energies

Article

Towards Cooperative Perception Services for ITS: Digital Twin in the Automotive Edge Cloud

Viktor Tihanyi, András Rövid, Viktor Remeli, Zsolt Vincze, Mihály Csonthó, Zsombor Pethő, Mátyás Szalai, Balázs Varga, Aws Khalil and Zsolt Szalay *

Department of Automotive Technologies, Faculty of Transportation Engineering and Vehicle Engineering, Budapest University of Technology and Economics, 1111 Budapest, Hungary; tihanyi.viktor@kjk.bme.hu (V.T.); rovid.andras@kjk.bme.hu (A.R.); remeli.viktor@kjk.bme.hu (V.R.); vincze.zsolt@kjk.bme.hu (Z.V.); csontho.mihaly@kjk.bme.hu (M.C.); petho.zsombor@kjk.bme.hu (Z.P.); szalai.matyas@kjk.bme.hu (M.S.); varga.balazs@kjk.bme.hu (B.V.); aws.khalil@bme.edu.hu (A.K.)

* Correspondence: szalay.zsolt@kjk.bme.hu; Tel.: +36-1-463-3226

Abstract: We demonstrate a working functional prototype of a cooperative perception system that maintains a real-time digital twin of the traffic environment, providing a more accurate and more reliable model than any of the participant subsystems—in this case, smart vehicles and infrastructure stations—would manage individually. The importance of such technology is that it can facilitate a spectrum of new derivative services, including cloud-assisted and cloud-controlled ADAS functions, dynamic map generation with analytics for traffic control and road infrastructure monitoring, a digital framework for operating vehicle testing grounds, logistics facilities, etc. In this paper, we constrain our discussion on the viability of the core concept and implement a system that provides a single service: the live visualization of our digital twin in a 3D simulation, which instantly and reliably matches the state of the real-world environment and showcases the advantages of real-time fusion of sensory data from various traffic participants. We envision this prototype system as part of a larger network of local information processing and integration nodes, i.e., the logically centralized digital twin is maintained in a physically distributed edge cloud.

Keywords: cooperative perception; ITS; digital twin; sensor fusion; edge cloud

check for updates

Citation: Tihanyi, V.; Rövid, A.; Remeli, V.; Vincze, Z.; Csonthó, M.; Pethő, Z.; Szalai, M.; Varga, B.; Khalil, A.; Szalay, Z. Towards Cooperative Perception Services for ITS: Digital Twin in the Automotive Edge Cloud. *Energies* **2021**, *14*, 5930. https://doi.org/10.3390/en14185930

Academic Editor: Marco Pau

Received: 13 July 2021
Accepted: 4 September 2021
Published: 18 September 2021

1. Introduction

1.1. Scope and Significance

The future of connected and automated vehicles (CAVs) and the development of intelligent transportation systems (ITSs) are actively researched topics which open up a multitude of possibilities. With the progress in computation and communication technology in the last decade, some formerly unrealistic constructs are becoming more practically viable and demand proof-of-concept implementations. We envision a future where traffic participants and observers like CAVs and ITSs share their information resources in real-time for a safer and more efficient transportation and traveling experience. Herein, we outline a so-called *Central System* architecture that enables such information sharing and integration. We use the term *central* in a strictly logical sense to denote the emergence of a single, fully integrated and logically consistent environment and decision model in the cloud (the *digital twin*), while the physical implementation itself remains highly *distributed*, i.e., computation and communication loads are delegated to a spatially localized edge processing nodes hierarchy, as well as a network of third-party partners such as trusted data and algorithm providers. Understanding the wide ranging general applicability of building a well-integrated smart road and vehicle IoT, we made a dedicated effort to design the *Central System* as an easily extensible integration framework using industry standard interfaces. In the coming months we expect to be able to connect a part of the fast-developing Hungarian ITS facilities into the *Central System* and to provide non-stop real-time integration of real traffic data and thus demonstrate the first large-scale industrial application. In the current paper, we report the positive results of small-scale experiments conducted in late 2020.

Arguably, the most useful kind of information for traffic participants might very well be the establishment of a so-called *digital twin*: a dynamic and real-time model of the environment, which is the focus of our current work and provides the major information source to build other services upon. The functional prototype we built and present in this paper therefore realizes only the fundamental real-time environment perception function of the *Central System*, which for clarity we will call *Central Perception*—distinguishing it from an envisioned suite of other dependent functionalities like traffic analytics and planning, road infrastructure monitoring and management, cloud-based traffic control and vehicle control, specific applications on CAV testing grounds and logistic grounds, etc. Of course, such derivative services must be introduced in at least some detail to highlight the expected practical significance of our initial efforts. A schematic overview of the planned *Central System*, its participants and functionalities is represented in Figure 1.

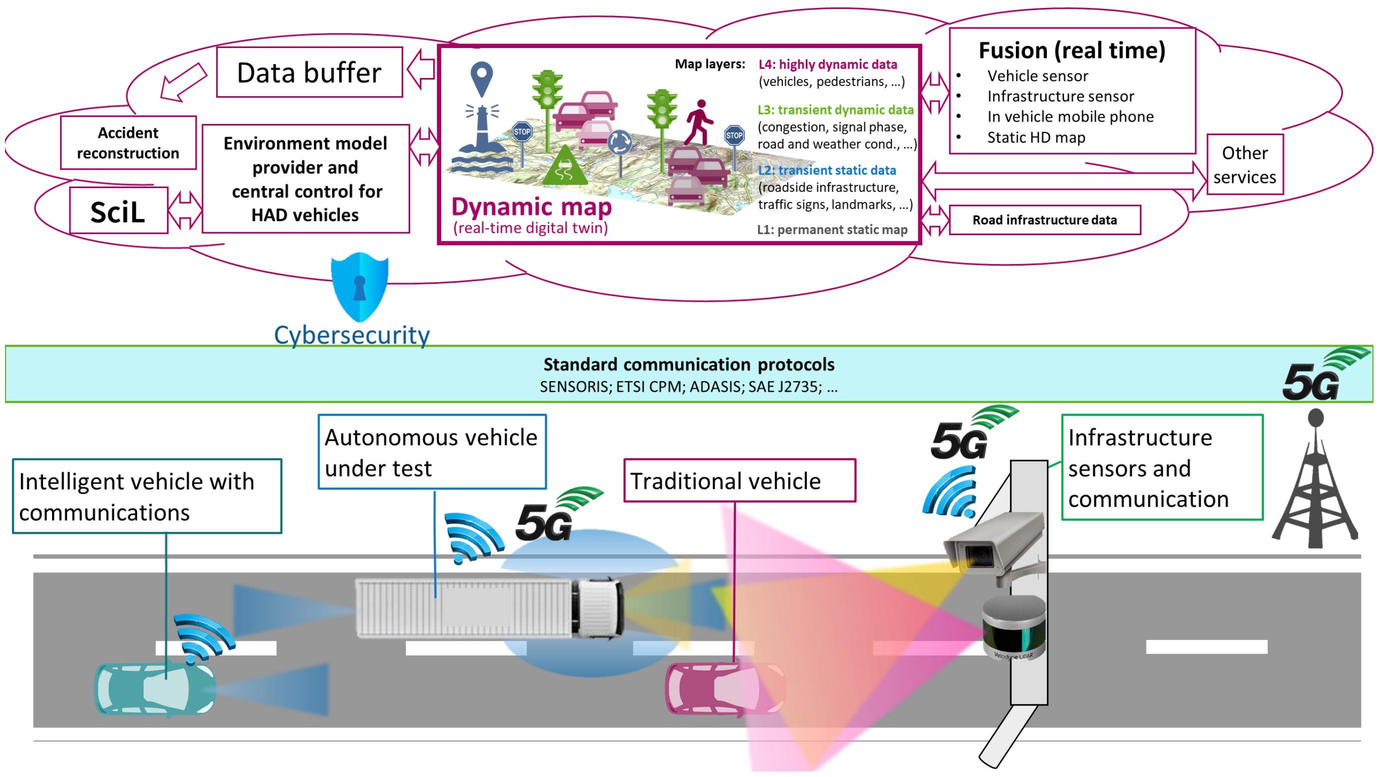
Data buffer
Accident reconstruction
Environment model provider and central control for HAD vehicles
SciL
Map layers:
L4: highly dynamic data (vehicles, pedestrians, ...)
L3: transient dynamic data (congestion, signal phase, road and weather cond., ...)
L2: transient static data (roadside infrastructure, traffic signs, landmarks, ...)
L1: permanent static map
Dynamic map (real-time digital twin)
Fusion (real time)
Vehicle sensor
Infrastructure sensor
In vehicle mobile phone
Static HD map
Other services
Road infrastructure data
Cybersecurity
Standard communication protocols
SENSORIS; ETSI CPM; ADASIS; SAE J2735; ...
5G
Intelligent vehicle with communications
Autonomous vehicle under test
Traditional vehicle
Infrastructure sensors and communication

Figure 1. *Central System* overview.

The premise of *Central Perception* is the following: assuming smart roads and vehicles in the near future, the same physical traffic environment will typically be perceived through many sensors and platforms of highly differing setups and capabilities at the same time, and these platforms may or may not be affiliated. Depending on application requirements like expected response time and reliability, the integration of environment perception information from various sources will pose certain challenges including communication, compatibility, synchronization, calibration, fusion, tracking, and end-to-end latency. Such an ad-hoc collective and collaborative perception system also requires certain sophistication in its architecture which must allow for scaling in a dynamically changing environment, especially considering CAVs that constantly change their location and whose sensor data must therefore be integrated with different stationary platforms at different points in time.

1.2. Prior Work

In the last two years, several connected automotive information system prototypes were realized, all of them demonstrating some degree of cooperative perception. Krämmer et al. from TU Munich and Fortiss built "Providentia" [1], an intelligent infrastructure system consisting of several gantries equipped with cameras and radars, serving as a support system for intelligent vehicles and aiding them in perceiving blind spots and objects behind cover. They process and fuse the sensory data both locally at the individual measurement stations and also centrally at designated edge computing nodes before communicating it to the consumer vehicles via 5G. Gabb et al. from Bosch and Karlsruhe showed theoretical guidelines (supported by experiments) for developing a similar system around the same time, at the Intelligent Vehicles conference in Paris [2]. The University of Tokyo released open source software for cooperative perception [3] while the University of California together with Toyota demonstrated a use-case of actually backpropagating the already integrated data from the cloud for driving assistance purposes [4]. Toyota also pursues similar topics with other Universities and independently, focusing on various settings like V2V communication [5] or camera and digital twin integration for visual guidance systems [6]. In Australia, researchers implemented the ETSI CPM messaging standard in an I2V setting, allowing sensor-less vehicles to perceive and autonomously react to pedestrians [7]. Recently, Chinese authorities announced to launch "world's first high-level cloud-controlled autonomous driving demonstration zone" (http://m.news.cctv.com/2020/09/11/ARTIeJEug9svYwuLazxQFzO3200911.shtml (accessed on 14 September 2021)) to be constructed in Beijing with similar long term targets as our *Central System* project.

