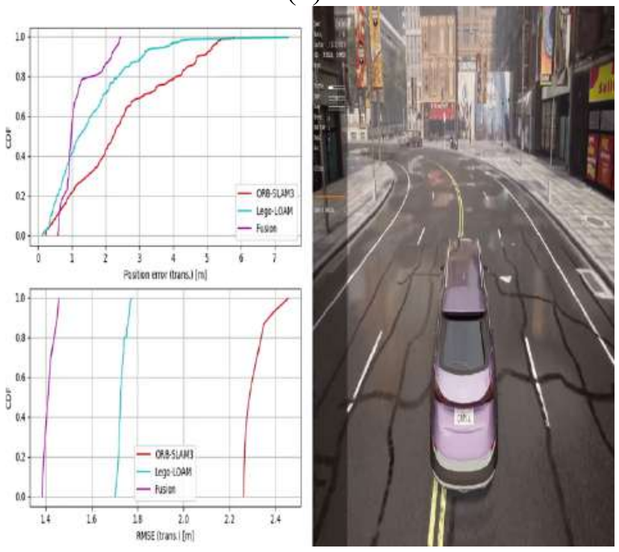
(a)



(b)



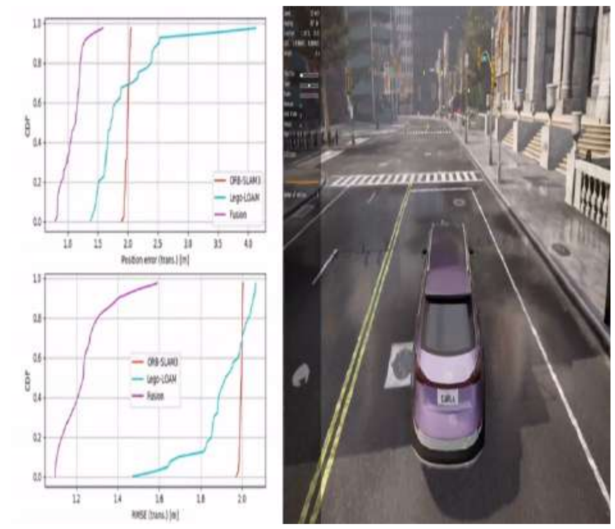
(c)



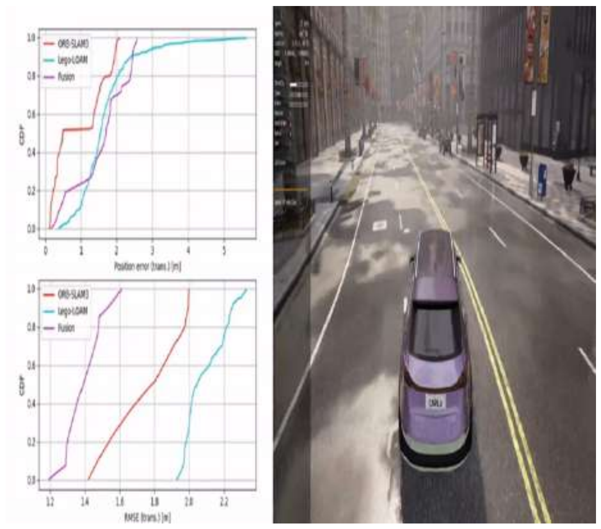
(d)

Figure 8: Simulated test case 4: Medium rain at noon

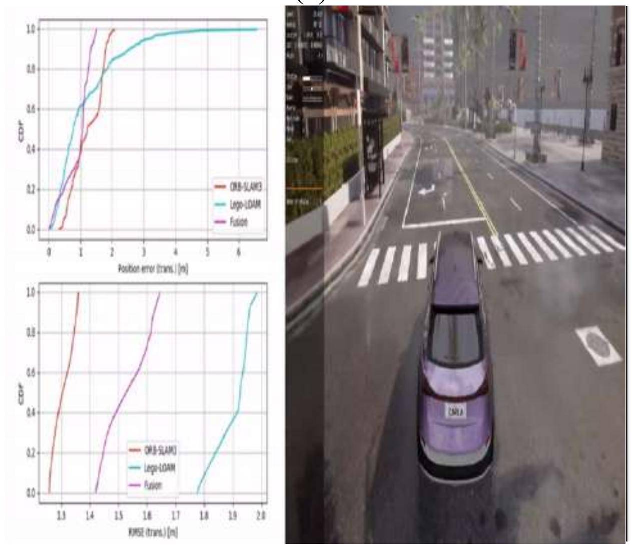
In the final testing scenario of Figure 9, the weather conditions remain as previously: medium rain at noon. We see that the performances of LeGO-LOAM and ORB-SLAM are very close, and in some cases the instantaneous position error of LeGO-LOAM exceeds that of ORB-SLAM. Fusion mechanism is able to achieve greatest accuracy in terms of overall translational error in three out of the four-time instances. More specifically, LeGO-LOAM, proposed Fusion and ORB-SLAM achieved maximum (instantaneous and overall) translational errors, respectively, equal to: 4.1m and 2.2m, 1.5m and 1.6m, and 2m and 2m in Figure 9-(a), 6m and 2.4m, 2.5m and 1.6m, and 2m and 2m in Figure 9-(b), 6.5m and 2m, 1.5m and 1.65m, and 2m and 1.35m in Figure 9-(c), and finally 4.9m and 1.7m, 1 m and 1.35m, and 3.4m and 1.6m in Figure 9-(d).



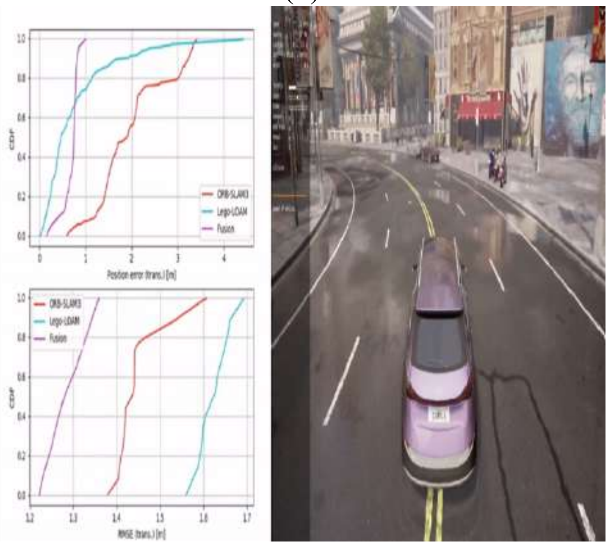
(a)



(b)



(c)



(d)

Figure 9: Simulated test case 5: Medium rain at noon

The extracted videos can be found here:

https://drive.google.com/drive/folders/10scWjwdJtUYlBf9MO09QCG0tyZvvg13m?usp=sharing

2.1.4 Evaluation studies with model compression acceleration

In order to evaluate multimodal fusion, we utilize SqueezeDet as a solution for 2D object detection and pointpillar as a solution for 3D object detection. The evaluation is also performed using model compression and acceleration techniques also presented in D3.1

SqueezeDet is a fully convolutional detection networks presented by Wu et al.[5], consisting of a feature-extraction part that extracts high dimensional feature maps for the input image, and ConvDet, a

[5] B. Wu, F. Iandola, P. H. Jin, and K. Keutzer, "SqueezeDet: Unified, Small, Low Power Fully Convolutional Neural Networks for Real-Time Object Detection for Autonomous Driving," IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Work., vol. 2017-July, pp. 446–454, 2017.

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 10 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 10, 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 still 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 with the purpose of reducing 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.

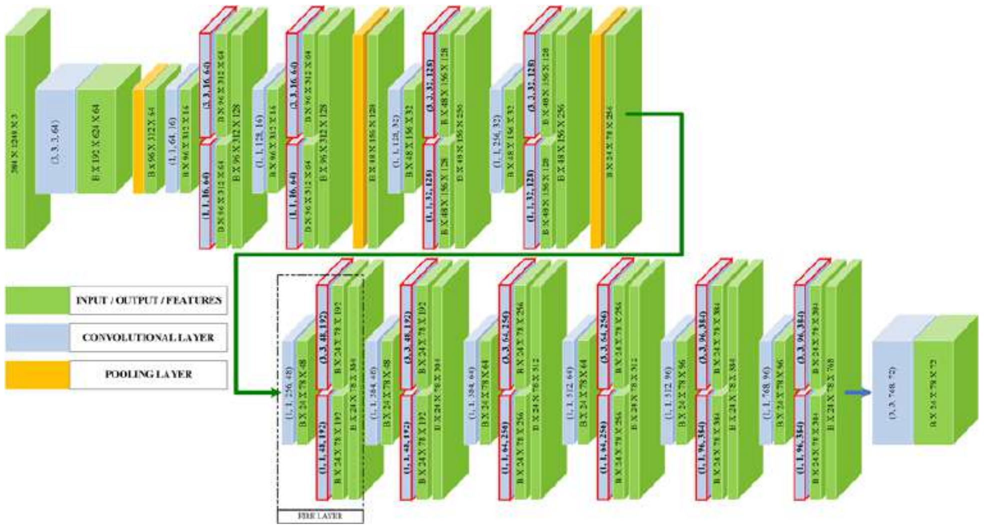


Figure 10: SqueezeDet architecture.

3D object detection from LIDAR point clouds is mainly a data-driven task due to the lack of apparent structure in the data. Pointpillars[6] proposed a novel encoder that utilizes PointNets to learn a representation of point clouds organized in vertical columns (pillars) presenting an accuracy of 74.31% in the same category.

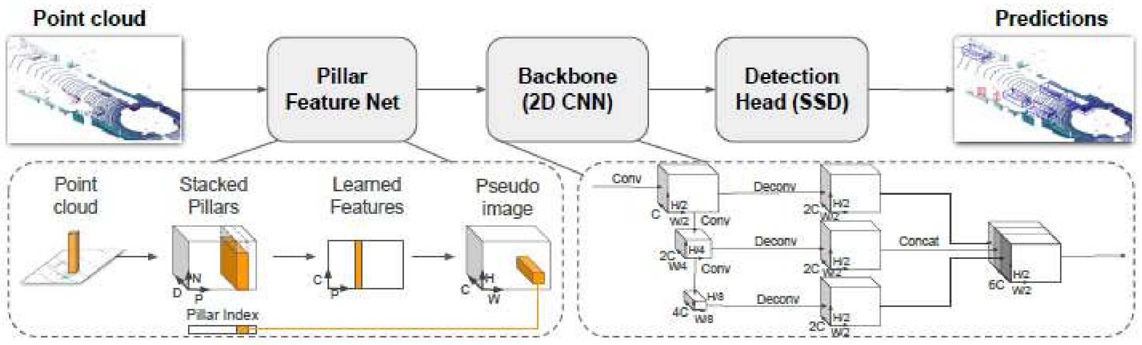


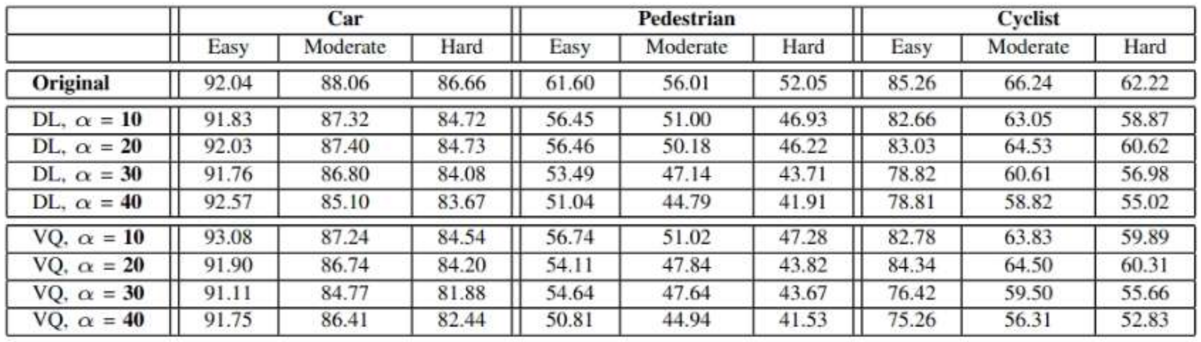
Figure 11: Pointpillars network overview

[6] A. H. Lang, S. Vora, H. Caesar, L. Zhou, J. Yang, and O. Beijbom, “Pointpillars: Fast encoders for object detection from point clouds,” Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit., vol. 2019-June, pp. 12689–12697, 2019.

The architecture of PointPillars consists of three stages as depicted in Figure 11. 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 plane, this stage summarizes the information of the points per pillar into 1D vectors. These vectors are rearranged appropriately to construct the pseudo-image that will feed the next stage. The second stage consists of a feature extraction backbone network that provides a high-level representation. This representation is subsequently processed by the third stage which is the adopted object detector, producing 3D bounding boxes and confidence scores for the classes of interest. In terms of computational complexity, the backbone network of the second stage, consisting of a number of 2D convolutions and 2D transpose convolutions, requires more than of the involved operations and, thus, we will focus on this stage in the following for its acceleration.

2.1.4.1 Dataset

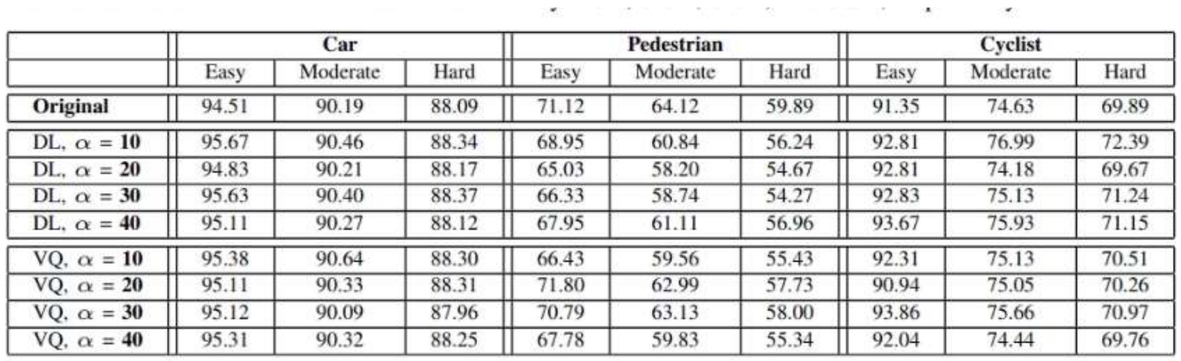
Both networks for 2D and 3D analysis were trained with the KITTI 3D object detection benchmark consisting of 7481 training images and 7518 test images as well as the corresponding point clouds, comprising a total of 80.256 labelled objects. In our study, three classes are mainly examined, cyclists, pedestrians and cars annotated with bounding boxes containing the objects in the 3D scene. 3716 annotated Velodyne point cloud scenes were used for training and 3769 annotated Velodyne point cloud scenes were used for testing and validation. For the deployment and retraining of PointPillars, the OpenPCDet framework[7] was employed. For the initial evaluation, pre-trained instances were used, while for the retraining, the Adam optimizer was employed with learning rate $l_r = 0.003$, weight decay rate $D_W = 10^{-2}$ and a batch size $B = 4$. Training took place in an NVIDIA Geforce RTX 2080 with 16GB RAM and compute capability 7.5. Furthermore, for the Pointpillars network the detection accuracy was evaluated on NVIDIA Jetson TX2, while for the PV-RCNN network, due to model size, the detection accuracy was evaluated on the NVIDIA Geforce RTX 2080. The following two figures present the average precision for Car, Pedestian and Cyclist classes for PoinPillar and PV-RCNN respectively.