As prior work we have carried out a measurement campaign (together with international industrial and academic partners) on a real-world motorway section in Hungary, which resulted sensory data useful for future automotive R&D activities due to the available ground truth for static as well as for dynamic content [8].

A possible first industrial application of *Central System* technology is likely to occur where experimental CAVs and ITSs are introduced earliest: on automotive testing grounds. In particular, development of *Central System* based Scenario-in-the-Loop (SciL) [9] control is already underway on the ZalaZONE [10] CAV proving ground in Hungary.

1.3. Primary Contribution

According to the best of our knowledge, we are the first to demonstrate a real-time cooperative perception platform that has both stationary and moving multi-sensor data sources and that combines several levels of data integration such as inter-sensor raw fusion, on-platform tracking and inter-platform local area fusion to finally create and visualize a simultaneous, centrally consistent model (*digital twin*) of all objects of interest in the area covered by the sensors' field of views.

We emphasize that our primary contribution lies in the demonstration of system-level possibilities as they were not demonstrated beforehand, i.e., the maintenance of a *digital twin* in a complex and heterogeneous environment. In this paper we do not claim any scientific novelty wrt. our individual subsystems, nor are they uniquely necessary for the development of our technological demonstration (we could have chosen alternative technological approaches to demonstrate the same concept).

To clarify, we define the following:

- **Digital twin:** a logically centralized, dynamic digital model of the traffic environment that integrates data from heterogeneous sources including both intelligent infrastructure and traffic participants in the cloud real-time.
- **Derivative services:** novel services that are expected to become available via a digital twin. These services fall into following major categories:
  - *Cooperative perception services:* more reliable centralized perception via sensor fusion in the cloud (*Central Perception*).

- *Dataset generation services:* more reliable perception enables more sophisticated scenario extraction and data referencing on multi-sensor datasets, as well as cross-validation of perception algorithms.
- *Accident reconstruction related services*
- *Cloud Control:* certain control functionalities (e.g., emergency braking) could be performed centrally based on cooperative perception inputs assuming sufficiently low latencies (a technological possibility in the near future).
- *Proving ground and logistical yard management:* some of the first potential areas of Central Perception and/or Cloud Control deployment.
- *Analytic services:* miscellaneous real-time datastream and historical data analytics.

2. Problem Definition

Arguably, the most crucial element in providing centrally maintained *digital twin* services is achieving a centrally integrated, real-time perception of the traffic environment. This paper presents a solution to the *Central Perception* problem that we specify as follows.

The overall system must acquire an integral and dynamic object-level view of the real-time traffic situation with at most 100 ms latency. The logical core of the system should be responsible for data integration while the peripheral measurement platforms will act as data sources. The overall architecture and the established interfaces must enable the simultaneous participation of connected intelligent equipment including static road-side infrastructure, mobile (vehicle-borne) and third-party measurement systems. Various sensor types and vendors must be supported, as well the precise and reliable detection of traffic participants including pedestrians, vehicles, obstacles, etc. As additional data sources, the system may use static HD maps and various traffic data (e.g., road meteorology) providers.

We specified and built a proof of concept system covering a substantial subset of above requirements, demonstrating the viability of the approach. Our functional sample provides following capabilities:

- a single central server (edge node);
- covering a spatially localized region of overlapping sensor field of views;
- fusing data from three multi-sensor camera-LiDAR platforms, one of them mobile;
- detecting (currently only) pedestrians;
- visualizing the digital twin in a realistic 3D simulation;
- in real-time (with less than 100 ms latency).

3. Overview of the Central Perception System Architecture

In order to support the requirements mentioned in Section 2, numerous design decisions had to be made regarding the prototype development of the measurement systems, the central server, and the communication between them. The following sections will discuss the overall design in some detail, the current section giving only a brief overview.

The realized system consists of one central server in the cloud and three wirelessly connected measurement systems, two of which are stationary and one that is mounted on a vehicle. At present we focus on fusion-based perception since we want to utilize the strengths of different sensor types on a single platform. Each measurement system has a sensory setup consisting of a camera-LiDAR pair, with the exception of the vehicle which has one camera and two LiDARs. For precise distance measurements and large-scale 3D reconstruction in automotive applications, stereo vision is becoming a less and less viable choice simply due to the precision loss at distances that are relevant to driving (compared to the high precision and falling prices of LiDAR technology) [11].

The measurement systems use the RTMaps software framework for data acquisition, synchronization, and data-flow processing. Detection and tracking is performed locally on the GPU-s and CPU-s of the measurement systems. The 3D pedestrian detection is done using low-level (raw) data fusion on the local system, i.e., the camera image and the corresponding LiDAR pointcloud are both necessary and are considered together in calculating the 3D position of the detected pedestrian (in contrast to object-level fusion where each sensor arrives at a detection estimate separately, and fusion occurs only afterwards). The fused detections of each sensor cluster are then tracked locally to smooth out any remaining uncertainties or missing datapoints. The tracks are then communicated using standard SENSORIS message formats over 5G or DSRC to the central server, where the inter-systems track fusion occurs. Finally the resulting locally and globally fused tracks are displayed real-time in a digital twin simulation developed in Unity. The overview of the system components and connections is shown in Figure 2.

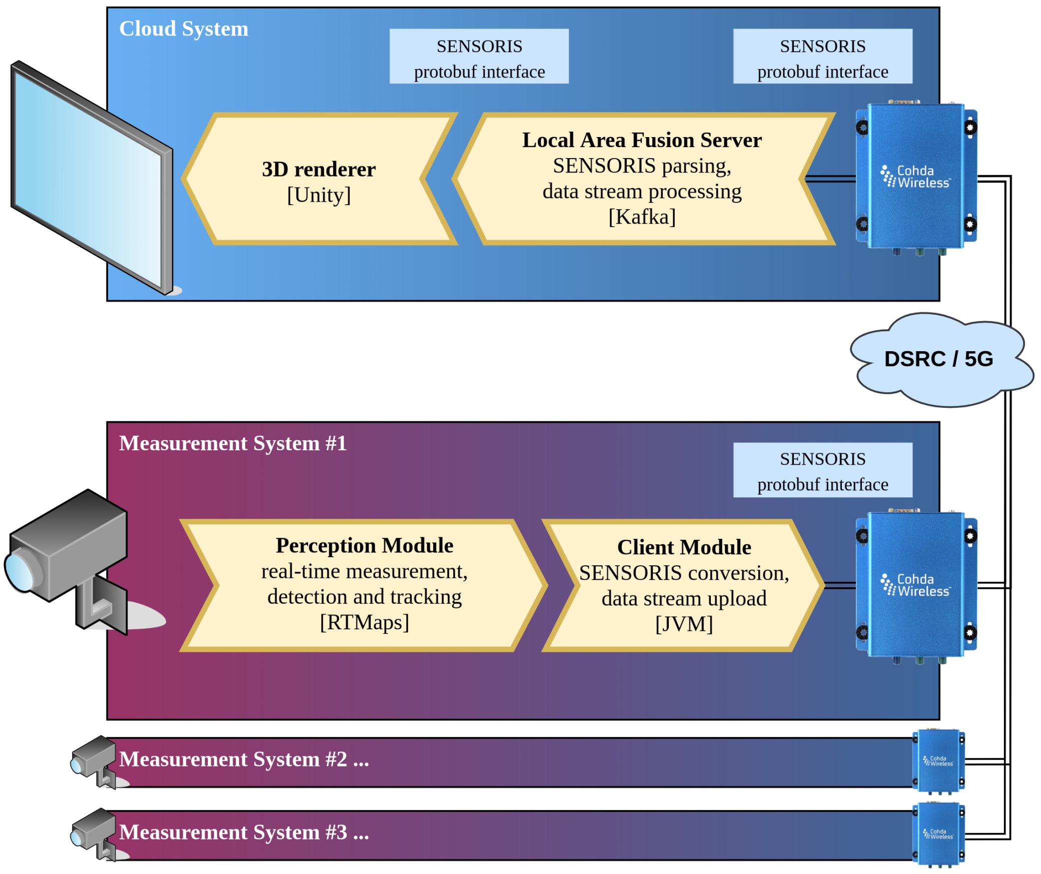
Cloud System
SENSORIS protobuf interface
SENSORIS protobuf interface
3D renderer [Unity]
Local Area Fusion Server SENSORIS parsing, data stream processing [Kafka]
Cohda Wireless
DSRC / 5G
Measurement System #1
SENSORIS protobuf interface
Perception Module real-time measurement, detection and tracking [RTMaps]
Client Module SENSORIS conversion, data stream upload [JVM]
Cohda Wireless
Measurement System #2 ...
Measurement System #3 ...

Figure 2. *Central Perception* prototype main components and protocols.

4. Perception Module

4.1. System Calibration

The proposed *Central System* integrates numerous different type of sensors (each having an assigned local coordinate system) such as LiDARs, cameras deployed in the infrastructure or attached to a vehicle. In addition to these sensors—in case of vehicles—the IMU/dGPS stand for an additional key element. For the system to work properly the calibration parameters have to be estimated first, i.e., all the intrinsics and extrinsics. Here, the intrinsics cover the internal parameters of cameras (such as focal length, principal point coordinates, skew, radial and tangential lens distortion) and the extrinsics stand for the transformations between the local coordinate systems of attached sensors as well as transformations from and to the Universal Transverse Mercator (UTM) frame which was selected to represent the global reference frame. For camera calibration the method published in [12] has been applied. The simplified calibration setup is illustrated by Figure 3. Let us briefly introduce the calibration approaches used to calibrate the proposed *Central System*.