	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Original	92.04	88.06	86.66	61.60	56.01	52.05	85.26	66.24	62.22
DL, α = **10**	91.83	87.32	84.72	56.45	51.00	46.93	82.66	63.05	58.87
DL, α = **20**	92.03	87.40	84.73	56.46	50.18	46.22	83.03	64.53	60.62
DL, α = **30**	91.76	86.80	84.08	53.49	47.14	43.71	78.82	60.61	56.98
DL, α = **40**	92.57	85.10	83.67	51.04	44.79	41.91	78.81	58.82	55.02
VQ, α = **10**	93.08	87.24	84.54	56.74	51.02	47.28	82.78	63.83	59.89
VQ, α = **20**	91.90	86.74	84.20	54.11	47.84	43.82	84.34	64.50	60.31
VQ, α = **30**	91.11	84.77	81.88	54.64	47.64	43.67	76.42	59.50	55.66
VQ, α = **40**	91.75	86.41	82.44	50.81	44.94	41.53	75.26	56.31	52.83

Figure 12: PointPillars BEV Average Precision. The shown acceleration ratios of α = 10, 20, 30, and 40 on the targeted layers, correspond to a total acceleration of PointPillars by 5.6×, 7.6×, 8.6×, and 9.2×, respectively

[7] O. D. Team, "OpenPCDet: An Open-source Toolbox for 3D Object Detection from Point Clouds." 2020.



	Car			Pedestrian			Cyclist		
	Easy	Moderate	Hard	Easy	Moderate	Hard	Easy	Moderate	Hard
Original	94.51	90.19	88.09	71.12	64.12	59.89	91.35	74.63	69.89
DL, α = **10**	95.67	90.46	88.34	68.95	60.84	56.24	92.81	76.99	72.39
DL, α = **20**	94.83	90.21	88.17	65.03	58.20	54.67	92.81	74.18	69.67
DL, α = **30**	95.63	90.40	88.37	66.33	58.74	54.27	92.83	75.13	71.24
DL, α = **40**	95.11	90.27	88.12	67.95	61.11	56.96	93.67	75.93	71.15
VQ, α = **10**	95.38	90.64	88.30	66.43	59.56	55.43	92.31	75.13	70.51
VQ, α = **20**	95.11	90.33	88.31	71.80	62.99	57.73	90.94	75.05	70.26
VQ, α = **30**	95.12	90.09	87.96	70.79	63.13	58.00	93.86	75.66	70.97
VQ, α = **40**	95.31	90.32	88.25	67.78	59.83	55.34	92.04	74.44	69.76

Figure 13: PV-RCNN BEV Average Precision. The presented acceleration ratios of α = 10, 20, 30, and 40 on the targeted layers, correspond to a total acceleration of the BEV-Backbone block by 4.5×, 5.5×, 6.0×, and 6.3×, respectively

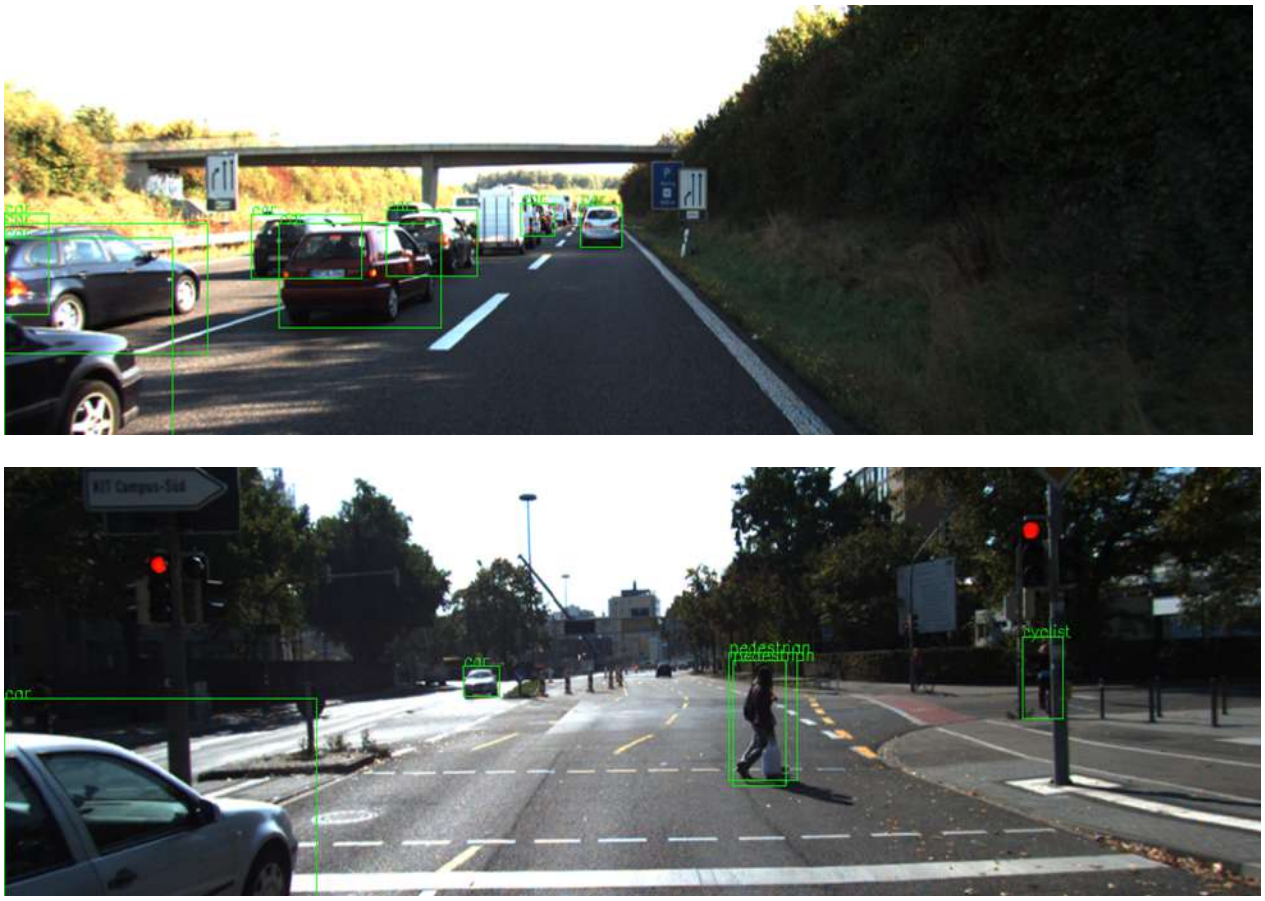


Figure 14: KITTI dataset examples.

Moreover, a data augmentation scheme was adopted, according to which the bounding boxes drift by $k_x * 150$ and $k_y * 150$ pixels across the x-axis and the y-axis, respectively, where $k_x, k_y \sim U(0,1)$. A 50% probability is also assumed to flip an object. For the training of the SqueezeDet architecture, Stochastic Gradient Descent (SGD) was employed with the following values for the hyperparameters (determined via experimentation); batch size $B = 8$, learning rate $LR = 10^{-4}$, with a weight decay rate $D_W = 10^{-4}$, a learning rate decay rate of $D_{LR} = 2 * LR/N_e$, number of steps $N_s = 3 \times N_{tr}$ and a dropout rate of 50%, over a total of $N_e = 300$ epochs. Training and testing took place in an NVIDIA GeForce GTX 1080 graphics card with 8GB VRAM and compute capability 6.1 in a Intel(R) Core(TM) i7-4790 CPU @ 3.60Hz based system with 32GB of RAM.

In all cases, training took place with a data augmentation scheme where the bounding boxes drift by pixels across the x-axis and pixels across the y-axis, where. A probability is also assumed to flip the object. For each detection, the Intersection Over Union (IOU) score is computed as the ratio of area of intersection to the area of union between the predicted and ground-truth bounding boxes. A true positive occurs when IOU and the predicted class is the same as the ground-truth class. A false positive occurs when IOU 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[8].

2.1.4.2 Metrics

For each detection, the Intersection Over Union (IOU) score is computed as the ratio of area of intersection to the area of union between the predicted and ground-truth bounding boxes. A true positive occurs when IOU> 0.5 and the predicted class is the same as the ground-truth class. A false positive occurs when IOU< 0.5 or a different class is detected, meaning that unmatched bounding boxes are taken as false positives for a given class.

The official KITTI evaluation detection metrics include bird eye view (BEV), 3D, 2D, and average orientation similarity (AOS). The 2D detection is done in the image plane and average orientation similarity assesses the average orientation (measured in BEV) similarity for 2D detections. The KITTI dataset is categorised into easy, moderate, and hard difficulties, and the official KITTI leaderboard is ranked by performance on moderate. For the sake of self-completeness, easy difficulty refers to a fully visible object with a minimum bounding height box of 40px and max truncation of 15%, moderate difficulty refers to a partially occluded object with a minimum bounding box height of 25px and max truncation of 30% and hard difficulty refers to a difficult to see an object with a minimum bounding box height of 40px and max truncation of 50%. Each 3D ground truth detection box is assigned to one out of three difficulty classes (*easy, moderate, hard*), and the used 40-point Interpolated Average Precision metric is separately computed on each difficulty class. It formulated the shape of the Precision/Recall curve as

$$\mathrm{AP}|_R = \frac{1}{|R|}\sum_{r \in R} \rho_{interp}(r) \tag{3}$$

averaging the precision values provided by $\rho_{interp}(r)$[9]. In our setting, we employ forty equally spaced recall levels,

$$R_{40} = \{1/40, 2/40, 3/40, \ldots, 1\} \tag{4}$$

The interpolation function is defined as

$$\rho_{interp}(r) = \max_{r\prime : r\prime \geq r} \rho(r') \tag{5}$$

where $\rho(r)$ gives the precision at recall r, meaning that instead of averaging over the actually observed precision values per point r, the maximum precision at recall value greater or equal than r is taken.

[8] D. M. W. Powers, "Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation," pp. 37–63, 2020.

[9] A. Simonelli, S. R. Bulo, L. Porzi, M. Ĺopez-Antequera, and P. Kontschieder, "Disentangling monocular 3d object detection," in Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1991–1999

2.1.4.3 Acceleration

We apply the detection models in a "full-model" acceleration scenario. It involves accelerating multiple (or all) convolutional layers of the original models and measuring the achieved performance of the accelerated networks.

It is noted, here, that, although full-range acceleration depends heavily on the performance of the technique used for the acceleration of each layer, it also involves experimentation over the strategy used for accelerating the layers and the involved fine-tuning (re-training) of the accelerated model. Here, we follow a stage-wise acceleration approach[10] with each stage involving the acceleration (and fixing) of one or more layers of the network, and, subsequently, fine-tuning (i.e., re-training) the remaining original layers. The starting point for each stage is the accelerated and fine-tuned version of the previous stage. The process begins with the original network and it is repeated until all target layers are accelerated. For fine-tuning and performance assessment, we use the training and validation datasets from KITTI, as previously explained. Integrating the accelerated version corresponds to the utilization of weights that have been accordingly processed with the proposed in D3.1 weight sharing approaches.

In our experiments, we apply the VQ and DL weight-sharing techniques to the PontPillars model, targeting the convolutional layers, and measuring the performance drop induced by the acceleration, compared to the original networks. The reported acceleration ratios are defined as the ratio of the original to the accelerated computational complexities, measured by the number of multiply-accumulate (MAC) operations.

PointPillars is a fully convolutional network with its feature-extraction part (both 2D and transposed convolution operators) being responsible for 97.7% of the total MAC operations required. In total Pointpillars network encompasses 4.835×10^6 parameters and require 63.835×10^9 MACs. For a good balance between acceleration and performance drop, we targeted the 2D convolutional layers of PointPillars (consuming approximately 47% of the total MACs), as well as the 4×4 transposed convolutional layer of the network (responsible for 44.4% of the total MACs), depicted with the red blocks in Fig. 2.3(a). Acceleration was performed in 16 acceleration stages with each stage involving the quantization of a particular layer, followed by fine-tuning. Using acceleration ratios of α = 10, 20, 30, and 40 on the targeted layers, lead to a reduction of the total required MACs by 82%, 86%, 88%, and 89%, or equivalently, to total model acceleration of PointPillars by $5.6\,\times$, $7.6\,\times$, $8.6\,\times$, and $9.2\,\times$, respectively.

2.1.4.4 Fusion

A late fusion strategy takes place combining 2D driven detections and 3D driven detections. Initially, 3D bounding boxes are projected upon the 2D plane and converted to 2D bounding boxes. To fuse 2D and 3D measurements a non-maximal suppression[11] driven approach takes place redefining the bounding boxes on the 2D space. Afterwards, to define vehicle range measurements 2D projects are matched to 3D points of the point cloud. Subsequently, each 3D point of the point cloud [x_i, y_i, z_i] is projected upon the 3D image.

The 3D bounding box is described by its center $T = [t_x, t_y, t_z]^T$, dimensions $D = [d_x, d_y, d_z]$, and orientation $R(\theta, \phi, \alpha)$ where θ is the azimuth, ϕ is the elevation and α is the roll angles. Given the pose

[10] 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.

[11] Rothe, Rasmus, Matthieu Guillaumin, and Luc Van Gool. "Non-maximum suppression for object detection by passing messages between windows." *Asian conference on computer vision*. Springer, Cham, 2014.

of the object in the camera coordinate frame $(R, T) \in SE(3)$ and the camera intrinsics matrix $R_{Rect}^{(0)}$, the transformation matrix $P_{Rect}^{(i)}$ the projection of a 3D point $X_o = [X, Y, Z, 1]^T$ in the object's coordinate frame into the image $x = [x, y, 1]^T$ is:

$$x = P_{Rect}^{(i)} * R_{Rect}^{(0)} * X$$

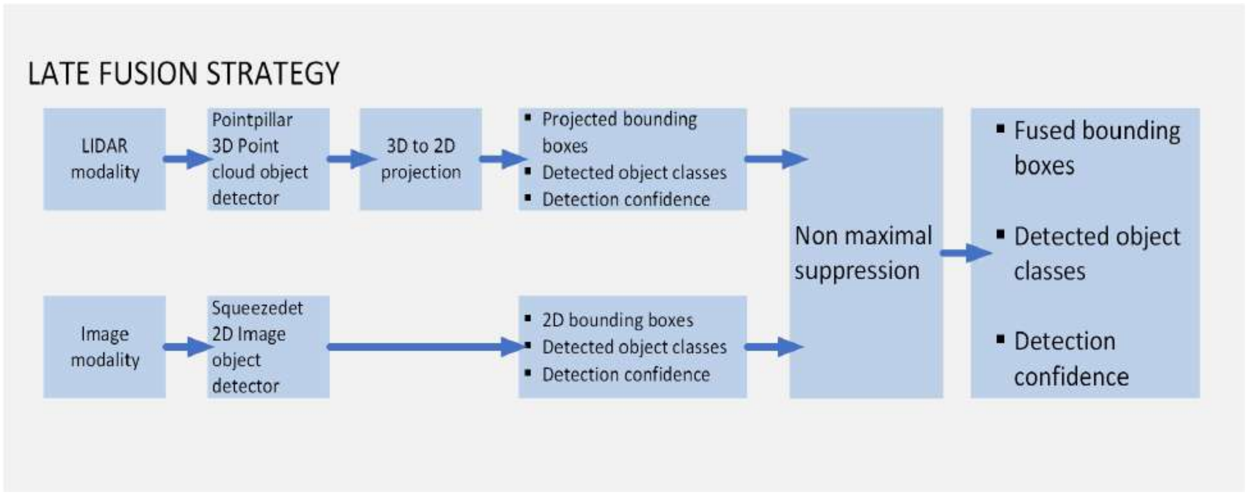


Figure 15: Fusion pipeline

The popular NMS algorithm is sequential in nature. At each iteration i, it selects the top scoring proposal $P(i)$ from the set $\mathcal{S}$ and removes all proposals in $\mathcal{S} - P(i)$ which have an overlap o greater than a threshold t. The complexity of each iteration is linear in the size of set $\mathcal{S}$.