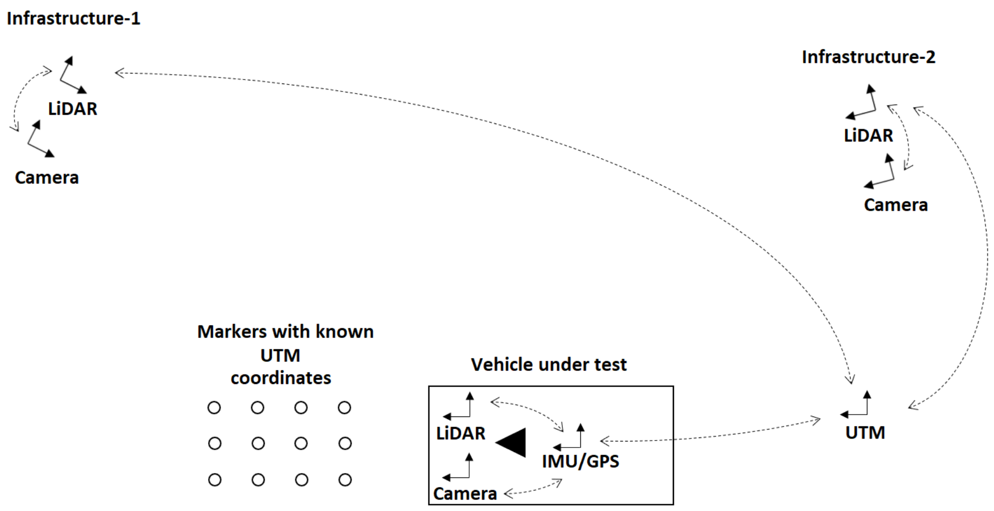

Figure 3. Illustration of the simplified setup of calibration containing two infrastructure stations and a vehicle each equipped with a camera and a LiDAR. In the vehicle there is an IMU/dGPS system, as well.

4.1.1. Chessboard Based Camera-LiDAR Calibration

The estimation of the rotation and translation between the camera-LiDAR pairs is more challenging than that of the camera-camera pairs, since we have to identify 3D points in the LiDAR point cloud and their corresponding image points in camera images. The estimation of LiDAR-camera extrinsics was based on the method proposed by authors in [13], which is a fully automatic extrinsic calibration approach aimed for LiDAR-camera extrinsics calibration by using a printed chessboard attached to a rigid planar surface. The key element of the method is to determine the 3D locations of chessboard corners in LiDAR's coordinate system. A full-scale model of the chessboard (A0 sized) is fitted to the segmented 3D points corresponding to the chessboard in the LiDAR point cloud. The intensities of light rays reflected form black and white patches of the chessboard are different and well distinguishable, thus the model is fitted to a 4D point cloud where the last dimension corresponds to the intensity of the given LiDAR point). The corners of the fitted model are considered to be the 3D corners of the chessboard.

The extrinsic calibration of the camera and LiDAR is performed by minimizing the re-projection error (given the estimated corners $\mathbf{M}_i$ in the LiDAR frame, their measured projections in the camera image $\mathbf{m}_i$ as well as the intrinsics of the camera (camera matrix $\mathbf{K}$, radial and tangential distortion coefficients p_1, p_2, p_3, q_1, q_2). N stands for the number of corner points considered.

$$\min_{\mathbf{R},\mathbf{t}} \sum_{i=1}^{N} \|\mathbf{m}_i - \hat{\mathbf{m}}_i(\mathbf{M}_i, \mathbf{K}, \mathbf{R}, \mathbf{t}, p_1, p_2, p_3, q_1, q_2)\|^2 \tag{1}$$

Figure 4 shows the process of data acquisition. The blue and yellow colors correspond to different LiDAR point intensities. Figure 5 shows the LiDAR points projected onto the camera image. In the chessboard image we can see both the detected corners and the 3D corners identified in the LiDAR point cloud and projected onto the camera image. The 3D viewer shows the detected corners together with the point cloud of the chessboard colored based upon the black and white patches of the fitted chessboard model. The achieved RMSE in case of five different poses of the chessboard can be followed in Table 1.

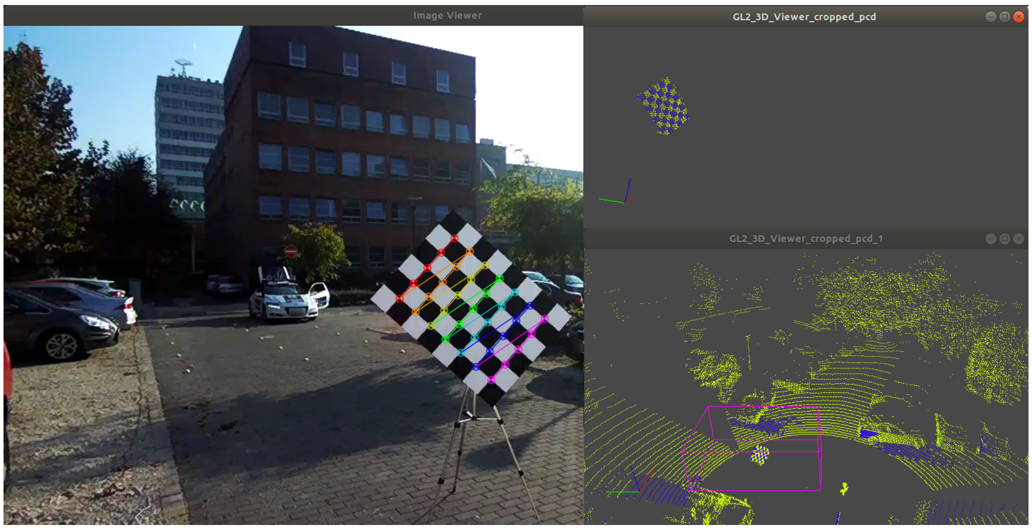

Figure 4. Data acquisition for chessboard-based camera-LiDAR calibration.

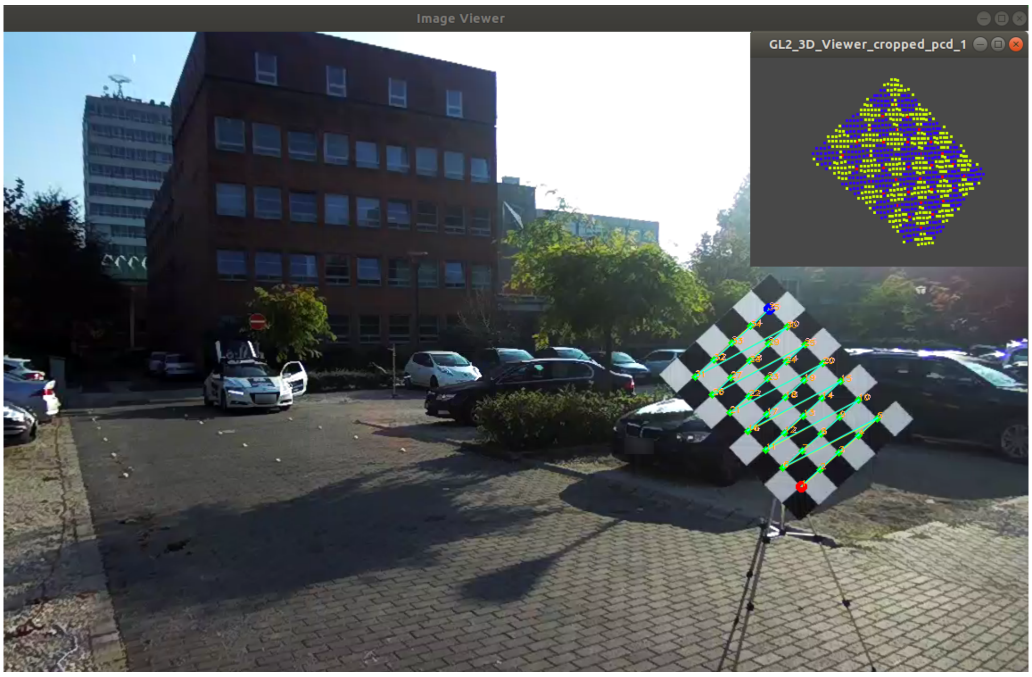

Figure 5. chessboard-based camera-LiDAR calibration results.

Table 1. The achieved RMSE in case of five poses.

Chessboard Pose Index	*RMSE [px]*	*Chessboard Size*
1	1.66	The chessboard pattern is 6 × 8 with a cell size of 140 × 140 mm
2	1.30	
3	1.49	
4	0.87	
5	0.87	
The whole image set	1.28	

4.1.2. Box Based Calibration

In this second approach, the Camera-LiDAR extrinsics calibration relies on box corners, instead of a chessboard. The calibration box is placed at different locations (with known UTM coordinates measured in advance by a portable dGPS modul) of the working area. For each pose of the box the point cloud and the corresponding camera image is acquired. In the LiDAR pointcloud, the box corners were determined by segmenting the points which correspond to the box in the point cloud followed by fitting the box model. Since the box corners are obtained with respect to the LiDAR coordinate frame and the corresponding UTM coordinates are known, the LiDAR pose wrt. the UTM frame can be estimated. Similarly, the camera pose can be determined by minimizing the re-projection error (see Equation (1)) associated with the selected corners with known UTM coordinates. Since the UTM coordinates of box corners as well as the IMU pose wrt. UTM are known, the transformation between the LiDAR and IMU coordinate systems can also be determined. Another well known approach to estimate LiDAR-UTM extrinsics is the hand-eye calibration which requires at least two motions (with non-parallel rotation axes) of the sensors (LiDAR and IMU) [14].

4.2. Data Synchronization

In order to keep the data streams synchronized among infrastructural and vehicular sensors, a common time source as well as a time protocol is needed. As the most commonly used time protocols, the Network Time Protocol (NTP) and the Precision Time Protocol (PTP) might be emphasized. During our experiments the NTP was utilized and the GPS time was used as time source. Each station (including two infrastructural stations and one measurement vehicle) was equipped with an on-board unit having an integrated GPS time source and running the NTP service (see Section 6.1). Each computing node's (PCs, DrivePX2 Tegra A and Tegra B) system clock has been synchronized with the GPS time by relying on the NTP protocol. Prior to testing the *Central Perception* system, the synchronicity of data streams from different sensors have been verified by experiments. To each data frame (independently on what type of sensor it originates from), a timestamp is assigned as it enters the computing framework. In the computing framework the data frames (from different streams) being closest in time are associated and processed afterwards as depicted by Figure 6.

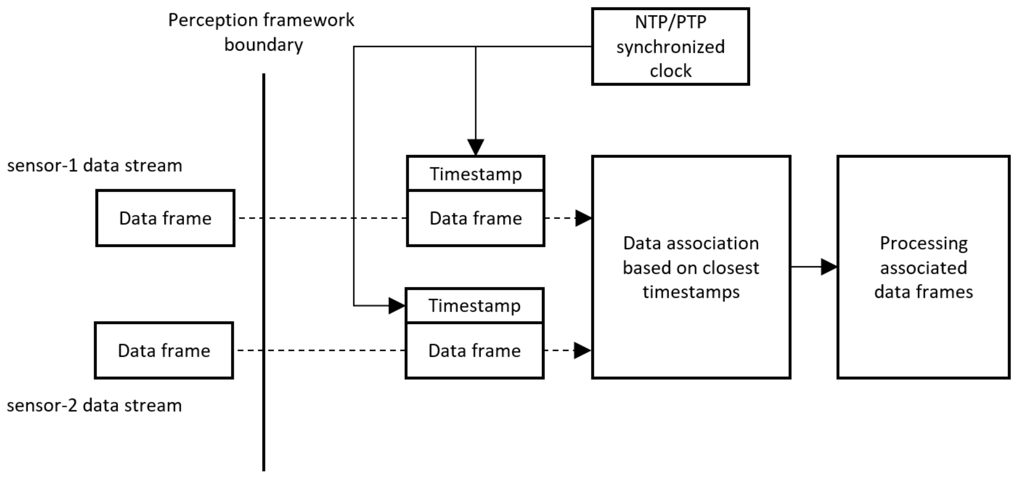
Perception framework boundary
NTP/PTP synchronized clock
sensor-1 data stream
Data frame
Timestamp
Data frame
Data association based on closest timestamps
Processing associated data frames
Timestamp
Data frame
Data frame
sensor-2 data stream

Figure 6. Illustration of the data flow and timestamp based assignment of data frames.

As another alternative for time synchronization the Precision Time Protocol might be used, which instead of millisecond-level synchronization, aims to achieve nanosecond- or even picosecond-level synchronization. In case of the PTP, switches with PTP support are required for each station. For most commercial and industrial applications, NTP is more than accurate enough [15].

4.3. 3D Object Detection

3D object detection plays crucial role in environment perception and understanding. During the development of the *Central System* we payed much attention on the development of robust 3D object detection methods which are considered to be essential from the overall system performance point of view. We have considered two type of approaches:

1. camera-lidar based approach benefits from the high resolution of cameras and the high position accuracy of 3D LiDAR points.
2. a single camera based detection where the YOLO 2D detection [16] algorithm and the homography between the road plane and the image plane was considered. This approach is cost efficient (due to the camera only requirement) and is preferred to be applied in such scenarios where the camera is static.

4.3.1. Yolo and Point Cloud Based Approach

Camera-based systems perform outstandingly well in case of recognition tasks, but when it comes to position estimation they are less accurate than LiDAR-based systems. Depending on the resolution and the number of used cameras, the baseline length, the accuracy of calibration as well as the accuracy of the pixel coordinates of points of interests, the position estimation might be improved; however, by including one or more LiDARs the location estimation of object's might significantly be improved.

The method presented below combines the advantages of the two sensors (i.e., the high resolution of cameras and the localization capabilities of LiDARs). In order to fuse camera images with LiDAR point clouds the sensors have to be calibrated (see Figure 3). Another crucial point here is to guarantee real-time processing which puts additional constraint (depending on the used hardware) on the complexity of applied algorithms. Nevertheless, the data streams of different sensors must be kept synchronized to ensure that data frames closest to each other in time are associated and processed accordingly (see Section 4.2).

As first step the detector receives images on its input and 2D object detection is performed by the YOLOv4 object detector [16]. The speed and accuracy of the algorithm are in line with the requirements defined, which means that the frame rate of the overall system was set to be at least 20 FPS (which currently stands for the upper limit for LiDARs). During the experiment pedestrians and cars have been considered as primary objects of interest, however the algorithm can easily be extended to detect additional classes such as motorcycles, buses, etc. The 2D detection may take several milliseconds even on the most powerful hardware ($\sim$30 ms).

In the next stage, the point cloud is projected onto the camera image and each estimated 2D bounding box gets associated with the LiDAR points which projections are bounded by the given box. As result a set of frustums is obtained (one for each 2D bounding box) containing the 3D points of the objects of interest. Let us denote the set of these frustums by $\mathcal{F}_i$. Given $\mathcal{F}_i$ the 3D bounding box corresponding to the given object might either be estimated on neural basis by a convolutional neural network trained to perform detection in frustums or the position of the object might be determined based on a simple reasoning.

By the reasoning based approach first the false points (foreground, background points) from $\mathcal{F}_i$ are eliminated and a small 2D window inside each bounding box is defined. The size and position of the window is proportional to the size of the original box. The scaling factor and position were set empirically based on the type of object. Since these windows generate significantly narrower frustums, the points falling inside it are more likely to belong to the object of interest. Let us denote the set of these points as $\mathcal{F}_i^{'}$. The location of the object is determined as the mean of the points falling inside the volume bounded by $\mathcal{F}_i$ and satisfying the constraint

$$d_{min} < \|\mathbf{p}_j - \mathbf{c}\| < d_{min} + \delta, \quad \mathbf{p}_j \in \mathcal{F}_i, \quad j = 1..N_i, \tag{2}$$

where $\mathbf{c}$ stands for the camera center and N_i represents the number of points in $\mathcal{F}_i$, $\delta = \max\{object_{width}, object_{height}\}$.

$$d_{min} = \underset{\mathbf{q} \in \mathcal{F}_i'}{\operatorname{argmin}} \|\mathbf{q} - \mathbf{c}\|. \tag{3}$$

This is an extremely simple and therefore very fast way to filter out unnecessary points and localize the object of interest Figure 7. The latency of the detection can be followed in Figure 8.

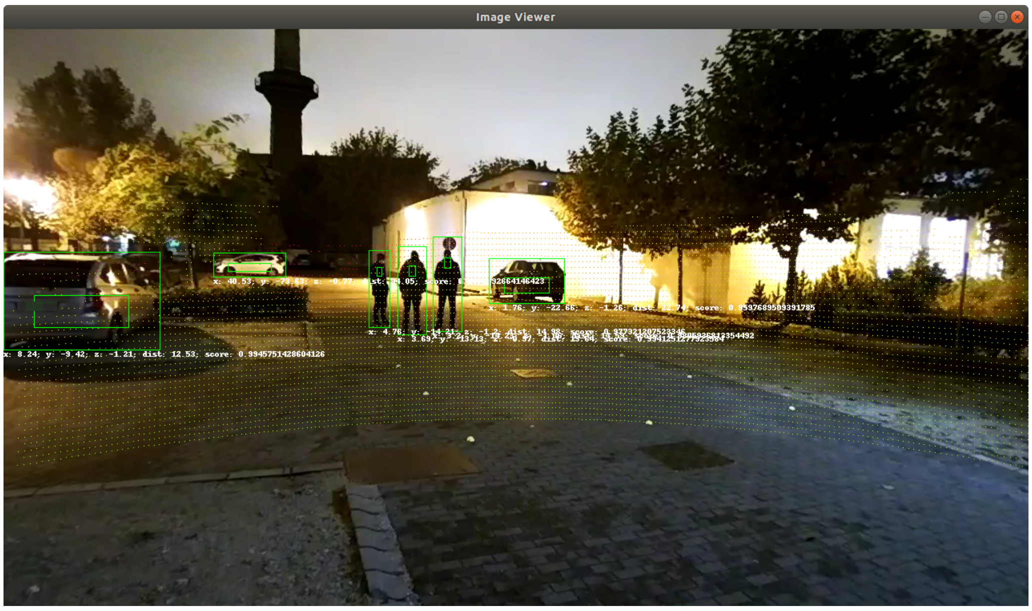
Image Viewer
x: 8.24; y: -9.42; z: -1.21; dist: 12.53; score: 0.9945751428604126
x: 40.53; y: -73.63; z: -0.77; dist: 84.05; score: 0.92664146423
x: 1.76; y: -22.66; z: -1.26; dist: 22.74; score: 0.9597689509391785
x: 4.76; y: -14.21; z: -1.2; dist: 14.98; score: 0.977321207523346
x: 3.69; y: -13.13; z: -0.97; dist: 13.64; score: 0.994125127732398

Figure 7. Detected objects.

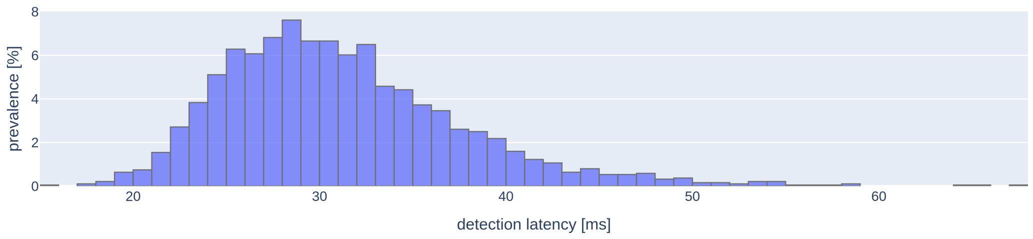

Figure 8. Yolo based object detector latency.

4.3.2. Yolo and Homography Based Approach

The detector described above uses the lidar point cloud to estimate the 3D location of the target, the method introduced in this section focuses on a single camera based 3D localization of targets. Homography and its estimation is well known topics in the literature, but let us briefly summarize it: Let us denote a world point by $\mathbf{M}$ and its image coordinates by $\mathbf{m}$. Let us consider the scenario when the world points of interest are lying on the XY plane, thus their Z coordinate is zero. These points are projected onto the image plane of the camera as follows:

$$\mathbf{m} = \mathbf{PM} = \mathbf{K}[\mathbf{R} \mid \mathbf{t}] \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{r}_3 & \mathbf{t} \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = \underbrace{\mathbf{K}\begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{t} \end{bmatrix}}_{\mathbf{H}} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}, \tag{4}$$

where r_i denote columns of the rotation matrix $\mathbf{R}$, $\mathbf{t}$ stands for the translation and $\mathbf{K}$ denotes the camera matrix containing the camera intrinsics. In order to estimate the homography

$\mathbf{H}$ the following cost function is minimized (measurement error is considered in both the image and world plane):

$$\min_{\mathbf{H},\hat{\mathbf{m}}'_i,\hat{\mathbf{m}}_i} \sum_{i=1}^{N} \|\mathbf{m}_i - \hat{\mathbf{m}}_i\|^2 + \|\mathbf{m}'_i - \hat{\mathbf{m}}'_i\|^2, st.\ \hat{\mathbf{m}}'_i = \mathbf{H}\hat{\mathbf{m}}_i,\ \forall\, i, \tag{5}$$

where $\mathbf{m}_i$ and $\mathbf{m}'_i$ stand for the measured point pairs while $\hat{\mathbf{m}}'_i$ and $\hat{\mathbf{m}}_i$ stand for the estimated perfectly matched correspondences, i.e., $\hat{\mathbf{m}}'_i = \mathbf{H}\hat{\mathbf{m}}_i$ [17].