Algorithm 1: Non-Maximum Suppression

```
Input: $B^c$, $S^c$, $N_t$
Initialisation:
$D^c \leftarrow \{\}$
while $B^c \neq empty$ do
    $m \leftarrow \text{argmax}\, S^c$
    $M \leftarrow b^c_m$
    $D^c \leftarrow D^c \cup M$
    $B^c \leftarrow B^c - M$
    for $b^c_i \in B^c$ do
        if $IoU(M, b^c_i) \geq N_t$ then
            $B^c \leftarrow B^c - b^c_i$
            $S^c \leftarrow S^c - s^c_i$
Output: $D^c$, $S^c$.
```

Figure 16: NMS algorithm

2.1.4.5 Results

This experiment aims to demonstrate the effectiveness of the multimodal fusion. Table 3 presents the average precision metrics for class car for image and LIDAR modalities and the fusion of both. The comparison is performed between the original networks and their accelerated versions either with the Vector Quantization or the Dictionary Learning based methods. All methods were compared with the Kitti Tracking dataset route[12] number #8. Furthermore, three difficulty levels are taken into account, namely Easy, Hard and Moderate. Their characteristics are presented in the Table 2

Table 2: KITTI difficulty levels

	Min. bounding box height	Max. occlusion level	Max. truncation
Easy	40px	Fully visible	15%
Moderate	25px	Partly occluded	30%
Hard	25px	Difficult to see	50%

As Table 3 reveals both image and LIDAR detectors and their fusion exhibit equivalent accuracies. However, applying the VQ acceleration strategy for acceleration factor equal to 10 significantly deteriorates the image detector with an accuracy drop from 76% to 63% for VQ and 73% for DL. For the same acceleration level the LIDAR detector seems not to be affected. Yet the fusion of both modalities maintains the average precision for accelerated networks nearly at the same level as the original network.

Table 3: Evaluation study results

#Route ID	Object Difficulty	Image			LIDAR			Fusion		
		Original	VQ10	DL10	Original	VQ10	DL10	Original	VQ10	DL10
0008	Easy	0.710	0.656	0.720	0.744	0.747	0.706	0.747	0.747	0.708
	Moderate	0.769	0.637	0.734	0.766	0.765	0.756	0.763	0.762	0.756
	Hard	0.748	0.616	0.681	0.767	0.761	0.758	0.763	0.751	0.757

The following figure presents a visual example of original and accelerated versions of image lidar and fusion driven car detections in a highway scene. The red boxes correspond to image detection, green 3D boxes to lidar detection, blue boxes to fusion of modalities and yellow boxes to groundtruth data. Even though the car on the left is clearly visible , the image based object detector fails in both cases and the reason could be that part of the car is hidden by the median strip. However the car is detected by LIDAR based detector and the fusion of modalities method.

[12] Geiger, A., Lenz, P., Stiller, C., &amp; Urtasun , R. (n.d.). The Kitti Vision Benchmark Suite : Tracking Dataset. The Kitti Vision Benchmark Suite. Retrieved August 11, 2022, from http://www.cvlibs.net/datasets/kitti/eval_tracking.php

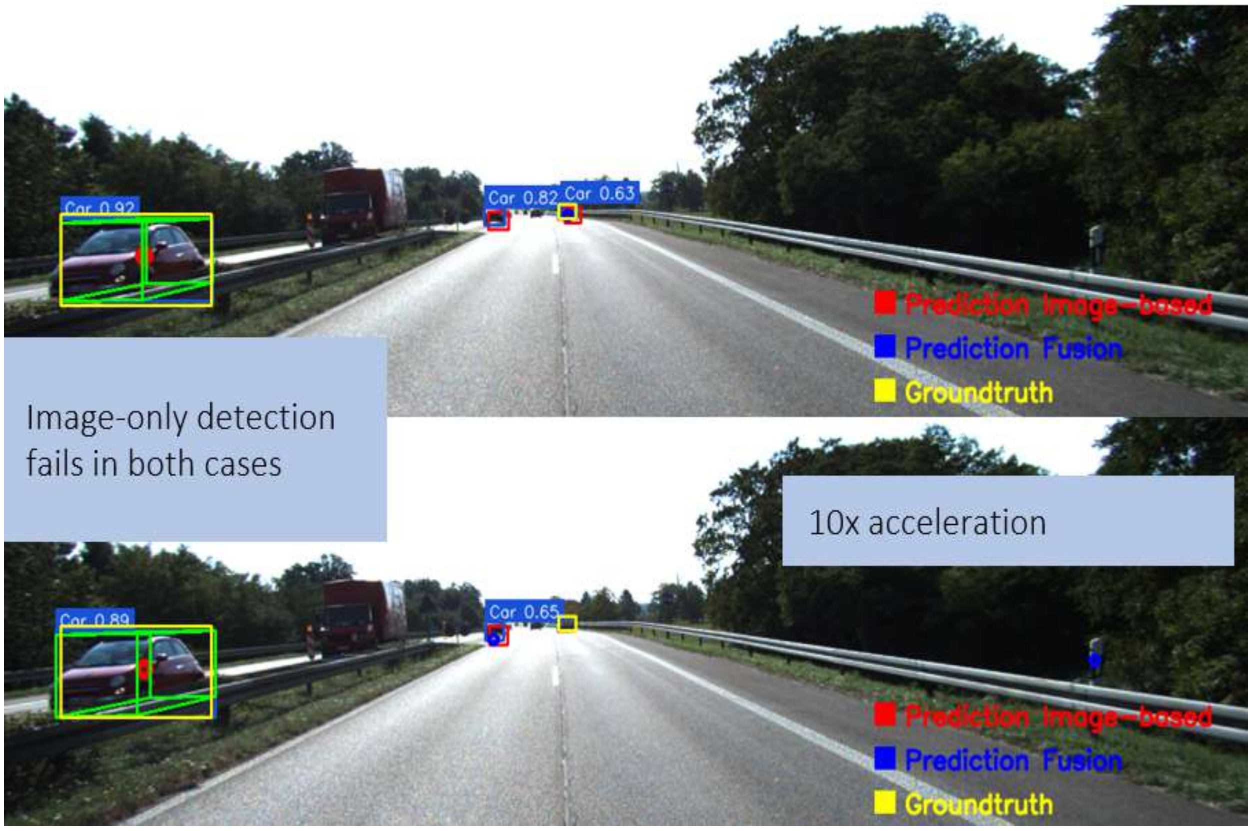


Figure 17: Detection example in highway scene

2.2 Semantic Data integration

This Section capitalizes on the previous approach of "early" fusion of selfish sensor agents and presents a **vertical AI mechanism based on a novel semantic data integration framework** for monitoring and safeguarding the ergonomics of human operators during a collaborative assembly task in an automotive manufacturing environment. The analysis outputs, which are not the raw sensor measurements as previously, are fed to an ontology-based semantic Knowledge Graph (KG) (which defines the set of incentives) through the flexible semantic data integration framework of CASPAR, already being deployed in various domains.

Semantic data integration (also known as "**semantic data fusion**") is the process of blending data from diverse and possibly heterogeneous sources, by employing a data-centric architecture built upon a semantic model[13]. The latter is typically based on **RDF** (**Resource Description Framework**)[14], a W3C standard for knowledge representation on the Web that represents resources as subject-predicate-object triples.

RDF-based semantic models are called ontologies and are stored in triplestores, which are special databases for the storage and retrieval of RDF triples. The ability to easily import and harmonise

[13] https://en.wikipedia.org/wiki/Semantic_integration

[14] https://www.w3.org/RDF/

heterogeneous data from multiple sources and interlink it as RDF statements into an RDF triplestore is essential for many knowledge management solutions running "on-top" of the semantic model. In addition, being the backbone of **semantic technologies**, RDF enables the **inference of new facts** from the existing data as well as the **enrichment of the available knowledge** by accessing Linked Open Data (LOD) resources available on the Web. Figure 18 gives an overview of the process of semantic integration. The semantic model is typically initially empty and contains only the schema of the application domain, namely, definitions of the basic concepts and interrelationships.

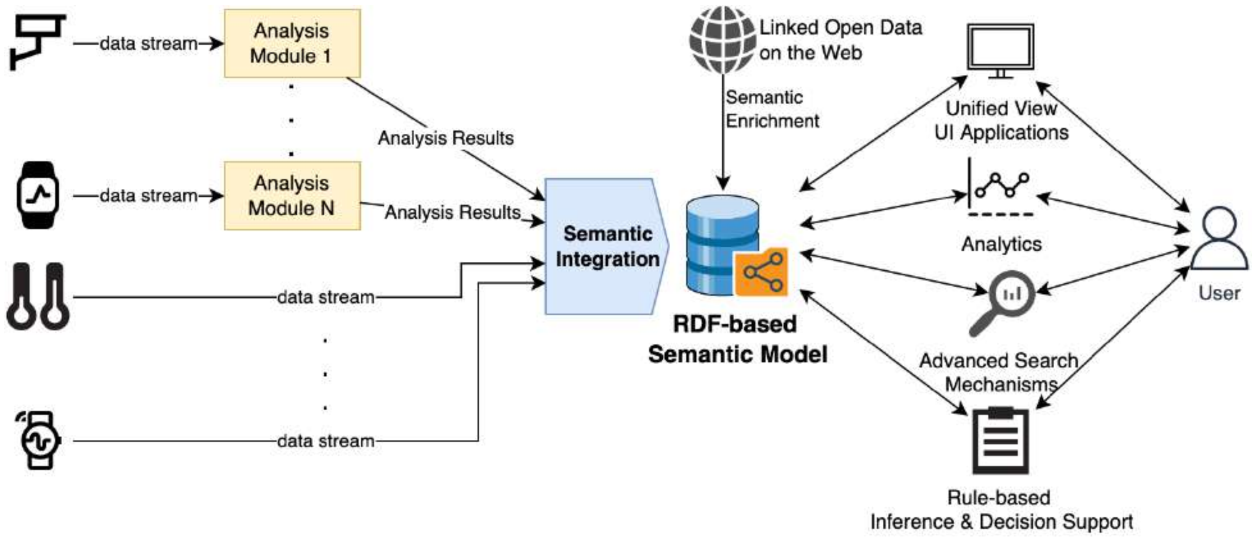


Figure 18: Overview of the process of semantic data integration.

The input to the semantic model can be either raw data (e.g., sensor measurements) or the result from analysing raw data (e.g., objects detected by a computer vision module). The result of the semantic integration process is **a semantic knowledge graph** (**KG**), i.e., a semantic model populated with the input data that delivers a unified view of the available information to the end-user. And various applications can run on-top of the semantic model, like, e.g., analytics, predictions, and rule-based decision support.

Within the context of CPSoSaware T4.5, we deployed a semantic data integration framework for ingesting outputs from other analysis components into a uniform semantic model. We investigated two application domains, coinciding with the two use cases in the project: automotive and manufacturing. For each case, we created an initially empty semantic model and a respective semantic data integration infrastructure for populating the model with instance data from other CPSoSaware components. CTL's proprietary semantic data integration framework, called CASPAR[15], was used in both use cases, and was extended accordingly, in order to satisfy the respective needs of each domain. Moreover, for both use cases, we developed domain ontologies based on the SSN/SOSA[16] W3C recommendation for representing sensors, observations, samples and actuations. The following subsections present in more detail the work conducted in each of the two use cases.

[15] https://caspar.catalink.eu/

[16] W3C Recommendation 19 October 2017: http://www.w3.org/ns/sosa/

2.2.1 Automotive Pillar

Within the automotive pillar, we focused on monitoring (a) **the system health**, and, (b) **the driver's state** during a driving session, and, thus, deployed the system displayed in Figure 19.

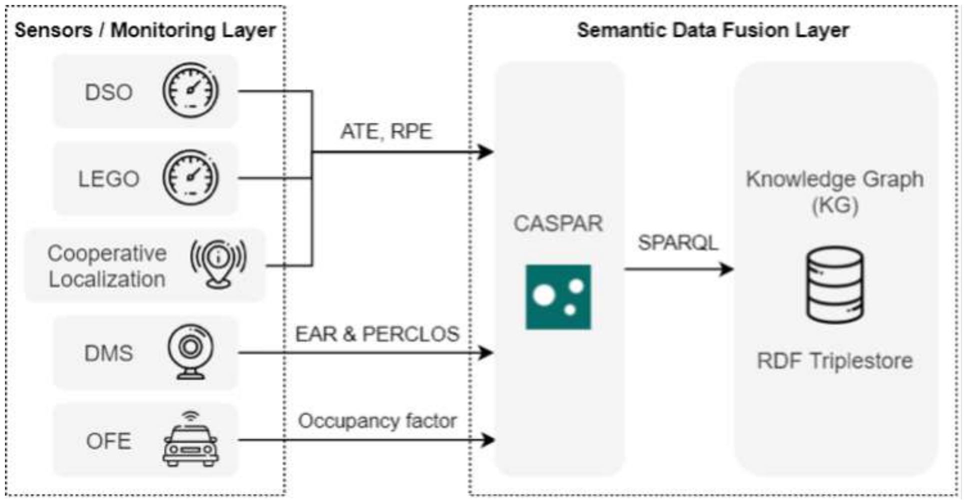


Figure 19: Overview of the system architecture for the automotive pillar

Adopting the **microservice methodology**[17], we defined a set of independent, replicable services that collaboratively fulfil the system's functionality. For the communications among services, we deployed RabbitMQ[18], a popular open-source message broker that is scalable and industry-ready.

The system components in the monitoring layer periodically collect and analyse data related to the driver, the vehicle and its surroundings. The following monitoring components are currently integrated:

Odometry Algorithms (Direct Sparse Odometry - DSO[19], LeGO-LOAM[20], Cooperative Localization - CL), calculating the Absolute Trajectory Error (ATE) and the Relative Pose Error (RPE).

Driver Monitoring System (DMS) capturing frontal facial images of the driver to assess fatigue levels based on the activity of the eyes. The driver's drowsiness is measured based on two metrics, the Eye Aspect Ratio (EAR)[21] and the PERcentage of Eye CLOSure (PERCLOS)[22].

Occupancy Factor (OF), which is an empirical metric, extracted by analysing the point cloud of the scene that has been acquired by the LiDAR device. It indicates how "clear and open" the road is beyond the driver's field of view.

---

[17] S. Newman, "Building microservices: designing fine-grained systems" O'Reilly Media Inc, 2015.

[18] https://www.rabbitmq.com/

[19] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry" IEEE transactions on pattern analysis and machine intelligence, vol. 40(3), pp. 611-625, 2017.

[20] T. Shan and B. Englot, "Lego-loam: Lightweight and ground-optimized LiDAR odometry and mapping on variable terrain" In 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS), IEEE Press, Oct. 2018, pp. 4758-4765.

[21] F. You, X. Li, Y. Gong, H. Wang, and H. Li, "A real-time driving drowsiness detection algorithm with individual differences consideration" IEEE Access, vol 7, pp. 179396-179408, 2019.

[22] D. F. Dinges and R. Grace, "PERCLOS: A valid psychophysiological measure of alertness as assessed by psychomotor vigilance" US Department of Transportation, Federal Highway Administration, Publication Number FHWA-MCRT-98-006, 1998.

The modularity provided by the microservice approach, coupled with the straightforward system design, enables the integration of additional third-party data sources (e.g., weather or traffic condition reports) with minimum effort.

The outputs and observations produced by the monitoring layer are communicated to the semantic data fusion layer via a dedicated RabbitMQ exchange. At this stage, they are mapped to ontology concepts, resulting in a unified KG, which is instantiated in an RDF triplestore by the CASPAR component.

The following subsections present the two scenarios we investigated within the context of the Automotive Use Case, which were also presented in more detail at the 15th Int. Conf. on Advances in Semantic Processing (SEMAPRO 2021)[23].

2.2.1.1 Scenario #1: Evaluating the Robustness of Odometry Algorithms

In the first scenario we rely on an end-to-end testing framework, based on the CARLA open-source urban driving simulator[24], for generating synthetic sensory data and evaluating the three aforementioned odometry algorithms against different weather and lighting conditions. Each algorithm uses a different modality and our purpose is to study the effect of the changing conditions on the efficiency of each algorithm. Based on the architecture described above, ATE and RPE measurements are sent via the message bus to the CASPAR semantic data fusion framework and are ingested into the KG. Indicatively, 1226 observations were submitted for a driving simulation of 126 seconds. Figure 20 displays an excerpt of the observations.