We have used 18 markers $\mathbf{m}'_i$ with known UTM coordinates (measured by a mobile GNSS system in advance with an accuracy of $\sim$20 mm) and their image projections $\mathbf{m}_i$ to estimate the homography. $\mathbf{m}_i$ stand for the undistorted normalized image points. The detection part of the approach uses the YOLO4 [16] neural network to detect targets of various types in images (during our experiment pedestrians were the main objects of interest, however other object types are also supported by the proposed perception system). The point of interest for each detected pedestrian was set to be the center point of the bottom edge of its 2D bounding box. Let us denote these points by $\mathbf{m}_i$. By applying the estimated homography $\mathbf{H}$, the image points m_i can be transformed to the XY plane of the UTM coordinate system as $\mathbf{m}'_i = \mathbf{H}\mathbf{m}_i,\ \forall i$. Here we omit the true altitude, thus it was set to zero for each point. Although this kind of 3D detection is very useful for static cameras (for example cameras installed in the infrastructure), in case of cameras attached to a vehicle, change in pitch or roll of the vehicle (caused for example when accelerating or making a hard turn, etc.), invalidates the estimated homography. In addition, the uncertainty of the measured image points $\mathbf{m}_i$ must also be considered. Given both the uncertainty of $\mathbf{H}$ and $\mathbf{m}_i$, the covariance of the estimated points $\mathbf{m}_i{}'$ is given by:

$$\mathbf{\Sigma}_{\mathbf{m}_i{}'} = \mathbf{J}_\mathbf{h}\, \mathbf{\Sigma}_\mathbf{h}\, \mathbf{J}_\mathbf{h}^T + \mathbf{J}_{\mathbf{m}_i}\, \mathbf{\Sigma}_{\mathbf{m}_i}\, \mathbf{J}_{\mathbf{m}_i{}'}^T, \tag{6}$$

where $\mathbf{\Sigma}_{\mathbf{m}_i{}'}$, $\mathbf{\Sigma}_{\mathbf{m}_i}$ and $\mathbf{\Sigma}_\mathbf{h}$ stand for the covariance matrix of the estimated road point $\mathbf{m_i}'$, the measured image point $\mathbf{m}_i$ and the estimated homography $\mathbf{h}$, respectively (vector $\mathbf{h}$ is composed from the concatenated rows of $\mathbf{H}$). Furthermore, $\mathbf{J}_{\mathbf{m}_i}$ and $\mathbf{J}_\mathbf{h}$ stand for the Jacobians of $\mathbf{m}'_i = \mathbf{H}\mathbf{m}_i$ wrt. $\mathbf{m}_i$ and $\mathbf{h}$, respectively [17].

Since the vehicle is moving, the detected objects have to be transformed according to the actual pose of the vehicle to a global coordinate system (e.g., UTM). Firstly, the detections should be estimated wrt. a selected reference coordinate system and then based on the actual pose of the vehicle transformed to the UTM frame. The reference coordinate system for the vehicle was set to be the coordinate system of the IMU shifted along the vertical axes to the ground level (road level). Let us refer to this coordinate system as IMU_{gl}. In order to estimate the homography which transforms the image points directly to IMU_{gl}, one needs to estimate the marker coordinates in IMU_{gl}. Since the IMU modul (used during our experiments) includes a differential GPS with a dual antenna system, the UTM coordinates and the heading of the vehicle can be measured with an accuracy of $\sim$20 mm which might be considered to be sufficient for autonomous driving related applications. Based on the measured pose of the vehicle, the UTM to IMU_{gl} rigid transformation can be determined, thus the markers in IMU_{gl} can be calculated. By applying the homography (estimated based upon markers in the IMU_{gl} and the corresponding image plane points) to points m_i, the 3D position of each detected target is obtained directly in IMU_{gl}. Since the pose of the vehicle is continuously measured with a sampling rate of 100 Hz, the detections can directly be transformed to the UTM frame in real-time.

4.4. Object Tracking

4.4.1. Overview

There are several different types and solutions for tracking objects [18]. Kalman-filter based methods are widely used for target position tracking.

For tracking objects with high manoeuvring capabilities, utilisation of the Interacting Multiple Model (IMM) filter is a good practice. Although the IMM filter is a well known approach for object tracking, let us briefly point out the basic principle. The IMM filter considers multiple motion models (e.g., constant velocity, constant acceleration, constant turn rate models) each associated with a dedicated Kalman filter. The Kalman filters are running simultaneously in parallel and their outputs are blended to generate the estimated state of the system according to the likelihoods of being in a certain motion mode. The higher the probability of a mode, the higher its contribution to the blended state. The state of a more probable mode is affected slightly by less probable modes [19,20]. During this process, the likelihoods of being in a certain mode (e.g., constant velocity mode) and the likelihoods of transitions between modes are calculated based on the last state. In order to reduce the transient period every filter is reinitialized with the mixed estimate of state and covariance [21].

4.4.2. Implementation

The tracker component receives a description data structure from all recognised objects as input. From that data structure, it pulls the position coordinates and the corresponding timestamp and combines them into a position list for a given frame. The core of the tracker is an IMM filter. The filter consists of three different motion models, which are the constant velocity, constant acceleration and constant turn-rate models. In each step, the tracker gives an estimation of current positions for all the registered tracks. Then the tracker component pairs the tracks with the input positions using Munkres global nearest neighbour assignment algorithm. Then it manages the tracks in the following manner: If no existing track can be paired with a position, it creates a new one. If a track was paired with any positions 5 times within the last 7 frames, then it flags it as confirmed. With this method, any false positive detection can be filtered out. The component deletes a track when it has not been assigned with any positions at least 22 times within the last 25 frames. These settings fit the Yolo and point cloud based approach. Due to behaviour differences, the Yolo and homography based approach requires other settings for the tracker component for best results. Therefore, a tracker with optimised settings has been implemented for each detector solution. After the track management, the component compiles a list with the positions of the confirmed tracks. The output of the component is a data structure that contains the ID and position of the tracks and the position and orientation information of the sensor system. The output data structure has the same format as the input structure. The pseudo code of the tracker is listed below:

The latency histogram of tracking can be followed in Figure 9. First of all it is influenced by the number of current tracks and detections. The average response time of the component is 690.7 µs for the sample sequence.

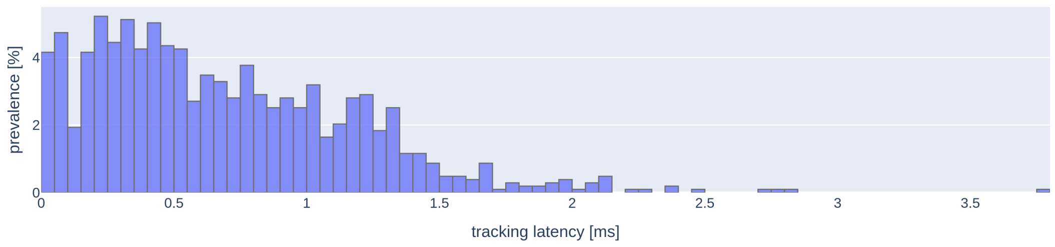

Figure 9. Latency histogram of tracking.

```
READ input
FOR elements in detection data:
READ position
FOR each track:
    LOAD last position
    ESTIMATE new position with IMM Filter
ASSIGN estimations with current detection positions
FOR each track:
    IF track was paired with a detection for n times in the last m
    frames:
        REGISTER track as ,,Confirmed''
    IF track was paired with detection less times than j in the last
    k frames:
        DELETE track
IF there are any detection which were not assigned to a track:
    FOR each unassigned detection:
        CREATE new track based on current detection
ASSEMBLE a list from the positions of ,,Confirmed'' tracks
CREATE output data structure
ADD system origin position and yaw information from input data
WRITE output
```

5. Local Area Fusion Server

5.1. Stream Setup

The local area fusion server we set up for our current demonstration automatically processes and converts the incoming detections streams across five sequential processing steps until we get the fused result in the final stream. The five so-called fusion-processors can be observed in Figure 10.

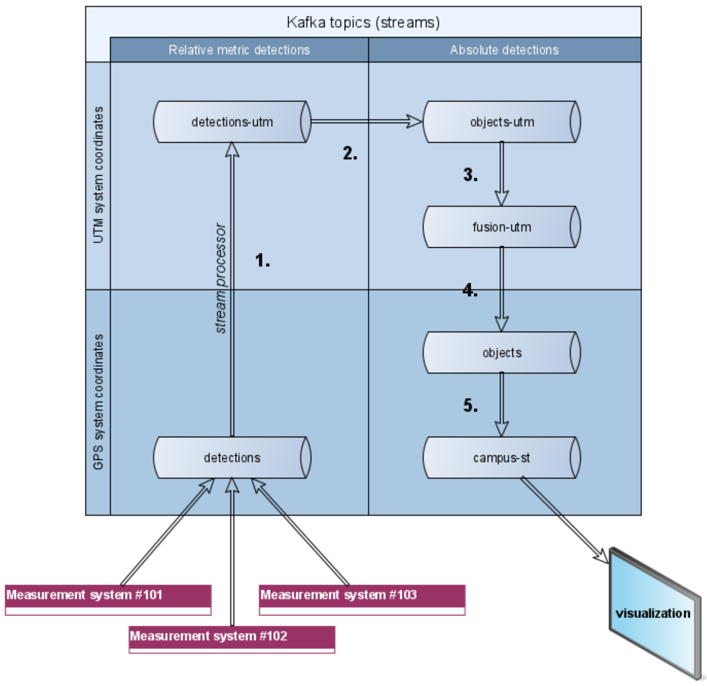

Figure 10. Current stream setup in the *Central Perception* functional sample server (cylinders represent topics/streams, while the numbered arrows represent stream processors).

The sequentially numbered stream processors from Figure 10 have the following responsibilities:

1. Converts system origin coordinates from WGS84 to UTM. Position uncertainty remains the same since the SENSORIS input was in (square) meters already. Rotation and its uncertainty do not change because WGS84 and UTM frames point in the same directions.
2. Recalculates all object coordinates from system-relative to absolute UTM.
   (a) **Position:** The transformation matrix is straightforward to derive from the relative frame (e.g., vehicle IMU): we just calculate the rotation matrix from the platform's current orientation and append its current position as a translation vector.
   (b) **Orientation:** The global heading is obtained by adding the relative (IMU-based) object yaw to the system yaw. Calculating global pitch and roll is more involved and was skipped since this data is not represented in our current environment model. The pitch and roll values are set to zero.
   (c) **Position covariance:** the object position covariance matrix has to be backrotated and added to system position covariance, assuming no cross-covariance between system and object positions since they are independent.

   $$\mathbf{\Sigma_{op}} = \mathbf{\Sigma_{pp}} + \mathbf{R}\ \mathbf{\Sigma_{rp}}\ \mathbf{R}^\top \tag{7}$$

   $\mathbf{R}$ denotes the IMU-to-UTM rotation matrix, while $\mathbf{\Sigma_{op}}, \mathbf{\Sigma_{pp}}, \mathbf{\Sigma_{rp}}$ denote the resulting object position covariance, the platform position covariance and the IMU-based relative object position covariance, respectively.
   (d) **Orientation covariance:** the object heading variance is added to the system heading variance, the pitch and roll uncertainties are disregarded (set to identity).
3. Here, the fusion algorithm itself is performed on the objects-utm stream. Exact details will be given in the next subsection. For convenient further utilization a very specific rule for populating the output fusion-utm stream is applied: the chosen fusion input messages and the fused output message are written sequentially to the out-stream. So later reading the messages in offset order will yield an alternating sequence of fusion inputs followed by the corresponding fusion output. Note that not every objects-utm message becomes a fusion input.
4. Converts all objects from UTM coordinates into WGS84 coordinates.
5. Optionally filters certain objects according to position in relation to demonstration area.

5.2. Fusion Algorithm

Assuming no cross-correlation between sources, we employed a Kalman filter and Global Nearest Neighbor (GNN) association based central tracking source-to-track fusion method called trackerGNN (https://www.mathworks.com/help/fusion/ref/trackergnn-system-object.html (accessed on 14 September 2021)), which is an integral part of the Sensor Fusion and Tracking Toolbox of Matlab. TrackerGNN maintains a single hypothesis (set of central tracks) about the environment and it follows the central tracking algorithm template detailed in the following subsection. The theory behind the implementation is based on [22]; notably it solves GNN association using the Kuhn-Munkres [23] algorithm, also known as the Hungarian method [24].

5.2.1. Central Tracking

Central tracking, sensor-to-track or source-to-track (S2T) fusion has detections from multiple sources (usually sensors) as inputs and is expected to produce a single set of central tracks as output. Therefore, the detections have to be integrated across time and across sources. If we first perform the time-integration (tracking) and subsequently perform the source-integration (fusion), we get the equivalent of a track-to-track (T2T) fusion approach.

In contrast, if we perform source-integration (fusion) before time-integration (tracking), we are talking about S2T fusion.

The general S2T fusion framework assumes the maintenance of a single set of central tracks throughout the filtering steps. An S2T fusion step usually follows the template given below:

1. Collect measurements within a time interval between previous and currently queried step time. Each potential source should provide exactly zero or one measurement containing a number of simultaneous detections. Sequential measurements from the same source within the data collection interval can be handled by (a) discarding all but the last, as done in our approach; or (b) keeping all but technically regarding them as different sources with shared measurement model parameters.
2. Assign each detection of each source to exactly one track (either pre-existing or newly-created). Make sure that no two detections from the same source are assigned to the same track. Thus each track is assigned 0 to s detections (s being the number of sources at this step).
    (a) Track lifecycle management is done during this step (trackerGNN uses parametrizable heuristics as detailed in Section 4.4.2).
    (b) The assignment algorithm may handle passage of time. A simple solution like trackerGNN would disregard time and only use spatial data for assignment. A sophisticated solution might have to assign and integrate each measurement individually, ordered by time, iterating between steps 2 and 3, increasing computation costs.
3. Filter (predict and update) each live track with the assigned measurements ordered by time.
4. Output the prediction for all tracks for the same moment in time (which was provided as the fusion query argument).

5.2.2. Integration

In order to make use of trackerGNN as part of an efficient stream processor outside of Matlab, several technical challenges had to be overcome, most notably:

- C code generation and compilation from Matlab,
- generation of a Java wrapper using the SWIG (http://swig.org/ (accessed on 14 September 2021)) framework for integration with Kafka Streams API,
- devising and implementing an appropriate input buffering scheme within the stream processor, and finally
- making sure both real-time and playback fusion options are supported.

5.2.3. Buffering

The issue of input message buffering is not entirely trivial, since the order of message arrival in the topic partition does not necessarily follow any kind of (e.g., timestamp-based) ordering. We have to somehow make sure that the tracker is always fed appropriately chosen inputs and that no inputs are wasted or discarded prematurely. The regular intervals the fusion is queried at (in our case 120 ms) require a flexible buffering method that supports fusion with missing or no data, old and future data, etc.

Our buffering method:

- tries to collect and buffer all available data immediately and continues to collect as long as the stream is accessible;
- discards messages with past timestamps that precede the most recent fusion step;
- preserves far-future data points without running out of memory;

- collects data falling—according to timestamp—into each inter-fusion time window into separate sets;
- when the time for the next fusion step comes, the cluster of messages that falls into the preceding time-window is regarded: a collection of one latest message per sensor is retained as fusion input, the rest discarded.

5.2.4. Real-Time vs. Playback Mode

The behavior of the fusion module should be different when we play back data from the past then when we stream the present data. In both cases fusion is performed at fixed real time (not timestep time) intervals, and fusion time can proceed only forward (strictly greater than previous) (see Table 2).

Table 2. Comparison of real-time and playback fusion modes.

Real-Time	**Playback**
Pace: always jump forward to the most recent available timestep.	Pace: always read input data with a natural pace: for every second passed in consumed timesteps, a second should pass in reality.
Missing data: our requirement for fused tracks is to disappear when no data is received from any of the sensors, i.e., to artificially advance fusion in time and "wind down" within a dozen simulated steps.	Missing data: when no data is received from any of the sensors, we want to freeze everything as it is and to not step the time forward. Empty (0-detection) data can clear the scene, but no-data should freeze it.
Fusion restart: not required, since real time can flow only forward.	Fusion restart: required when rewinding.

5.2.5. TrackerGNN Parametrization

The method parameters for the trackerGNN fusion component were set to:

```
tracker = trackerGNN( ...
    'TrackerIndex', 0, ...
    'FilterInitializationFcn', @initcvkf, ...
    'Assignment', 'MatchPairs', ...
    'AssignmentThreshold', 15*[1 Inf], ...
    'TrackLogic', 'History', ... \% History|Score
    'ConfirmationThreshold', [2 3], ...
    'DeletionThreshold', [5 5], ...
    'DetectionProbability', 0.9, ...
    'FalseAlarmRate', 1e-6, ...
    'Beta', 1, ...
    'Volume', 1, ...
    'MaxNumTracks', 100, ...
    'MaxNumSensors', 20, ...
    'StateParameters', struct(), ...
    'HasDetectableTrackIDsInput', false, ...
    'HasCostMatrixInput', false ...
);
```

The detection-to-track assignment upper threshold was set to half of the default since pedestrians are smaller and slower than vehicles and are expected to have smaller uncertainty. Track confirmation threshold was set to 2 out of 3 detections, although our detections often come with a nonzero object type meaning instant confirmation. The track deletion threshold was set to 5 out of 5 misses. Besides, a custom rule was also introduced removing all input detections with any position covariance value larger then an experimentally chosen threshold $\epsilon = 5.0\ \text{m}^2$.

5.2.6. Results

The processing time of the local area fusion component was measured and found sufficiently performant for our requirements. Latencies are on the order of 2–3 ms, counting not only fusion itself, but including also stream (de)serialization and message parsing. There were some acceptably rare outliers: 0.14% of cases required more than 5 ms and none more than 35 ms (see Figure 11).

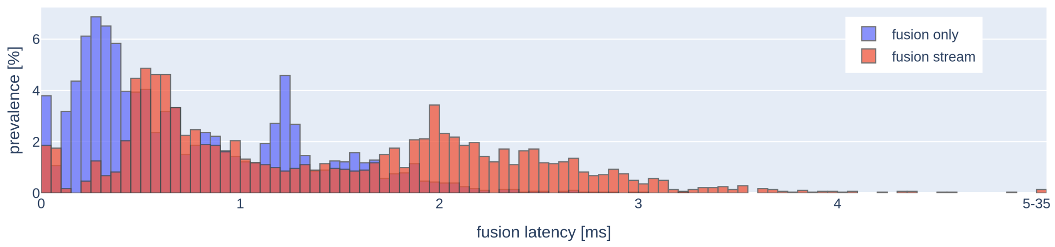

Figure 11. Distribution of processing times of the local area fusion component.

We have developed two distinct tools for visualizing fusion outputs. One is a 3D rendering demonstration tool described in Section 7. The other is a 2D monitoring dashboard for internal use that lets us step through each fusion cycle individually. A screenshot of the fusion results is presented in Figure 12 below.

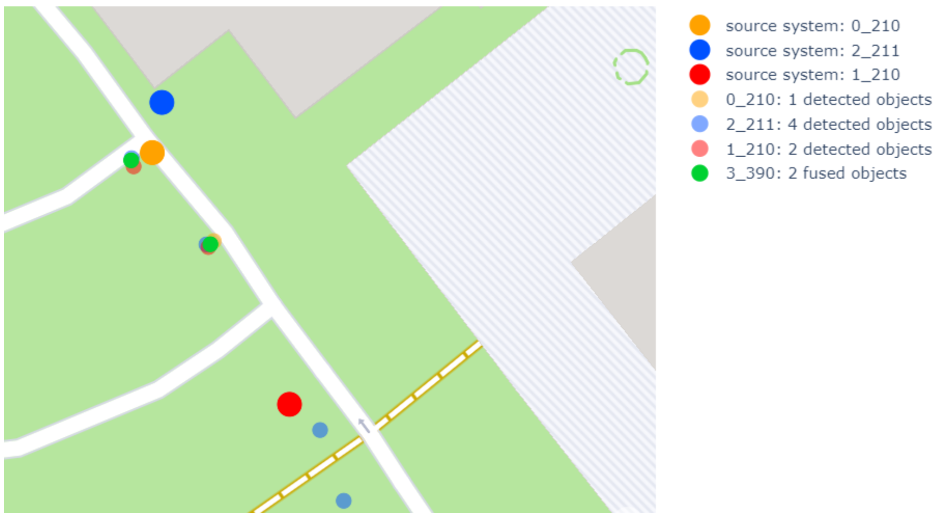

Figure 12. *Central System* Dashboard: our monitoring and analysis tool.

6. Client Module

6.1. DSRC Communication

Real-time communication, such as 5G cellular network or WiFi-based 802.11p (DSRC—Dedicated Short Range Communication), plays a major role in the system architecture. In the *Central Perception* functional sample of the *Central System*, the dedicated DSRC 5.9 GHz radio communication has made it possible for distant system components—such as the infrastructure stations, the vehicle, and the central server—to communicate with each other in real-time via radio frequency (RF).