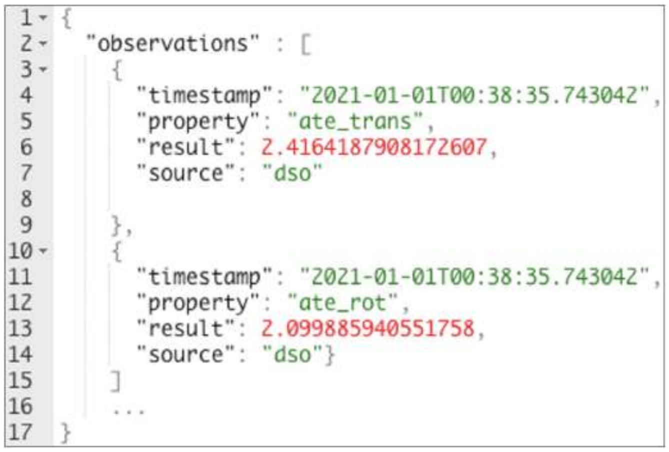

```
{
  "observations" : [
    {
      "timestamp": "2021-01-01T00:38:35.743042",
      "property": "ate_trans",
      "result": 2.4164187908172607,
      "source": "dso"

    },
    {
      "timestamp": "2021-01-01T00:38:35.743042",
      "property": "ate_rot",
      "result": 2.099885940551758,
      "source": "dso"}
    ]
    ...
}
```

Figure 20: Excerpt of the ATE and RPE observations

CASPAR converts the inputs into RDF-compatible representations via the use of user-defined mappings, which associate input data fields to semantic entities (concepts, relationships, etc.). The following listing displays the mapping for converting the JSON excerpt in Figure 20 into SPARQL queries that populate the semantic model with the appropriate instance data.

```
{
  "templates": [
```

[23] Kontopoulos, E. et al.: An Extensible Semantic Data Fusion Framework for Auton-omous Vehicles. In: 15th Int. Conf. on Advances in Semantic Processing (SEMAPRO 2021), pp. 5-11. IARIA (2021).

[24] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun, "CARLA: An open urban driving simulator" In Conference on robot learning (PMLR), Oct. 2017, pp. 1-16.

```
{
  "prefixes": [
    {
      "prefix": "rdf",
      "namespace": "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    },
    {
      "prefix": "rdfs",
      "namespace": "http://www.w3.org/2000/01/rdf-schema#"
    },
    {
      "prefix": "xsd",
      "namespace": "http://www.w3.org/2001/XMLSchema#"
    },
    {
      "prefix": "owl",
      "namespace": "http://www.w3.org/2002/07/owl#"
    },
    {
      "prefix": "sosa",
      "namespace": "http://www.w3.org/ns/sosa/"
    },
          {
      "prefix": "cpsos",
      "namespace": "http://www.example.com/cpsosaware#"
    }
  ],
  "individuals": [
    {
      "path": "observations.[*]",
      "namespace": "http://www.example.com/cpsosaware#",
      "classes": [
        "sosa:Observation"
      ],
      "properties": [
                  {
          "predicates": [
            "sosa:observedProperty"
          ],
          "object": {
            "path": "observations.[*].property",
          }
        },
                  {
          "predicates": [
            "sosa:madeBySensor"
          ],
          "object": {
            "path": "observations.[*].source",
          }
                  },
                  {
          "predicates": [
            "sosa:hasResultTime"
          ],
          "object": {
            "path": "observations.[*].timestamp",
                        "datatype": "xsd:dateTime"
          }
                  },
                  {
          "predicates": [
            "sosa:hasSimpleResult"
          ],
          "object": {
            "path": "observations.[*].result",
                        "datatype": "xsd:decimal"
          }
                  }
      ],
      "update_on": []
    },
          {
      "path": "observations.[*].property",
      "namespace": "http://www.example.com/cpsosaware#",
      "classes": [
        "sosa:ObservableProperty"
      ],
      "properties": [
                  {
          "predicates": [
            "rdfs:label"
          ],
          "object": {
            "path": "observations.[*].property"
          }
        }
      ],
      "update_on": [
        {
          "predicates": ["rdfs:label"],
          "object": {
            "path": "observations.[*].property"
          }
        }
      ]
    },
          {
      "path": "observations.[*].source",
      "namespace": "http://www.example.com/cpsosaware#",
      "classes": [
        "sosa:Sensor"
      ],
      "properties": [
                  {
          "predicates": [
            "rdfs:label"
          ],
            "object": {
              "path": "observations.[*].source"
            }
          }
        ],
        "update_on": [
          {
            "predicates": ["rdfs:label"],
            "object": {
              "path": "observations.[*].source"
            }
          }
        ]
      }
    ]
  }
]
}
```

The process of ontology population corresponds to the generation of new individuals (nodes) and properties (edges) in the KG. In a nutshell, mappings are JSON-structured files composed of definitions of templates, individuals and properties:

A `template` serves as the mechanism for focusing on specific parts of the input.

A `template` contains a set of individuals, which declare the nodes that need to be created or updated in the KG. From the perspective of using an ontology as the KG schema, an individual is an instance of one or more classes.

A `property` indicates a desired edge that needs to be created in the KG, connecting a node with another node or with a literal value. In ontology terms, a property may correspond to an object property (i.e., relating two instances with each other) or a datatype property (i.e., relating an instance to a literal value). Properties in mappings are defined by a set of predicates, meaning the relationship types of the ontology, and objects, which indicate the value that will be given to the property. Therefore, objects, which are also declared as JSON paths, can either point to literal values in the input JSON or to other individuals.

The optional `update_on` field allows associating newly arrived knowledge with nodes already existing in the KG.

Figure 21 illustrates the representation of the sample observations from Figure 20 in Graffoo format[25] fused inside the KG, after the process of ontology population is completed by CASPAR.

[25] https://essepuntato.it/graffoo/

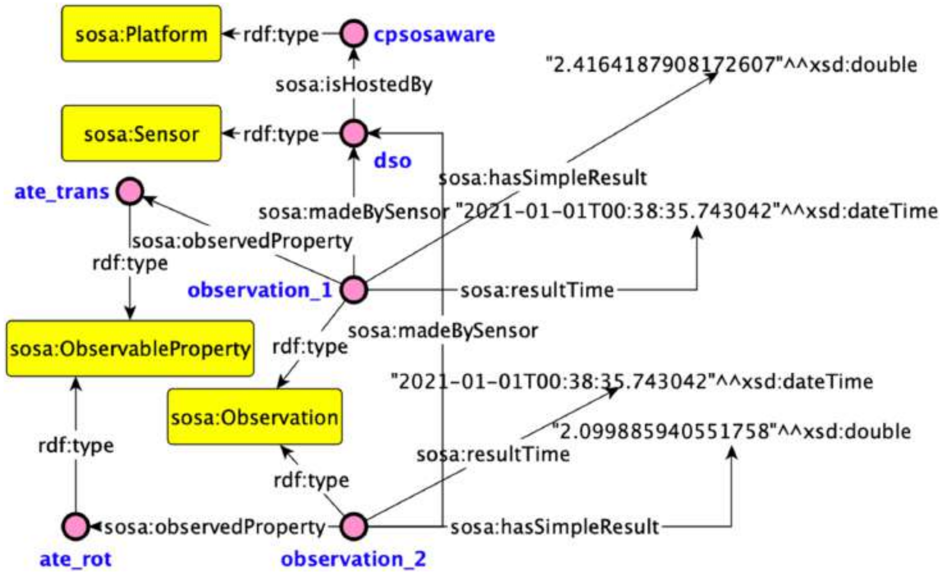


Figure 21: Representation of the sample observations in the KG

After the population of the KG is complete, useful insights regarding the performance of the algorithms can be extracted. Figure 22 displays such an example, where the LiDAR-based odometry algorithm (LeGO-LOAM) presents better results with regards to RPE and seems to be more robust, constituting thus a better candidate in conditions similar to the specific driving simulation session.

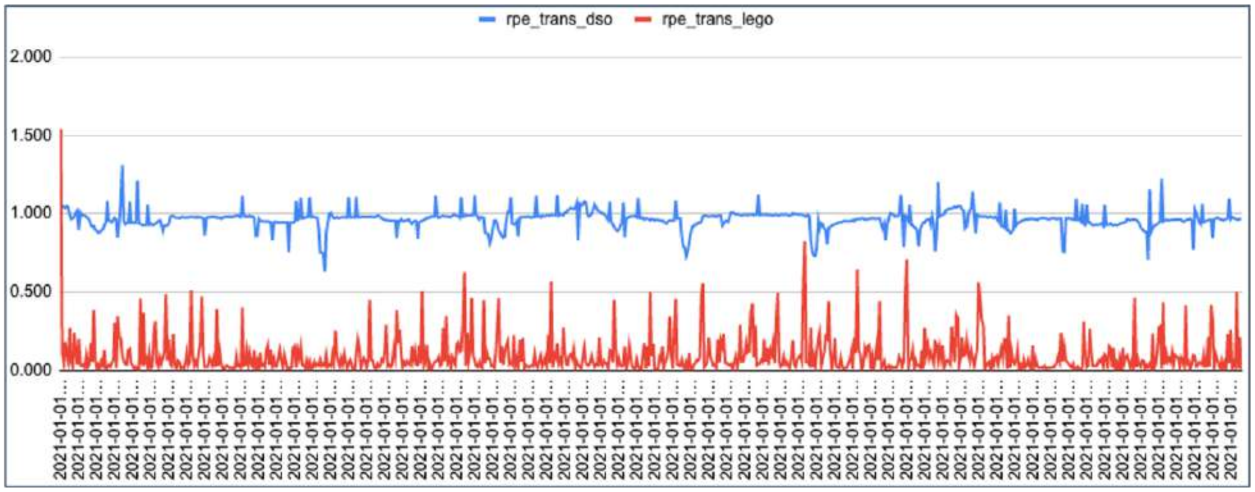


Figure 22: Performance comparison of LeGO vs DSO for a simulation session.

2.2.1.2 *Scenario #2: Calculating Risk Levels during a Driving Session*

In the second scenario, our objective is to inform the driver about potential risks during a driving session. We focus on two factors: The driver's drowsiness and the free available space of the road. For this purpose, two components have been developed: (a) the Driver Monitoring System (DMS) component, and (b) the Occupancy Factor Estimation (OFE) component. For the evaluation of our implementation, we integrated the DMS with CARLA, whose spectacularly photorealistic graphics provide an immersive driving experience. The simulator provides the flexibility to design a variety of driving scenarios under different states of driver's drowsiness and different conditions of the road (e.g., the state of the traffic), in a safe environment for the operator who tests the implementation. The setup of the integration uses the Logitech G29 steering wheel for enhancing the driving sense, as well as a static web camera that captures the face of the driver in real-time.

Similar to the previous scenario, the DMS and OFE modules submit their observations, namely the PERCLOS and OF measurements, to CASPAR via RabbitMQ. However, an upgrade compared to scenario #1 entails a set of rules for calculating the risk levels during the simulated driving session (see Table 4).

Table 4: Set of rules for calculating the risk level.

	PERCLOS < 0.25	**PERCLOS >= 0.25 & <0.7**	**PERCLOS >= 0.7**
OF > 280	Low risk (1)	Be aware (2)	High risk (3)
OF <= 280 & >200	Low risk (1)	Be aware (2)	High risk (3)
OF <= 200	Be aware (2)	High risk (3)	High risk (3)

Risk level 1 corresponds to a "low risk" driving situation, where the driver is focused and drives carefully in a full open-eyed state (without any observed drowsiness). Moreover, the road is free from other vehicles, thus providing an unobstructed area for driving. On the other hand, risk level 3 corresponds to a "high risk" driving situation where the driver demonstrates intense drowsiness, as identified by the facial analysis of the DMS component, with intense drowsiness and/or the unobstructed area of the road is restricted (due to obstacles, a lot of traffic, small-ranged road, etc).

After the KG is populated through CASPAR, the above ruleset is executed in the form of a respective SPARQL query "on-top" of the KG. The result is a risk level report, as illustrated in Figure 23. Outputs like this can constitute parts of reports, e.g., after traffic accidents.

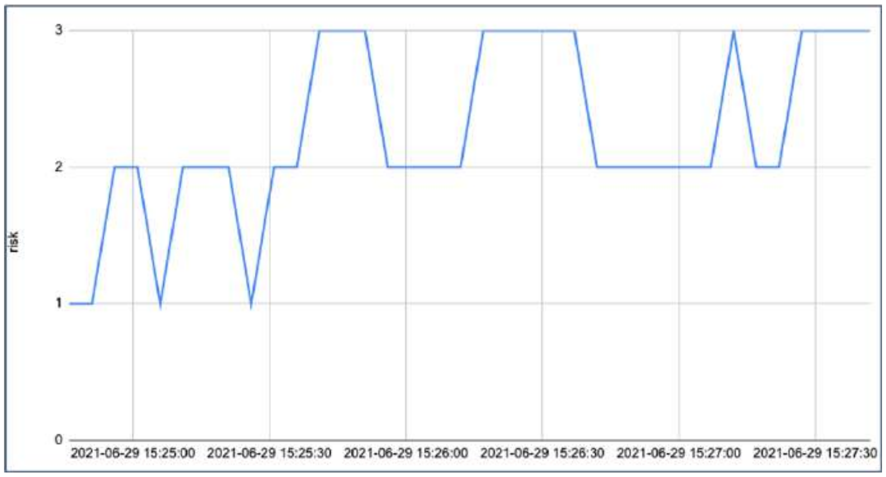


Figure 23: Graph indicating the risk levels during a driving session.

2.2.2 Manufacturing Pillar

The scenario we examined in the context of the Manufacturing Use Case revolves around a collaborative application in an automotive assembly line and was presented at the Industry Track of the 2022 Extended

Semantic Web Conference (ESWC'22)26. According to the scenario, a human operator performs manual assembly operations on a windshield handled and moved by a robot before assembly on the chassis. Our overarching aim is to protect the operators from injuries and muscle strain and to reduce their body's strain by performing biophysics assessment for ergonomic optimization. Towards this end, we deploy a semantic data integration framework for monitoring the human operators' safety and well-being as they are performing the requested operations.

The proposed implementation focuses on adjusting the position of the windshield according to the operator's ergonomics and providing personalised suggestions and warnings to the operator based on their postures and the way that they use their body to perform an operation, in order to avoid long-term musculoskeletal problems and other health and/or safety risks. The foreseen benefits of our solution are: (a) improvement of the workers' wellbeing at work; (b) mitigation of risks and accidents; (c) flexibility of workplace management.

2.2.2.1 Setup and Deployment

A set of IoT sensors submit their measurements to respective analysis components: (a) footage from static cameras analysed by computer vision components for estimating the operator's anthropometrics parameters (i.e., posture); (b) wearables (inertial measurement units – IMUs, i.e., accelerometers and gyroscopes) for motion analysis and body tracking. The analysis outputs (and not the raw sensor measurements) are then fed to an ontology-based semantic Knowledge Graph (KG) through CASPAR. Our overall aim is to perform a proactive ergonomics optimization of the equipment. Figure 24 gives a diagrammatic overview of the workflow. Note that only the higher-level analysis outputs are stored and not the raw data itself, preventing issues of performance and storage costs.

The deployment is currently being tested in a virtual environment (i.e., simulator designed in Unity[27]) and will soon be tested in a real factory setting at CRF premises. The simulation involves three static RGB cameras located in three different areas of the working environment monitoring the "human's" (i.e., a digital human model) actions, while he collaborates with a robot to perform a specific task together.

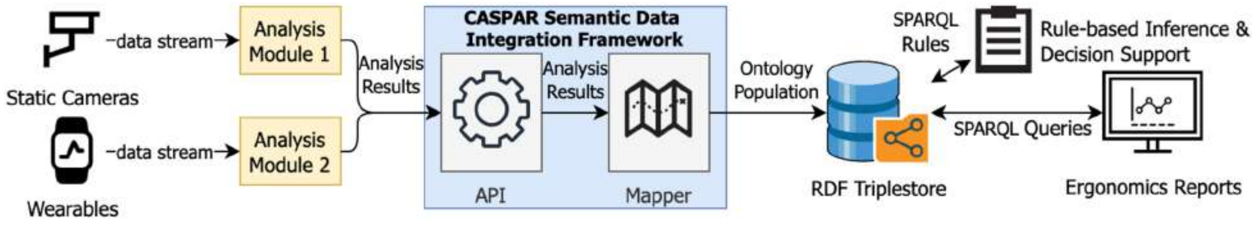


Figure 24: Workflow overview.

Figure 25 illustrates a set of snapshots from the three different views in the simulated environment. A pose estimation algorithm extracts in real time the posture landmarks and a confidence rate for each estimation. The outputs are fed into the ontology via semantic data integration, while a set of rules determine the camera with the best view. Based on this, our next aim will be to calculate the RULA (Rapid Upper Limb Assessment) score, based on the joint angles. RULA is a well-established metric for calculating the risk of musculoskeletal loading within upper limbs and neck.

[26] https://2022.eswc-conferences.org/wp-content/uploads/2022/05/industry_Kontopoulos_et_al_paper_205.pdf

[27] Unity homepage: https://unity.com/

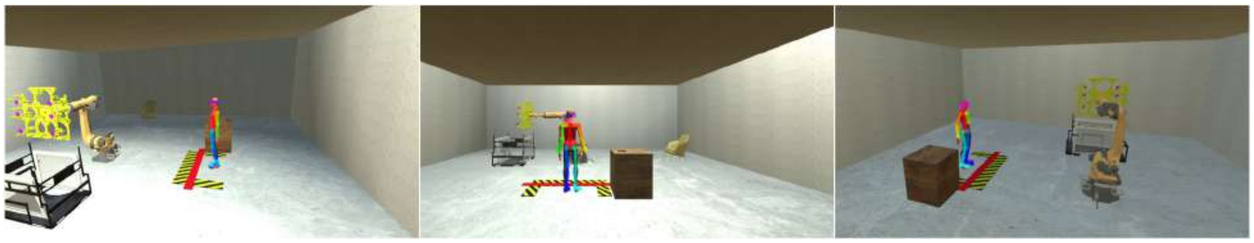


Figure 25: Simulated environment snapshot.

The same pipeline will be adopted in the real-life industrial environment in the coming months, with camera-based estimations from computer vision algorithms now coupled with body joints' estimation based on the IMU sensors in online monitoring.

2.2.2.2 Input Data

The pose estimation algorithm extracts posture landmarks and confidence rates roughly at a per second rate and generates JSON outputs in the following format:

```
{
   "timestamp": "02-09-2021 02:11:09.000",
   "source": "backView",
   "results": [
     {
        "property": "landmark_1",
        "result": 0.86293
     },
     {
        "property": "landmark_2",
        "result": 0.895670333333333335
     },
     ...
     {
        "property": "avg_cr",
        "result": 0.70686023555555555
     }
   ]
}
```

The `source` field takes one of three values, `backView`, `frontView`, `sideView`, while a set of 25 landmarks on the human operator's body are detected, along with respective confidence rates and an overall average confidence rate for the whole set of landmarks per timestamp.

Similarly, to what has already been demonstrated above, the generated JSON outputs are fed to the CASPAR semantic data integration framework, which populates the underlying semantic model accordingly, via suitable SPARQL update queries.

2.2.2.3 Semantic Model

Figure 26 below displays in Graffoo notation the core semantic model, which extends the W3C-recommended SOSA/SSN, similarly to what was also demonstrated for the first CPSoSaware use case.

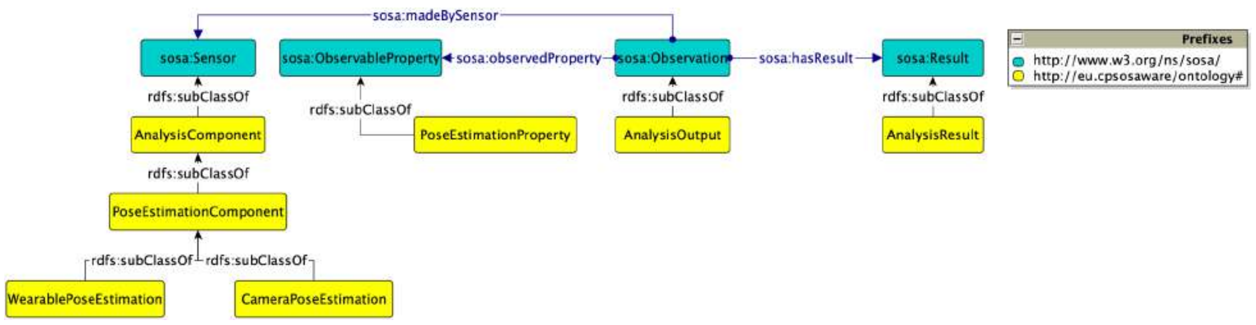


Figure 26: Core semantic model for the Manufacturing Use Case

Below is a description of the core concepts included in the semantic model:

- `AnalysisComponent` (and its specialisations) represents the components receiving the raw data measurements (e.g., camera feed and measurements from wearables), performing the respective analyses, and generating the results.
- `AnalysisOutput` represents the "observations", i.e., the outputs generated by the analysis components, accompanied by a respective timestamp.
- `AnalysisResult` represents the actual results from the analysis, i.e., values and units (if applicable).
- `PoseEstimationProperty` represents the observable property that is relevant to estimating the correctness or not of the human operator's pose.

2.2.2.4 Semantic Data Integration

Through the use of a specified mapping, CASPAR populates the core semantic model with instance data coming from the pose estimation algorithm. Figure 27 illustrates a sample instantiation representing the JSON excerpt presented in Subsection 'Input Data'.

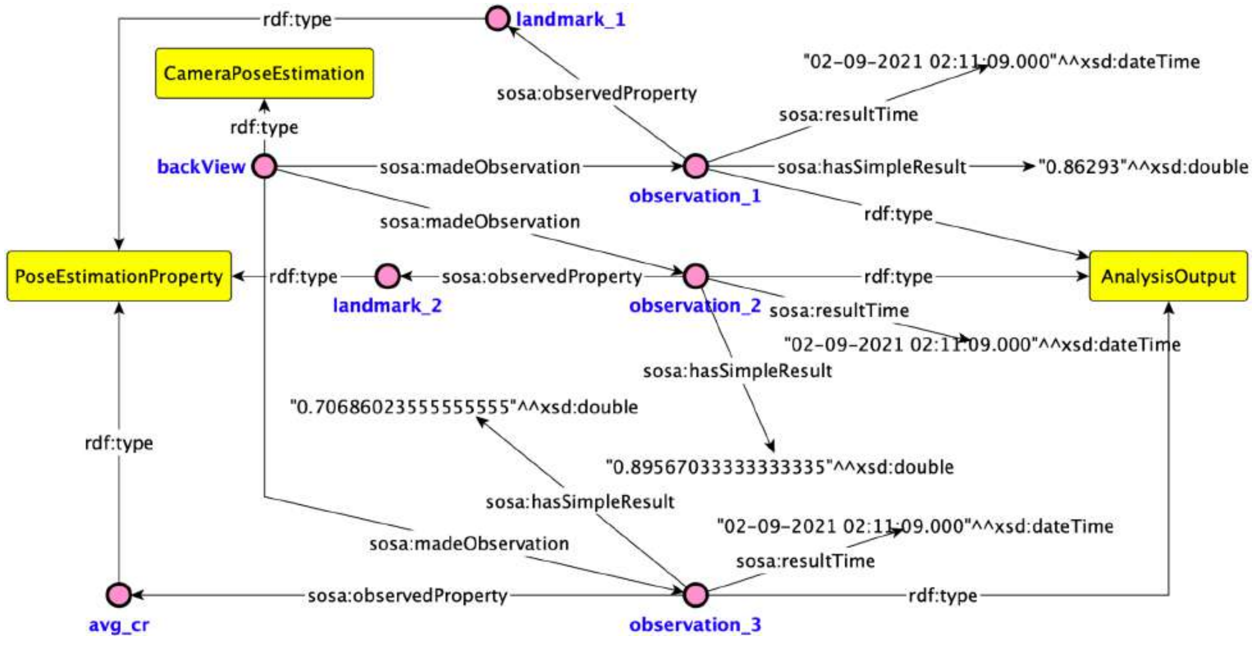


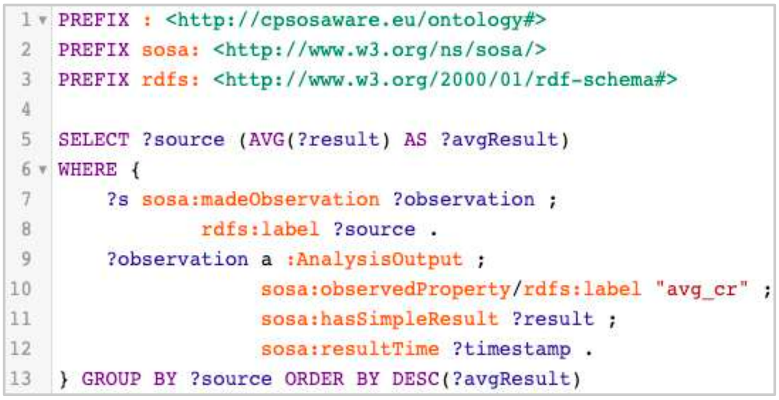
Figure 27: Sample instantiations

2.2.2.5 Results

This subsection presents insights generated by running respective SPARQL queries on top of the populated semantic model.

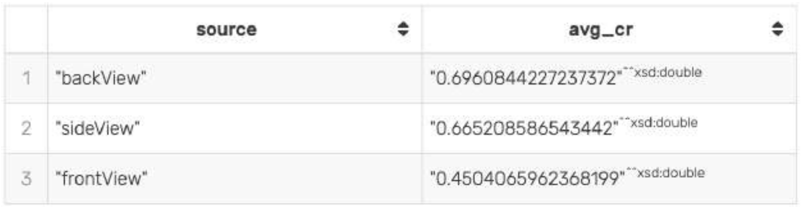
1. Retrieve the Average Confidence Rate (CR) per Source

SPARQL Query



```
PREFIX : <http://cpsosaware.eu/ontology#>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?source (AVG(?result) AS ?avgResult)
WHERE {
    ?s sosa:madeObservation ?observation ;
            rdfs:label ?source .
    ?observation a :AnalysisOutput ;
                 sosa:observedProperty/rdfs:label "avg_cr" ;
                 sosa:hasSimpleResult ?result ;
                 sosa:resultTime ?timestamp .
} GROUP BY ?source ORDER BY DESC(?avgResult)
```

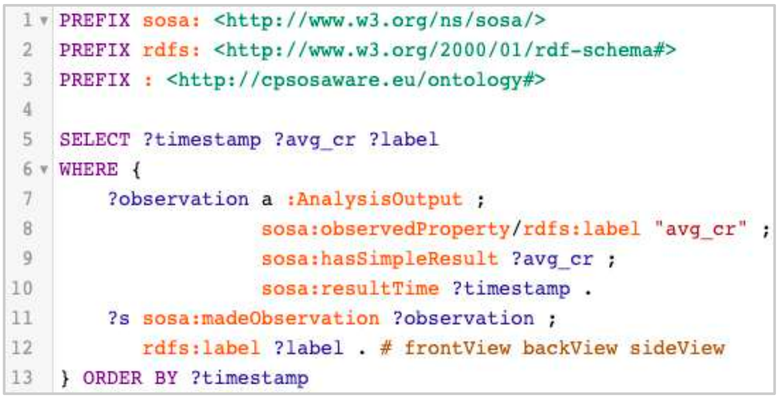
Result



	source	avg_cr
1	"backView"	"0.6960844227237372"^^xsd:double
2	"sideView"	"0.665208586543442"^^xsd:double
3	"frontView"	"0.4504065962368199"^^xsd:double

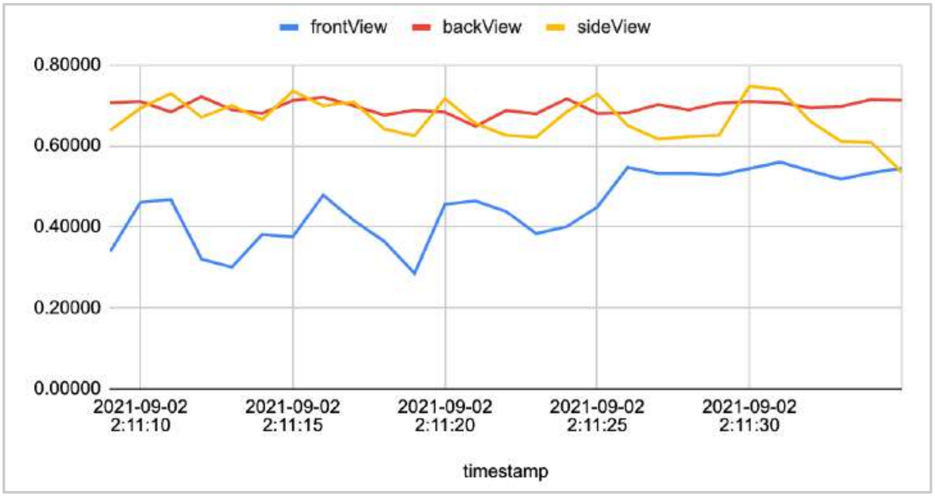
2. Evolution of Average CR per Source for the Whole Session

SPARQL Query



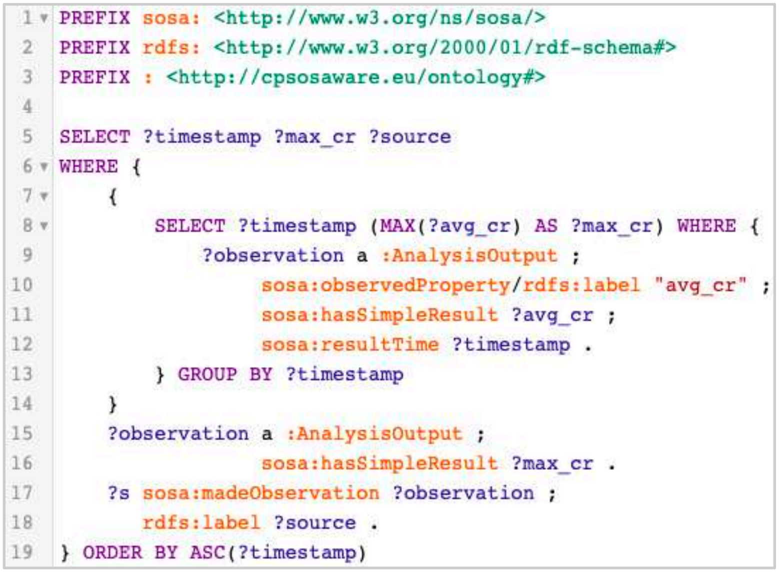
```
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://cpsosaware.eu/ontology#>

SELECT ?timestamp ?avg_cr ?label
WHERE {
    ?observation a :AnalysisOutput ;
                 sosa:observedProperty/rdfs:label "avg_cr" ;
                 sosa:hasSimpleResult ?avg_cr ;
                 sosa:resultTime ?timestamp .
    ?s sosa:madeObservation ?observation ;
       rdfs:label ?label . # frontView backView sideView
} ORDER BY ?timestamp
```

Result



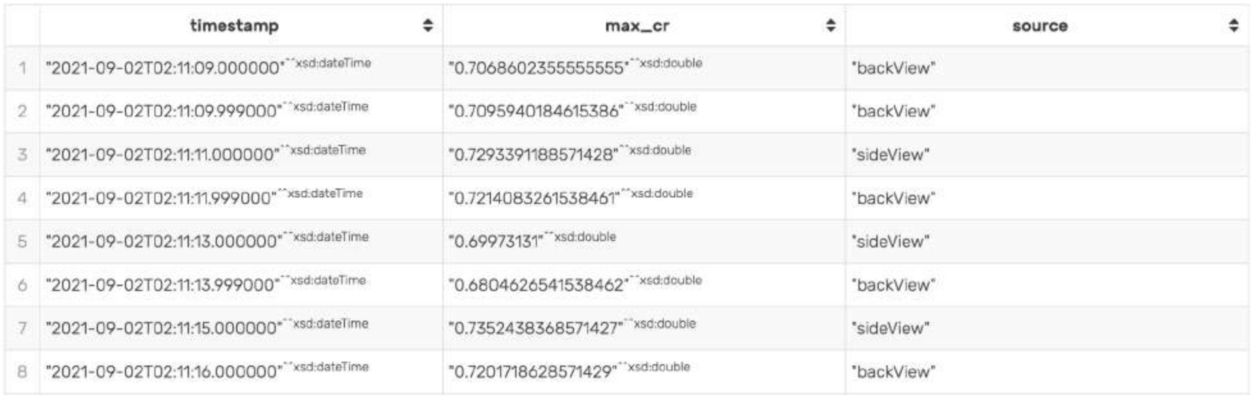
3. Retrieve the Source with the Best CR per Timestamp