For DSRC in our functional sample we used Cohda Wireless MK5 OBUs, which use the Software-Defined Radio (SDR) baseband processor SAF5100 and the dual-radio (antenna A, B) multi band RF transceiver TEF5100. These chips offer adjustable parameters for radio wave modulation schemes. The unit includes a dedicated HSM (Hardware Secure Module) for data encryption, compression, decryption and also the keys used for encryption using this chip [25]. The data rate corresponding to the modulation schemes of the device (BPSK, QPSK, QAM, etc.) can be changed from 3 Mbps to a maximum of 27 Mbps. At faster data rates, one of the most critical metrics, the PDR (Packet Delivery Ratio) is less than 100% so a trade-off had to be made and a medium rate, more reliable modulation option than BPSK (Binary Phase Shift Keying) was chosen.

The MK5 OBU complies with the following standards and protocols: IEEE 802.11 (part of the IEEE 802 set, the most widely used wireless networking standard), IEEE 1609 WAVE (Standard for Wireless Access in Vehicular Environments), ETSI ES 202 663 (European profile standard for the physical and medium access control layer for Intelligent Transport System operating in the 5 Ghz frequency band), SAE J2735 (Dedicated Short Range Communications (DSRC) Message Set Dictionary).

The 802.11p protocol compliance grants the following advantages:

- No additional infrastructure requirement: 802.11p does not require any additional infrastructure part, just the receiver and the transceiver units. This is because an ad-hoc network is formed, as soon as two DSRC units come in each other's radio range.
- Low latency: Road experiments have shown the latency at MAC layer to be around 2 ms or less in an optimal setup. The latency value depends on several different factors, such as payload size, vehicle speed (if the unit is mounted in a vehicle), radio interference, line of sight, etc.
- Range: The range is dependent on other variable factors like data rate and environmental factors. According to documentation it offers data exchange among vehicles and roadside infrastructure within a range of 1000 m, with a transmission rate of up to 27 Mbps and a vehicle speed up to 260 km/h.

The OBUs have another key role in the *Central System* architecture because they are also used for time synchronization, using the Chrony module and the GNSS antenna. The MK5 runs a gpsd server to allow applications to access GPS data. Chrony [26] is a versatile implementation of the NTP (Network Time Protocol), and it can synchronize the system clock with the NTP servers and reference clocks. With the help of the Chrony module all of the OBUs can be configured to have a reference time with microsecond accuracy.

Regarding the network topology in the DSRC setup, we define four subnetworks: two infrastructure stations, one vehicle, and one central server. Every subnetwork contains one PC for data acquisition, processing and visualization, one wireless router and one Cohda Wireless MK5 OBU.

The MK5 module has an Ethernet connection interface, which supports Ethernet 100 Base-T with 100 Mbps data rate. For the *Central Perception* functional sample the Cohda OBUs have been configured as IPv4 (Internet Protocol v4) gateways to provide a fully transparent communication between subnetworks. This means that all subnets are seen by each other, so real-time data exchange between nodes can be easily achieved. Figure 13 represents the subnetwork layout of the communication architecture of the *Central Perception* prototype.

6.2. *Kafka Streaming Platform*

For communication middleware we have chosen to use Kafka, the popular open-source "dumb broker" streaming platform maintained by the Apache Foundation. Judging by its main functionality Kafka can also be considered a distributed commit log, although it is primarily used for messaging. The aim of the project is to provide a real-time, high-throughput, low-latency streaming platform. Kafka provides horizontal scalability via distribution of message topic partitions across respective partition leader brokers while also providing fault tolerance by replicating each partition across non-leader brokers in a way reminiscent of RAID redundancy and fallback mechanisms. The distributed brokers and topic partitions architecture perfectly fits our long-term hierarchical edge computing vision if we assume that message topics should be divided into partitions according to the source area of measurements. We already tested our system in a 3-broker, 3-way-partitioned and triply-replicated (one original and two replicas) setup and experienced no perceptible lag or slowdown. When a broker was deliberately terminated, one of the remaining brokers automatically took up partition leadership; and when the temporarily disabled broker came back to life, the load balancing mechanism automatically reassigned it to partition leadership once again.

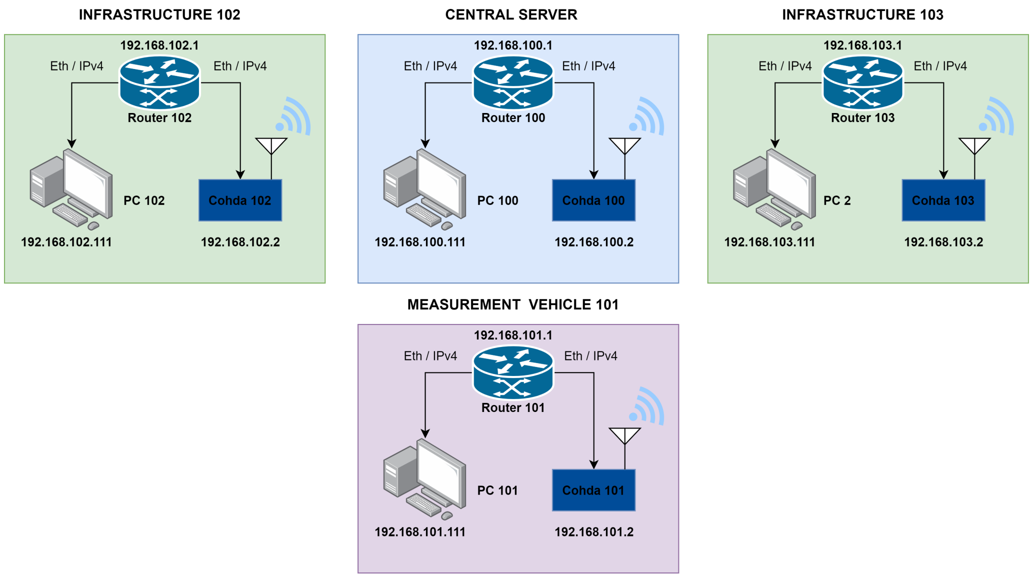

Figure 13. Subnetwork layout of the communication architecture.

In order to connect to the Kafka middleware, we developed a universal and platform-independent client module that runs in the Java Virtual Machine runtime environment in order to create a convenient socket-based API for uploading processed sensor data (detections, tracks, source system positions, etc.) to the distributed Kafka cloud in all the supported standard business domain level formats and protocols. This currently extends to the SENSORIS and ETSI CPM protocols, of which SENSORIS was used in the prototype demonstration. The client module's API encapsulates Kafka specifics and accepts standard SENSORIS messages. For further convenience we also provided a python wrapper API that we can easily call from the RT Maps client-side real-time dataflow-processing framework.

6.3. SENSORIS Message Standard

Exactly one message is sent for each source's each output (after each measurement-detection-tracking cycle). The source is not necessarily a single sensor, it might be, e.g., a raw sensor fusion based untracked detection, or the output of a tracker. The data that we collect via SENSORIS v1.0.0 (https://sensoris.org/ (accessed on 14 September 2021)) messages therefore contains the following elements:

- Message identification
- Source system information
  - identification (platform UUID, sensor UUID, sensor SUID) (UUID stands for universally unique identifier; SUID stands for system-wide unique identifier)
  - GPS PPS synchronized timestamp of originating measurement (event time)
  - localization (position, orientation) and its uncertainty
- Detected objects (i.e., detections or tracks) information [given for each object]
  - Object SUID

- Object existence uncertainty
- Object type and type uncertainty
- Object position and orientation in relative coordinate system and its uncertainty
- Object size and uncertainty

7. 3D Renderer

In order to visually represent the information provided by the individual sensor systems, as well as the central fusion system, it is necessary to use a digital twin rendering module. A properly constructed 3D visualization demonstrates the cooperative perception of scenario participants in scenarios where a single on-site sensor would not ensure proper operation. The visualization system must communicate with the *Central System*, including reading SENSORIS messages, and being able to decode and display this information in real time. It is also assumed that a digitized 3D model of the real environment of the on-site demonstration is available so that the visualized information can be compared with the real-world scenario. In our case, we used Unity 3D software to implement the visualization, which communicates with the *Central System* over a TCP connection. The localization of the measurement stations and their respective object detections (fused or raw) are available on a Kafka topic as encoded Sensoris messages. In order to visualize the measurement vehicle and the surrounding pedestrians, these data are accessed and forwarded to the visualization module in a proper structure.

7.1. Virtual Environment

Virtual imaging of the real environment is most accurate when based on laser measurements. Therefore, testing on the university campus was preceded by a laser measurement that provides a digitized description of the area as a LiDAR point cloud. This point cloud had to be brought from las format to some readable, xyz format to display within Unity software. In addition to the transformation of the format, it is important to place the lateral and longitudinal coordinate pairs relative to some center point in the x-y coordinate system defined by us so that the distances can also be interpreted in the Unity software. This transformation requires the use of the ellipsoid WGS84 as well as the determination of a clearly definable (0, 0) coordinate. This coordinate will later become the center of Unity's virtual world, as well as the basis for the transformation of all information that comes in during testing. The xyz data created in this way can already be read in a csv or txt file, and spheres representing the points can be created for the coordinate points it contains. In this way, it becomes interpretable in the virtual space of Unity, and based on this the various landmarks are clearly outlined (Figures 14 and 15). During the demonstration, the most important thing is that the roads are positioned correctly in the digital world, so we performed additional GPS measurements at their corner points. The origin of the virtual world was also determined during these measurements.

The shape and texture of the buildings surrounding the campus have been modelled according to reality. The shape and location of the vegetation and other components in the parking lot could be modeled based on the point cloud. The vegetation has been designed to vary the colour and density of the foliage according to the seasons. Unity software also provides the ability to model current lighting conditions using various skyboxes. However, for proper running performance, the generation of lights is not done in real-time. Still, a so-called baked lightmap is created, which predetermines illumination with the given settings.

The digital replica of the environment is best presented through cameras that can be matched to each real sensor. For a scenario to be well demonstrated, it is necessary to be able to present the given environment from several perspectives. We placed virtual cameras in the positions corresponding to the two infrastructure cameras as well as the cameras placed on the test vehicle, applying the basic properties of the real sensors.