SPARQL Query



```
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://cpsosaware.eu/ontology#>

SELECT ?timestamp ?max_cr ?source
WHERE {
    {
        SELECT ?timestamp (MAX(?avg_cr) AS ?max_cr) WHERE {
            ?observation a :AnalysisOutput ;
                 sosa:observedProperty/rdfs:label "avg_cr" ;
                 sosa:hasSimpleResult ?avg_cr ;
                 sosa:resultTime ?timestamp .
        } GROUP BY ?timestamp
    }
    ?observation a :AnalysisOutput ;
                 sosa:hasSimpleResult ?max_cr .
    ?s sosa:madeObservation ?observation ;
       rdfs:label ?source .
} ORDER BY ASC(?timestamp)
```

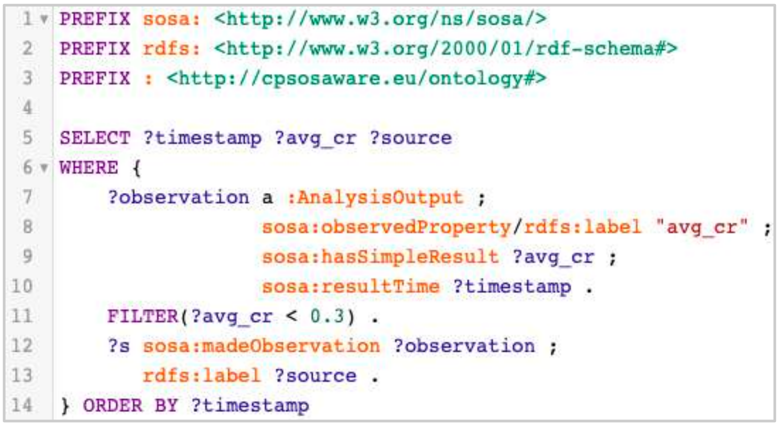
Result (for the first 8 timestamps)



	timestamp	max_cr	source
1	"2021-09-02T02:11:09.000000"^^xsd:dateTime	"0.7068602355555555"^^xsd:double	"backView"
2	"2021-09-02T02:11:09.999000"^^xsd:dateTime	"0.7095940184615386"^^xsd:double	"backView"
3	"2021-09-02T02:11:11.000000"^^xsd:dateTime	"0.7293391188571428"^^xsd:double	"sideView"
4	"2021-09-02T02:11:11.999000"^^xsd:dateTime	"0.7214083261538461"^^xsd:double	"backView"
5	"2021-09-02T02:11:13.000000"^^xsd:dateTime	"0.69973131"^^xsd:double	"sideView"
6	"2021-09-02T02:11:13.999000"^^xsd:dateTime	"0.6804626541538462"^^xsd:double	"backView"
7	"2021-09-02T02:11:15.000000"^^xsd:dateTime	"0.7352438368571427"^^xsd:double	"sideView"
8	"2021-09-02T02:11:16.000000"^^xsd:dateTime	"0.7201718628571429"^^xsd:double	"backView"

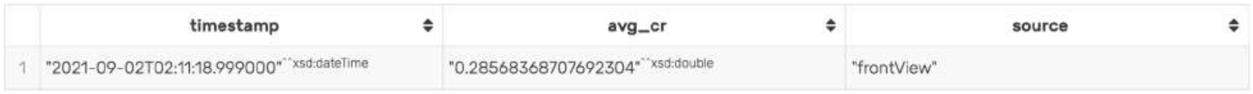
4. Indicate low CRs based on User-defined Threshold

SPARQL Query



```
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX : <http://cpsosaware.eu/ontology#>

SELECT ?timestamp ?avg_cr ?source
WHERE {
    ?observation a :AnalysisOutput ;
                 sosa:observedProperty/rdfs:label "avg_cr" ;
                 sosa:hasSimpleResult ?avg_cr ;
                 sosa:resultTime ?timestamp .
    FILTER(?avg_cr < 0.3) .
    ?s sosa:madeObservation ?observation ;
       rdfs:label ?source .
} ORDER BY ?timestamp
```

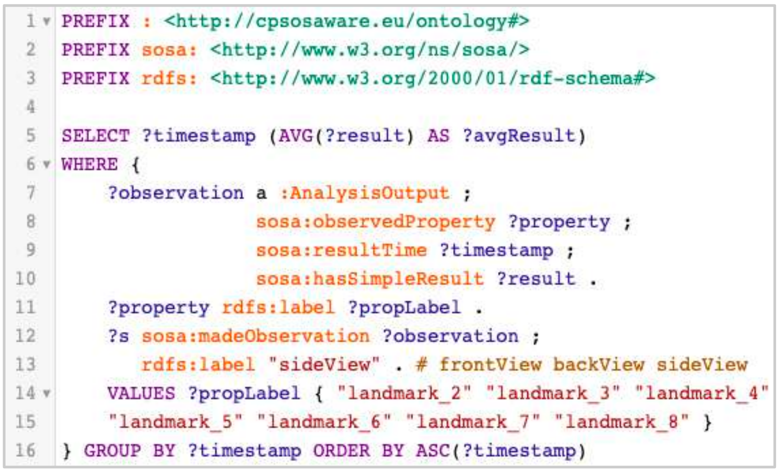
Result (for avgCR < 0.3, threshold specified by user)



	timestamp	avg_cr	source
1	"2021-09-02T02:11:18.999000"^^xsd:dateTime	"0.28568368707692304"^^xsd:double	"frontView"

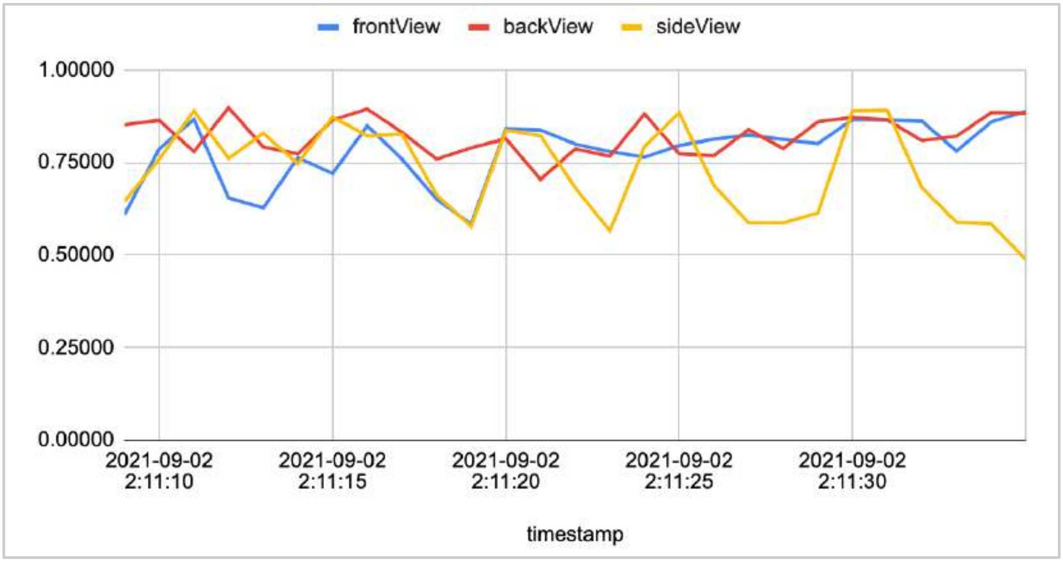
5. Retrieve Average Landmark Values for Upper Torso per Source

SPARQL Query



```
PREFIX : <http://cpsosaware.eu/ontology#>
PREFIX sosa: <http://www.w3.org/ns/sosa/>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>

SELECT ?timestamp (AVG(?result) AS ?avgResult)
WHERE {
    ?observation a :AnalysisOutput ;
                 sosa:observedProperty ?property ;
                 sosa:resultTime ?timestamp ;
                 sosa:hasSimpleResult ?result .
    ?property rdfs:label ?propLabel .
    ?s sosa:madeObservation ?observation ;
       rdfs:label "sideView" . # frontView backView sideView
    VALUES ?propLabel { "landmark_2" "landmark_3" "landmark_4"
    "landmark_5" "landmark_6" "landmark_7" "landmark_8" }
} GROUP BY ?timestamp ORDER BY ASC(?timestamp)
```

Result (for landmarks 2 to 8, corresponding to upper torso)

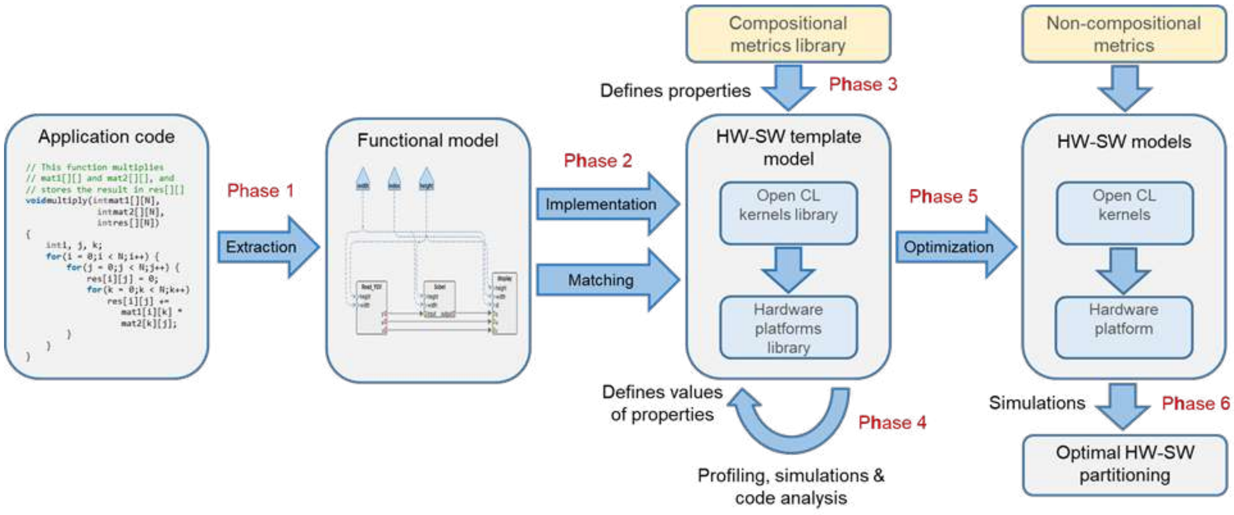


3 Model-Based design and re-design

In this section we explain CPSoSaware model-based design and redesign methodology, present DSM model, and show an application of CPSoSaware model-based design and redesign methodology to DSM model.

3.1 Model-based design and re-design methodology

CPSoSaware model-based design and re-design methodology is based on the HW-SW partitioning optimization approach developed in Task 4.1 and described in detail in deliverables D4.1 and D4.6. This approach contains several phases that are presented in Figure 28.



```
// This function multiplies
// mat1[][] and mat2[][], and
// stores the result in res[][]
voidmultiply(intmat1[][N],
             intmat2[][N],
             intres[][N])
{
    inti, j, k;
    for(i = 0;i < N;i++) {
        for(j = 0;j < N;j++) {
            res[i][j] = 0;
            for(k = 0;k < N;k++)
                res[i][j] +=
                  mat1[i][k] *
                  mat2[k][j];
        }
    }
}
```

Figure 28: HW-SW partitioning optimization approach

In the context of Task 4.6 we extended simulation methods (Phase 4 and Phase 6 of partitioning optimization approach) to handle various high-level system and SoS metrics such as the accuracy of AI models implemented as corresponding application code in multiple scenarios. We also extend optimization methods (Phase 5) to support multi-scenario optimization that enables the re-design phase. Multi-scenario optimization allows us to determine a set of system models that are optimal in different scenarios under different situational goals/objectives that also may change from scenario to scenario. The support of multiple scenarios is also required in the modelling phase, so that the concise modelling methodology described in D4.1, D4.6 is extended to support scenario definitions on the functional level. This support is enabled by applying stereotype <scenario> to the functional block and the part used for the definition of different scenarios. Once <scenario> part is present in the functional model, the concise modelling plugin automatically translates it to a multi-scenario optimization model and runs optimization sequentially for each scenario with respect to the situational goal specific for the scenario. The obtained optimal solutions are then translated back to collections of Pareto optimal models, where each collection relates to the corresponding scenario. Finally, optimal models are analysed by an expert and a re-design strategy is created. This strategy can be presented as a set of rules, where each rule defines the conditions when re-design decision should be triggered by CPSoS and which components should be commissioned/decommissioned in the re-design phase to enable smooth adaptation of CPSoS to changes in the environmental conditions. Resulting rules could be deployed to target CPSoS as part of AI component such as Knowledge Graph as described in the Section 2.2 above. AI component will determine when environmental conditions change to meet the definitions of another scenario and then trigger the commissioning/decommissioning mechanism described in detail in D4.2/D4.3 to re-design target SoS and adopt it to the changing environment.

3.2 DSM model

In this section we present initial evaluation of driver monitoring system model (DSM) performed in Task 4.4 and 5.1 and reported in deliverable D4.6, D5.1. The DSM solution uses a camera installed inside a car to monitor the face of the driver. Each frame of this camera is analysed, and facial landmarks are aligned. The position of these landmarks and their distance can be used to recognize whether the driver is yawning, or if his eyes are sleepy. Moreover, the pose of his head can also reveal information about the driver's drowsiness level. The 2D facial landmark alignment method used in the algorithm is implemented with C++ in the open-source libraries DLIB and Deformable Shape Tracking (DEST), is used in several applications such as driver drowsiness detection, recognition of facial expressions, etc. The most challenging of these applications require fast processing of video frames. Therefore, the alignment of the facial landmarks in a single video frame has to be performed with the minimum possible latency without precision loss. The fast 2D facial landmark detection algorithm that has been presented by Kazemi and Sullivan where an Ensemble of Regression Trees (ERT) is used to estimate the position of the facial landmarks, has been adopted in the DSM runnables. The algorithms and the overall experiments on how to correctly design the runnables are presented in D4.6 and particular hardware and software runnables that have been developed based on OpenCL and/or HLS C/C++ code using High level Synthesis tools (for hardware) or OpenCL compilers (for software) presented in D5.1. The parameters of the ERT face alignment model are described in D4.6 in detail. The ones that affect the implementation of the HW kernels are listed in Table 5. From the description of the ERT parameters in this table it is clear that a trade-off has to be made between accuracy and speed. Additionally, one should consider different conditions that affect accuracy of different ERT models as well as power consumption of corresponding HW kernels, that may affect other application and/or need to be considered in battery-low regimes.

Table 5: ERT parameters customized for the DSM application

Parameter Name	Default value	Experimentation in the range	Description
Cascade stages (Tcs)	10	9-12	Fewer cascade stages result in faster shape estimation but with lower accuracy. However error floor prevents accuracy improvement if excessive cascade stages are added
Trees (Ntr)	500	400-600	Fewer trees increase speed but reduce accuracy
Tree depth (Td)	5	4-6	Number of tree nodes is 2Td-1. Shorter tree depth (and consequently tree nodes) results in faster operation but with lower accuracy
Reference pixels (Nc)	600	400-800	This is the number of pixels that the sparse image consists of. Fewer reference pixels are expected to increase speed with accuracy penalty

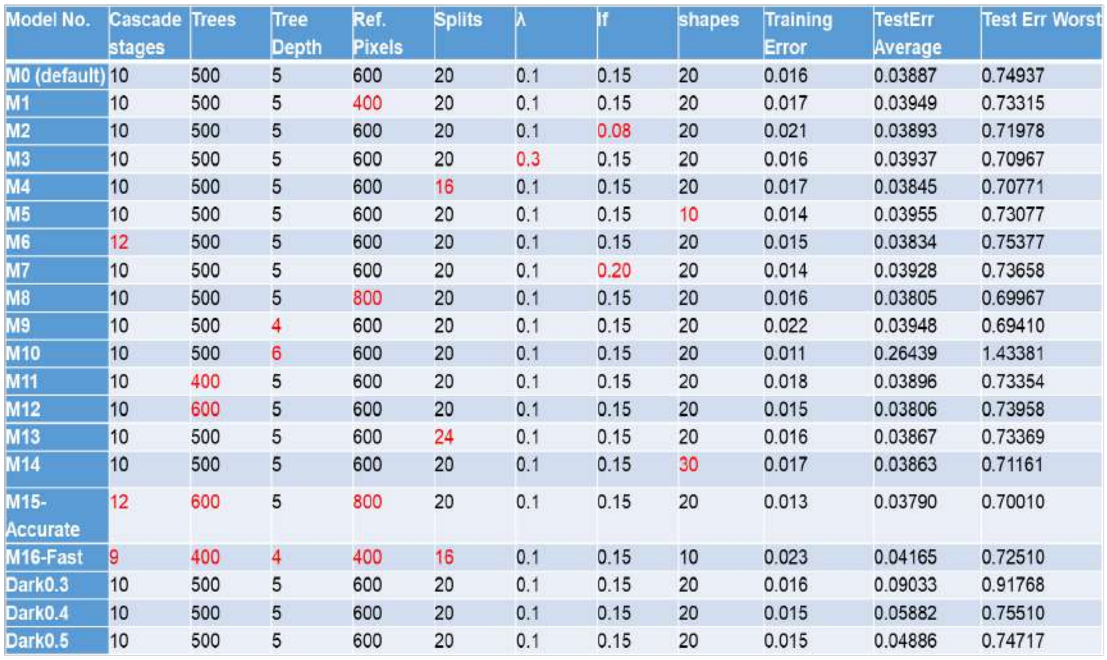
3.2.1 ERT models developed

Various models have been trained with the same Helen general purpose dataset. Table 6 shows the combinations tested and the training error exposed by the DEST training application that has also been ported to Ubuntu. A different application is used to evaluate a model using a test set of 300 photos, different from the Helen dataset. The details of these models are discussed in D4.6.

Models M15 and M16 have been defined as combinations of various parameter values for the best accuracy and the highest speed, respectively. Since we did not have a dataset available with driver images and especially in nighttime lighting conditions, we defined three models called Dark0.3, Dark0.4 and Dark0.5 that have been trained from the Helen images again but after artificially darkening them during the training process.

The ERT parameters with the default values were used but a dynamic range adaptation has been applied in grayscale to compress the pixel values to the 30%, 40% or 50% of their original range in the models Dark0.3, Dark0.4 and Dark0.5, respectively. For example, if the pixel intensity is initially between 0 and 100, this intensity is linearly adapted to shorten the range between 0 and 30 in the Dark0.3 model. It is obvious that Dark0.3 has been trained with darker images while Dark0.5 is closer to the M0 model. The test error is worse than the other models. However, in a real car environment and nighttime drive it is expected that the behavior of the Dark0.* models may be better than several other models listed in Table 6.

Table 6: ERT models used, based on different ERT parameters



Model No.	Cascade stages	Trees	Tree Depth	Ref. Pixels	Splits	λ	lf	shapes	Training Error	TestErr Average	Test Err Worst
M0 (default)	10	500	5	600	20	0.1	0.15	20	0.016	0.03887	0.74937
M1	10	500	5	400	20	0.1	0.15	20	0.017	0.03949	0.73315
M2	10	500	5	600	20	0.1	0.08	20	0.021	0.03893	0.71978
M3	10	500	5	600	20	0.3	0.15	20	0.016	0.03937	0.70967
M4	10	500	5	600	16	0.1	0.15	20	0.017	0.03845	0.70771
M5	10	500	5	600	20	0.1	0.15	10	0.014	0.03955	0.73077
M6	12	500	5	600	20	0.1	0.15	20	0.015	0.03834	0.75377
M7	10	500	5	600	20	0.1	0.20	20	0.014	0.03928	0.73658
M8	10	500	5	800	20	0.1	0.15	20	0.016	0.03805	0.69967
M9	10	500	4	600	20	0.1	0.15	20	0.022	0.03948	0.69410
M10	10	500	6	600	20	0.1	0.15	20	0.011	0.26439	1.43381
M11	10	400	5	600	20	0.1	0.15	20	0.018	0.03896	0.73354
M12	10	600	5	600	20	0.1	0.15	20	0.015	0.03806	0.73958
M13	10	500	5	600	24	0.1	0.15	20	0.016	0.03867	0.73369
M14	10	500	5	600	20	0.1	0.15	30	0.017	0.03863	0.71161
M15-Accurate	12	600	5	800	20	0.1	0.15	20	0.013	0.03790	0.70010
M16-Fast	9	400	4	400	16	0.1	0.15	10	0.023	0.04165	0.72510
Dark0.3	10	500	5	600	20	0.1	0.15	20	0.016	0.09033	0.91768
Dark0.4	10	500	5	600	20	0.1	0.15	20	0.015	0.05882	0.75510
Dark0.5	10	500	5	600	20	0.1	0.15	20	0.015	0.04886	0.74717

3.2.2 Accuracy of the models

Exhaustive experiments are described in D4.6 concerning the accuracy of the models listed in Table 7 under different environmental conditions. Table 7 lists the conclusions about which models are more appropriate for certain combinations of driver gender/lighting conditions/mount position of the camera. The F1-score has been used to sort the model accuracy since this metric is a combination of precision and sensitivity. The male and female drivers in daytime conditions have been evaluated with the YawDD dataset (camera mounted on the mirror or the dash). The nighttime is evaluated for the present, only with the 7 videos that we have developed. More exhaustive tests will be performed when the NYSEM dataset will be available.

Table 7: Top-3 models with the highest accuracy in yawning measurement.

Condition	Model
Male-Daytime-Dash	M16, M8, M15
Male-Daytime-Mirror	Dark0.3, M8, Dark0.5
Female-Daytime-Dash	M12, M15, M16
Female-Daytime-Mirror	Dark0.3, M8, Dark0.5
Male-Nighttime (from Table 5)	M8, M16, M15

3.2.3 Latency and power consumption

The estimated power consumption of the models referenced in Table 7 is analyzed in Table 8 which shows the dynamic power consumption and in Table 9. Since we are interested in the power consumption of the kernels that are implemented in the PL system we get the total power dissipation, static and dynamic, by adding all the power estimations for the PL part and the results are listed in Table 10. As we can see the power consumption ranges between 2.665W (M16) and 2.905W (M15).

Regarding the latency of each HW kernel implementation and its effect on the overall processing time needed by a single frame, the results are listed in Table 11. In the first row of Table 11 the latency of a single frame is shown, as it is measured from the software side with appropriate print messages. In the second row the HW kernel latency is listed as it is profiled using the XRT real time monitoring facilities that generate a runtime csv file (profile_summary.csv). Since it was not possible to migrate data pixel intensities using wide port and local BRAM in the same way we implemented migrate of the other HW kernel arguments, its latency was measured separately as shown in the last row of Table 11.

Table 8: Dynamic power consumption of the HW kernels of the models referenced in Table 7.

Model/ Power	M0, M4, Dark0.3, Dark0.4, Dark0.5	M8	M12	M15	M16
Clocks	0.496W (10%)	0.498W (10%)	0.515W (11%)	0.515W (10%)	0.461W (10%)
Signals	0.488W (10%)	0.510W (11%)	0.540W (11%)	0.561W (11%)	0.454W (10%)
Logic	0.369W (8%)	0.374W (8%)	0.399W (8%)	0.401W (8%)	0.350W (7%)
BRAM	0.585W (12%)	0.584W (12%)	0.586W (12%)	0.604W (12%)	0.576W (12%)
DSP	0.088W (2%)	0.089W (2%)	0.092W (2%)	0.088W (2%)	0.089W (2%)
MMCM	0.097W (2%)	0.097W (2%)	0.097W (2%)	0.097W (2%)	0.097W (2%)
PS	2.659W (56%)	2.659W (55%)	2.659W (54%)	2.659W (55%)	2.659W (57%)

Table 9: Static power consumption of the HW kernels of the models referenced in Table 7

Model/ Power	M0, M4, Dark0.3, Dark0.4, Dark0.5	M8	M12	M15	M16
PL	0.638W (86%)	0.638W (86%)	0.639W (86%)	0.639W (86%)	0.638W (86%)
PS	0.100W (14%)	0.100W (14%)	0.100W (14%)	0.100W (14%)	0.100W (14%)

Table 10: Total power dissipation of the PL part

Model/ Power	M0, M4, Dark0.3, Dark0.4, Dark0.5	M8	M12	M15	M16
PL	2.761W	2.79W	2.868W	2.905W	2.665W

Table 11: Estimation of the single frame processing latency

Model/Latencies	M0 model	M4 Model	M8 Model	M12 Model	M15 Model	M16 Model	Dark0.3 Model	Dark0.4 Model	Dark0.5 Model	Notes
Frame processing time	33.166 ms	34.301 ms	34.658 ms	38.629 ms	47.756 ms	29.314 ms	33.816 ms	37.717 ms	33.371 ms	1st Frame
HW kernel Predict_Trees (XRT profile_summary.csv)	2.33 ms	2.42 ms	2.37 ms	2.86 ms	2.90 ms	2.09 ms	2.26 ms	2.26 ms	2.40 ms	Worst Case (1st frame)
Migrate pixel intensities (XRT profile_summary.csv)	0.076 ms	0.149 ms	0.160 ms	0.151 ms	0.153 ms	0.145 ms	0.149 ms	0.149 ms	0.114 ms	Worst Case (1st Frame)
Maximum supported frame rate	30 fps	29 fps	28 fps	25 fps	20 fps	34 fps	29 fps	26 fps	29 fps	

3.3 Evaluation of model-based design and re-design methodology

3.3.1 System model and input data

DSM application high-level system model is presented in Figure 29. This model is created using concise modelling extension of SysML[28] described in detail in D4.1 and D4.6. The model consists of an Application block and a Software Block. The Application block includes itsScenario and itsModel part that used to describe different scenarios and different ERT models correspondingly. The itsScenario part has attributes

[28] https://www.omg.org/spec/SysML

that characterize particular scenario such as gender of the driver (male/female), camera positions (dash/mirror) and light conditions (daytime /nighttime). In order to handle this part properly in the automatic model a translation process is performed by the concise plugin we apply <scenario> stereotype to this part. In total we have 8 different scenarios with different Id's, that provided in excel table that is a part of concise model.

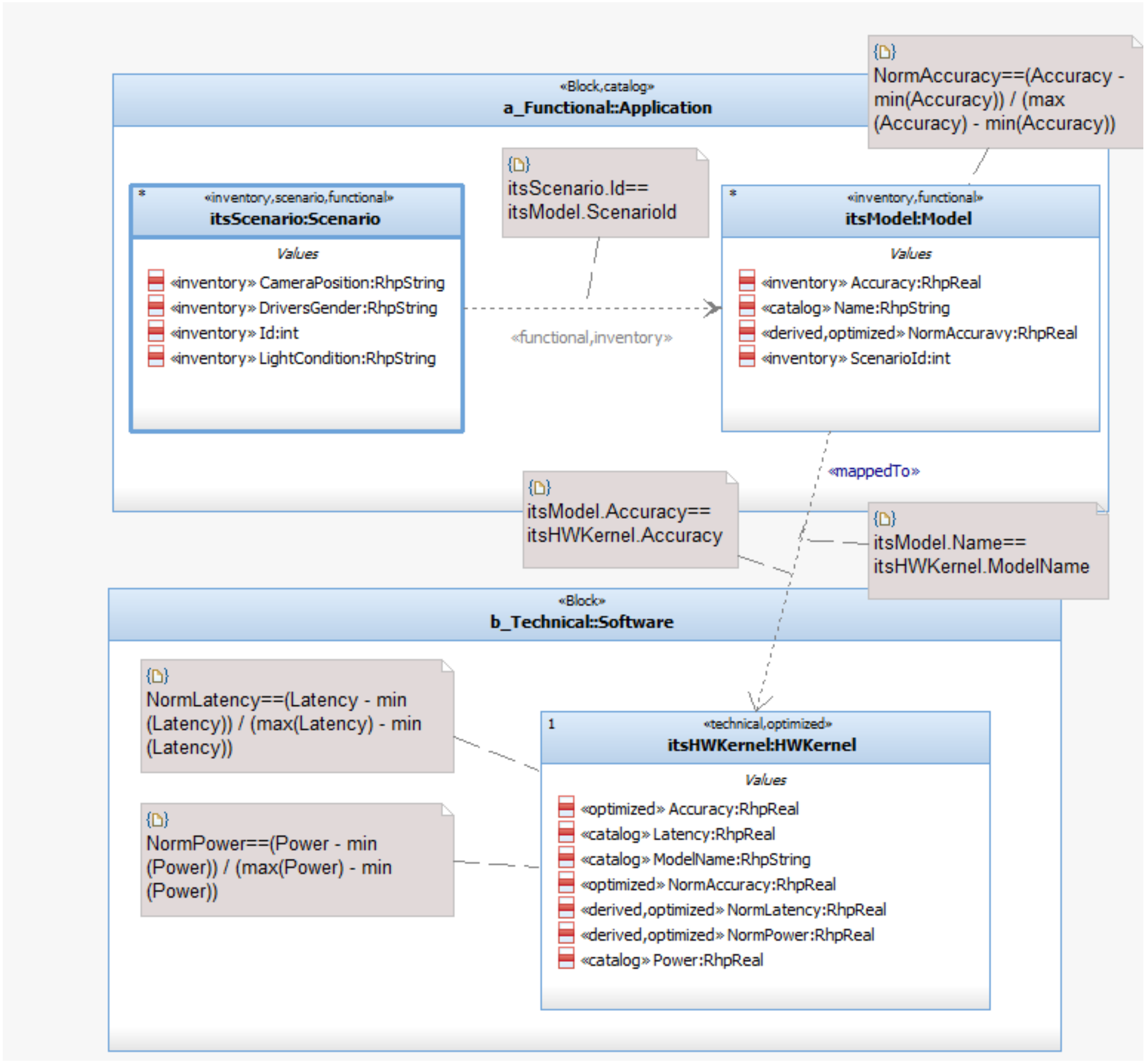
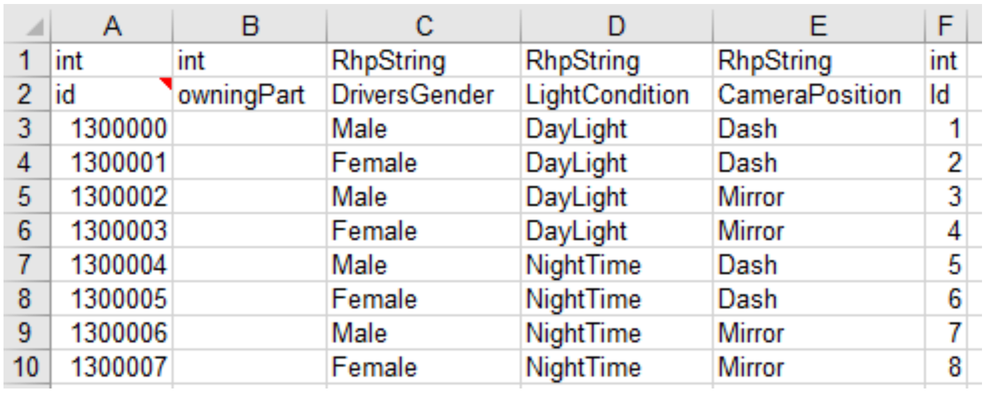

«Block,catalog»
a_Functional::Application
NormAccuracy==(Accuracy - min(Accuracy)) / (max (Accuracy) - min(Accuracy))
«inventory,scenario,functional»
itsScenario:Scenario
Values
«inventory» CameraPosition:RhpString
«inventory» DriversGender:RhpString
«inventory» Id:int
«inventory» LightCondition:RhpString
itsScenario.Id== itsModel.ScenarioId
«functional,inventory»
«inventory,functional»
itsModel:Model
Values
«inventory» Accuracy:RhpReal
«catalog» Name:RhpString
«derived,optimized» NormAccuravy:RhpReal
«inventory» ScenarioId:int
«mappedTo»
itsModel.Accuracy== itsHWKernel.Accuracy
itsModel.Name== itsHWKernel.ModelName
«Block»
b_Technical::Software
NormLatency==(Latency - min (Latency)) / (max(Latency) - min (Latency))
NormPower==(Power - min (Power)) / (max(Power) - min (Power))
1
«technical,optimized»
itsHWKernel:HWKernel
Values
«optimized» Accuracy:RhpReal
«catalog» Latency:RhpReal
«catalog» ModelName:RhpString
«optimized» NormAccuracy:RhpReal
«derived,optimized» NormLatency:RhpReal
«derived,optimized» NormPower:RhpReal
«catalog» Power:RhpReal

Figure 29: Concise model of DSM application - design

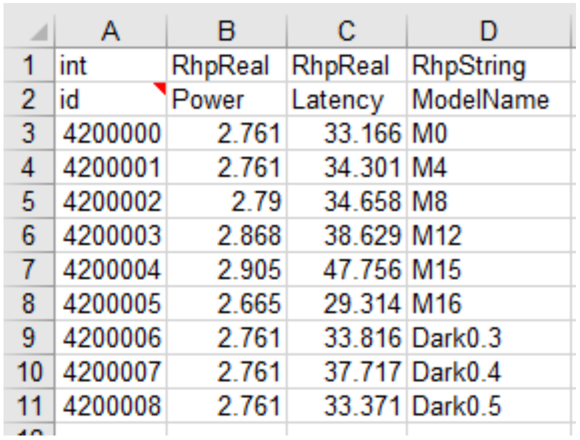
Excel table with different scenarios presented in Figure 30. The itsModel part contains model name, scenario Id and Accuracy attributes. This part has an Excel table with corresponding data behind, where the Accuracy attribute represents an F1-score of corresponding models measured in the corresponding scenario (note that measurement performed not for all possible scenarios). The itsScenario and itsModel parts are connected by functional flow with attached constraint, meaning that the Scenario Id attribute of the model should be equal to the Id attribute of Scenario.

The Software block contains itsHWKernel part that represents our decision on optimal model. <optimized> stereotype on itsHWKernel part means that particular hardware kernel should be chosen by the optimization and multiplicity of itsHWKernel part that equal to 1 means that only one hardware kernel should be chosen in each particular scenario. The part is chosen from catalog that contains data on Latency (equivalent to the frame processing time in Table 11) and power consumption (see Table 10) of different ERT models. The Excel table representing this catalog is a part of the concise model and presented in Figure 31.



	A	B	C	D	E	F
1	int	int	RhpString	RhpString	RhpString	int
2	id	owningPart	DriversGender	LightCondition	CameraPosition	Id
3	1300000		Male	DayLight	Dash	1
4	1300001		Female	DayLight	Dash	2
5	1300002		Male	DayLight	Mirror	3
6	1300003		Female	DayLight	Mirror	4
7	1300004		Male	NightTime	Dash	5
8	1300005		Female	NightTime	Dash	6
9	1300006		Male	NightTime	Mirror	7
10	1300007		Female	NightTime	Mirror	8

Figure 30: Concise model of DSM application – scenario table



	A	B	C	D
1	int	RhpReal	RhpReal	RhpString
2	id	Power	Latency	ModelName
3	4200000	2.761	33.166	M0
4	4200001	2.761	34.301	M4
5	4200002	2.79	34.658	M8
6	4200003	2.868	38.629	M12
7	4200004	2.905	47.756	M15
8	4200005	2.665	29.314	M16
9	4200006	2.761	33.816	Dark0.3
10	4200007	2.761	37.717	Dark0.4
11	4200008	2.761	33.371	Dark0.5

Figure 31: Concise model of DSM application – hardware kernels catalog

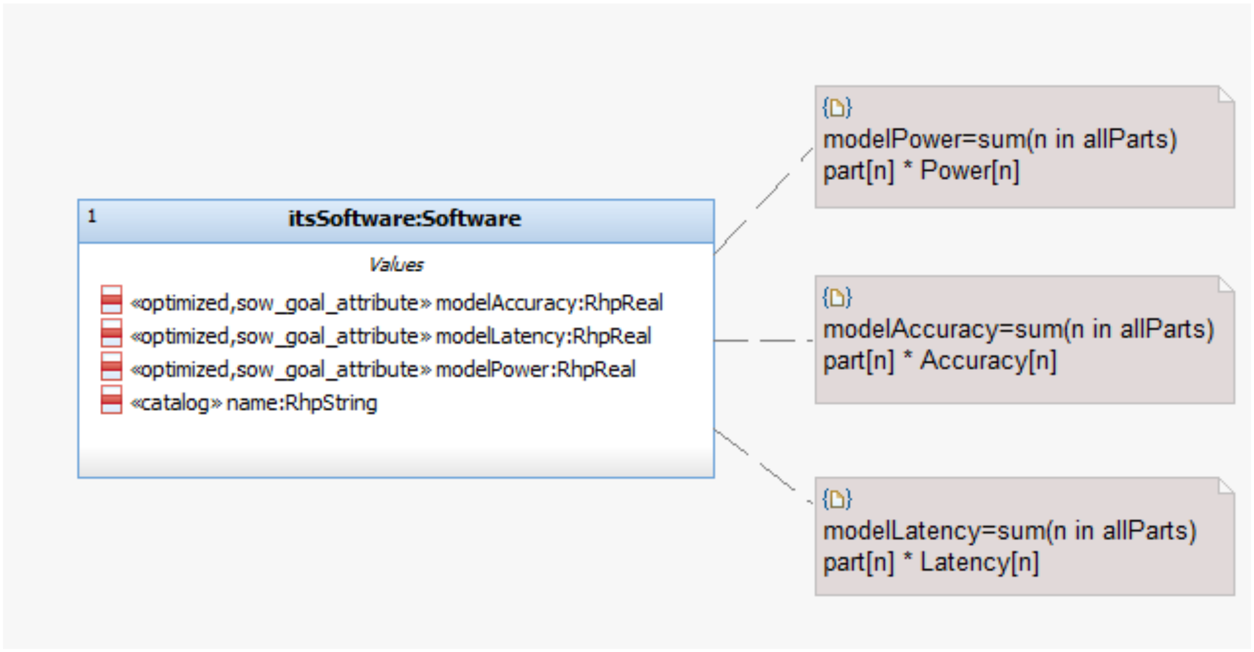
ItsModel part of the Functional block mapped to itsHWKernel part of the software block, meaning that the hardware kernel performs functions of the corresponding model. Constraints on the mapping required to ensure equality of the corresponding attributes in model and kernel.

To neutralize the effect of the unit's measurements on the optimization model, we need to normalize all the parameters of each ERT model and set them in the interval [0,1]. That way we fairly can priories the different criterions with weights, without being affected from higher values related with one of the criteria. In order to do this, we added additional attributes and constraints to the concise model. Attributes NormAccuracy of ItsModel part and attributes NormPower and NormLatency of itsHWKernel part are automatically calculated by formulas provided via corresponding constraints that are anchored to these parts (see Figure 29). The following table describe the normalized power and latency parameters.

Table 12: Normalized latency and power consumption

Model/Latencies	M0	M4	M8	M12	M15	M16	Dark0.3	Dark0.4	Dark0.5
Power	0.0330	0.0330	0.0430	0.0698	0.08261	0	0.0330	0.0330	0.0330
Latency		0.1044	0.1119	0.195	0.3861	0	0.0942	0.1759	0.0849

Besides the accuracy that measures the quality of the model, each of the models can be evaluated according to two other goals that are power consumption and latency. Optimization goals are modelled as Software block attributes with corresponding constraints that connects these attributes to the corresponding attributes of itsHWKernel part. Corresponding model presented in Figure 32.



1
itsSoftware:Software
Values
«optimized,sow_goal_attribute» modelAccuracy:RhpReal
«optimized,sow_goal_attribute» modelLatency:RhpReal
«optimized,sow_goal_attribute» modelPower:RhpReal
«catalog» name:RhpString
modelPower=sum(n in allParts) part[n] * Power[n]
modelAccuracy=sum(n in allParts) part[n] * Accuracy[n]
modelLatency=sum(n in allParts) part[n] * Latency[n]

Figure 32: Concise model of DSM application – optimization goals

3.3.2 Optimization model

Based on high-level system model presented in Figure 29, concise application automatically creates a multi-scenario multi-objective optimization model that considers all three different goals (see Figure 32) recognized with each model according to the relevancy and importance we allocate to each goal. The more importance we choose to allocate to one goal, the better the goal our optimization model will choose in the sense of that goal; for example if power consumption has a higher priority, a more energy-saving hardware kernel will be chosen. In addition, we included in the optimization model strict-bounds option (optimization constraints), that allows choosing the best model from only those models with parameters that perform better than the fixed bounds. For example, for each scenario we need to choose an optimal model from the models where the accuracy has a value above some threshold, otherwise DSM application does not fulfill its function and such design is meaningless.

Below we present mathematical optimization model that automatically created from the concise model. We wish to find the optimal model/kernel according to three goals: accuracy, power consumption and latency (we refer them here as goal 1, goal 2 and goal 3 respectively, so that goals are enumerated by $j = 1,2,3$). We have $k = 1, \dots, K$ scenarios and our decision variables are $x_i \in \{0,1\}$ for each model/kernel $i = 1, \dots, I$. Thus, our optimization functions for second and third goal (power consumption and latency) could be modelled by the following formula:

$$min_x\, f_j(x) = \sum_{i=1}^{I} t_i^j x_i$$

where by t_i^j we denote the normalized value of the goal $j \in \{2,3\}$ for the i'th model/kernel. In the case of the first goal, we are interested in model/kernel hat maximize the accuracy. Hence the appropriate formula is

$$max_x\, f_j(x) = \sum_{i=1}^{I} t_{i,k}^1 x_i$$

where by $t_{i,k}^1$ we denote the normalized value of the first goal (accuracy) for the i'th model/kernel in the scenario k. Alternatively, denoting by $t_{i,k}^1$ the **minus** of the normalized value of the first criterion for the i'th model/kernel and scenario, our objective functions turns into a minimization of the form

$$min_x\, f_j(x) = \sum_{i=1}^{n} t_{i,k}^j x_i \text{ , for all } j = 1, \dots, 3.$$

Note that corresponding formulas in the OPL optimization language[29] are create automatically from constraints presented in Figure 32.

In order to get only one optimal model/kernel that fits our optimization goals for each specific scenario we optimization model includes constraint $\sum_{i=1}^{n} x_i = 1$ that automatically created from the multiplicity of itsHWKernel part. These constraints ensures that optimization will choose the model i so that its appropriate decision variable x_i will be equal to 1 in the optimal solution. In addition, we add a set of constraints to ensure that the optimized model/kernel will satisfy our fixed bounds:

$$\begin{cases} \sum_{i=1}^{I} s_{i,k}^j x_i \geq b^l{}_j \\ \sum_{i=1}^{I} s_{i,k}^j x_i \leq b^u{}_j \end{cases} \text{for } j = 1, \dots, 3,$$

where by $b^l{}_j, b^u{}_j$, we denote the fixed (lower and upper) bounds to which our model should satisfy and by $s_{i,k}^j$ we denote the real (not normalized) value of the parameter j for the i'th model/kernel in the scenario k. The sign of inequality depends by the type of bound (upper or lower) that we would like to apply to the system. In our case these means that we would like to have a model with accuracy that above some threshold and latency and power consumption that is below some threshold.

To combine multi-objective optimization model into single-objective ILP (Integer Linear Programming) problem concise application add weight parameters to each optimization objective that define different priorities which optimization allocates to the different goals in order to find an optimized model/kernel that considers those priorities. This could be presented by the following formula

$$f(x) = \sum_{j=1}^{m} (w_j \sum_{i=1}^{n} t_{i,k}^j x_i),$$

where w is the relative weights vector, representing how much weight optimization allocates to each of the objectives. For the resulting single-objective ILP concise application further adds constraint $k = k'$

[29] https://www.ibm.com/docs/en/icos/22.1.0?topic=opl-optimization-programming-language

where k' is a number of scenarios for which optimization should find optimal model/kernel. Resulting single-scenario single-objective ILP solved by CPLEX optimization solver[30] for each scenario and for the different combinations of weights for the different objectives.

3.3.3 Results

Following tables summarize the optimization problems we solved for all the scenarios (different driver identity and camera position) using our model. We used different choices of weights to find the optimize model under different conditions and priorities of our three main criterions (F1-score, power consumption and latency). The following tables summarize different types of optimization problems that we solved for the various scenarios (different driver identity, day period and camera position) using our model. We used two choices of weights. The first one emphasis the quality of the model over considerations of latency and energy consumption. These settings should be considered when more frequent frame rate environment is in use since high latency is balanced by the high frequent sampling.

The other choice of weights allocates an equal attention to all goals and achieved that by setting the weight of each goal to 0.33. These settings should be considered when low frequent frame rate environment is in use, to prevent long waiting for the system feedback.

Table 13: Optimization results - Male-Dash Scenario

Frame Processing Rate	Weights	Bounds	Best Model
Rate=2	(0.9, 0.05, 0.05)	(0, 10, 40)	Model 16
Rate=2	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16
Rate=10	(0.9, 0.05, 0.05)	(0, 10, 40)	Model 4
Rate=10	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16

Table 14: Optimization results - Male-Mirror Scenario

Frame Processing Rate	Weights	Bounds	Best Model
Rate=2	(0.9, 0.05, 0.05)	(0, 10, 40)	Model Dark0.4
Rate=2	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16
Rate=10	(0.9, 0.05, 0.05)	(0, 10, 40)	Model Dark0.3
Rate=10	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16

Table 15: Optimization results - Female-Dash Scenario

Frame Processing Rate	Weights	Bounds	Best Model
Rate=2	(0.9, 0.05, 0.05)	(0, 10, 40)	Model 16
Rate=2	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16
Rate=10	(0.9, 0.05, 0.05)	(0, 10, 40)	Model 4
Rate=10	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16

[30] https://www.ibm.com/analytics/cplex-optimizer

Table 16: Optimization results - Female-Mirror Scenario

Frame Processing Rate	Weights	Bounds	Best Model
Rate=2	(0.9, 0.05, 0.05)	(0, 10, 40)	Model Dark0.4
Rate=2	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16
Rate=10	(0.9, 0.05, 0.05)	(0, 10, 40)	Model Dark0.3
Rate=10	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16

Table 17: Optimization results - Night-time Scenario

Frame Processing Rate	Weights	Bounds	Best Model
Rate=2	(0.9, 0.05, 0.05)	(0, 10, 40)	Model 16
Rate=2	(0.33, 0.33, 0.33)	(0, 10, 40)	Model 16

According to the results presented above the best model/kernel for most scenarios is Model16, however model re-design required in a case of high accuracy requirements in some scenarios. Alternative models are:

- Model 4 in daytime dash scenarios when processing rate is low (Rate=10)
- Model Dark0.4 in daytime mirror scenario with fast processing rate (Rate=2)
- Model Dark0.3 in daytime mirror scenario with slow processing rate (Rate=10)

The optimal model for all case does not depends on gender of the driver (male/female).

4 Conclusions

In this document we presented CPSoSaware AI framework and model-based design and re-design methodology. In the context of AI framework, we performed multimodal odometer fusion evaluation studies that based on realizes sensor cooperation approach that is important for improving two major tasks of autonomous driving: vehicle odometry or SLAM and scene analysis and understanding. The sensor cooperation indeed necessary in order to maximize overall performance of the vehicle. Furthermore, a vertical fusion strategy has been developed for each one of the associated tasks, which integrates in a combined estimation framework the data from "selfish nodes", i.e., outputs of individual sensors, and provides more accurate pose information as well as object detection. This methodology evaluated on the simulated scenarios provided by CARLA-ROS framework and the evaluation results shows it importance in the context for the autonomous driving use-case. Moreover, the approach of "early" fusion of selfish sensor agents extended by a vertical AI mechanism based on a novel semantic data integration. In this case the analysis outputs, which are not the raw sensor measurements as previously, are fed to an ontology-based semantic Knowledge Graph (which defines the set of incentives) through the flexible semantic data integration framework of CASPAR. This vertical AI mechanism validated both in automotive and manufacturing use-cases.

CPSoSaware model-based design and re-design methodology based on HW-SW partitioning optimization approach that developed in Task 4.1. To enable re-design capabilities model-based optimization approach extended by additional simulation and optimization capabilities in order to support multi-objective multi-scenario optimization, that is crucial for the re-design phase. Developed methodology validated on the driver state monitoring application and set of optimal models for different scenarios are provided. The results are presents re-design strategy w.r.t different scenarios and different importance of objectives (accuracy/latency/power) and provides an incentive for development of CPSoS adaptation strategy that will use commissioning/decommissioning mechanism to adopt DSM application to the different environmental conditions.