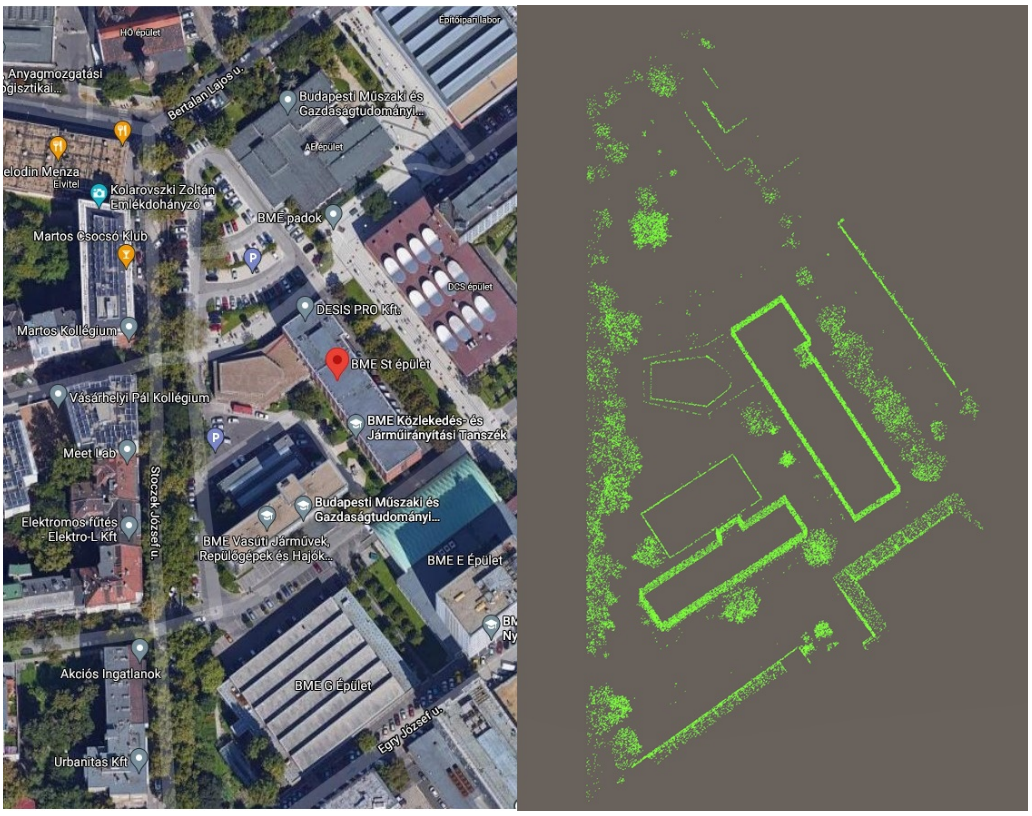

Figure 14. University campus based on LiDAR point cloud (map source: http://maps.google.com (accessed on 14 September 2021)).

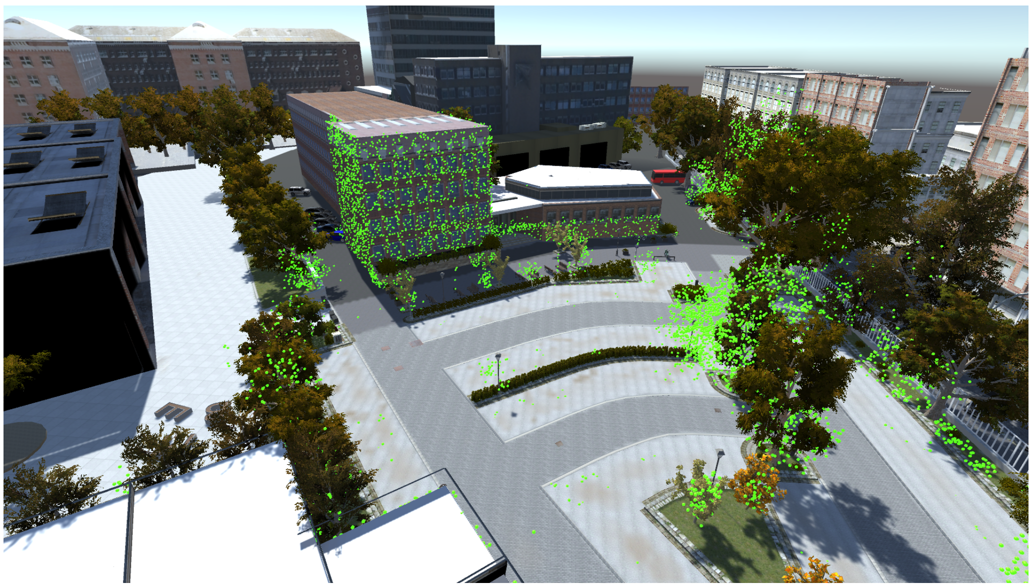

Figure 15. The Unity model of the University campus with the point cloud.

7.2. Sensor Detection Visualization

The test vehicle is displayed according to the method detailed in [27]. High-frequency real-time GPS data is available from the test vehicle that is accurate enough to place the vehicle in the virtual world. In this case, the lateral and longitudinal position of the vehicle and its heading are used. The movement of the vehicle's wheels was not modelled. When handling sensor detections—either from static stations or from the moving vehicle—it is necessary to separate information from different sources, as well as to handle different objects. Although only pedestrians were detected during our current measurements, the system is also prepared to handle all static and dynamic objects defined by SENSORIS. The detections from the sensors always reflect the state closest to real-time, i.e., only the most recent objects are always displayed. This also means that the objects existing in the previous update step must be moved or deleted. We also had to consider that the frequency of messages from different stations and sensor types is not the same in all cases, and may change dynamically. Residual detections—object tracks that get no confirmation within a short time period—are only rendered for the time specified by a parameter, after which they are automatically deleted. In our simulations, this time was set to 0.25 s. With these solutions, the movement of the detected pedestrians is continuous, there is no vibration in the display process, and the objects do not multiply during the movement, they do not draw a strip.

A visual distinction was made between detections of different origin, which helps us to understand the scenario. In addition, different sources also assign different tags to objects, which allows one to treat the objects belonging to that tag as a group, whether it is to turn off the display of objects or even delete objects. Pedestrian objects are generated based on a predefined cylindrical shape, the properties of which, such as size, colour, or permeability, are set based on the data associated with the detection.

The system provides a sufficiently high frequency to ensure that the motion is clearly continuous. There are two ways to test the visualization system, displaying real-time uploaded detections online and playing back data already present on the server offline. During the tests, we had two main expectations for the viewer, the first of which was to display the detection sent to it in real-time, and the second was to position both the environment and the detections accurately in the virtual world. With these conditions fulfilled, we observe a complete and synchronous copy of reality within the simulation. The system also allows one to turn off the display of detections of any given sensor for separate analysis. Figure 16 simultaneously shows the simulated camera FoV areas of all three sending infrastructures and a larger camera FoV overlooking the entire simulation environment. This figure also shows that the vehicle sensor sees a garage door (lower right corner), meaning that the snapshot came from a replay when the test vehicle did not participate in the measurement.

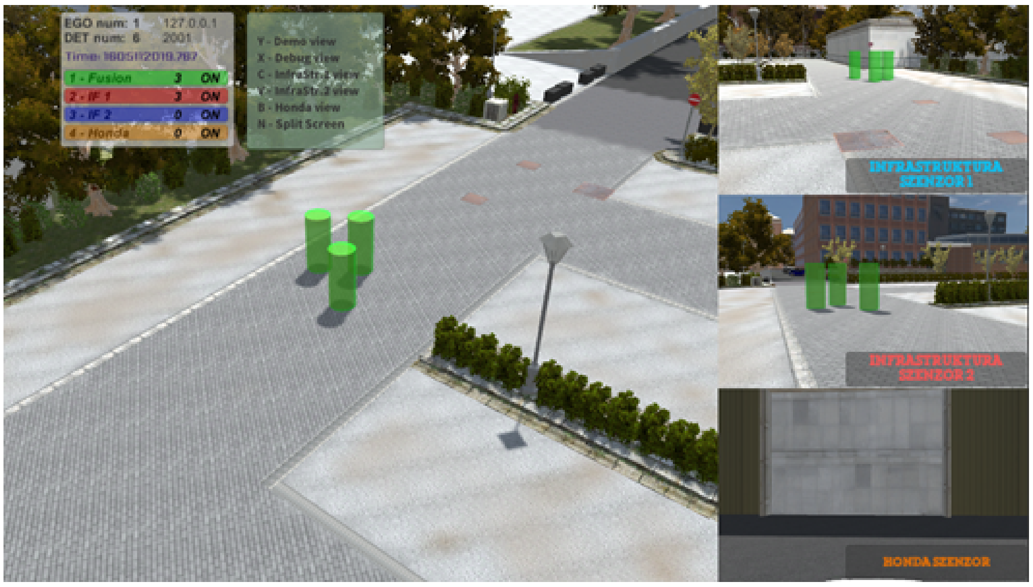

Figure 16. Detections in simulation.

We can best evaluate our digital twin during real-time tests, where we can see the scenario in reality and in its digital version at the same time. The streaming of raw camera images also provides additional checking possibilities. In the offline state, when a recorded stream is played back, the video played can also reveal whether there is a substantial difference in the digitized world compared to reality, as can be observed in Figure 17.

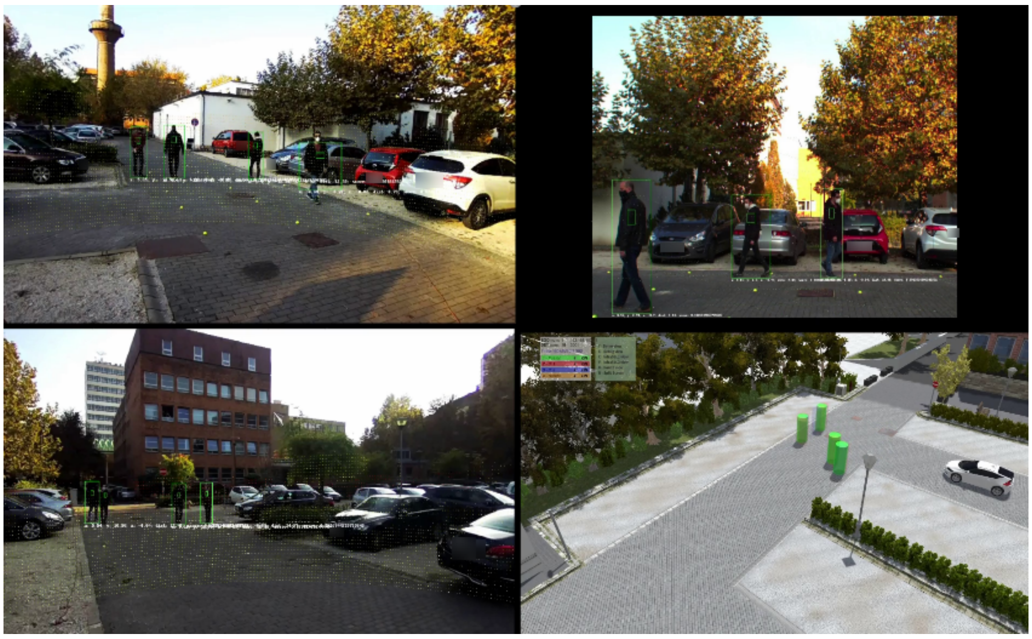

Figure 17. Top-left image: objects sensed by the infrastructure-1; **Bottom-left** image: objects sensed by the infrastructure-2; **Top-right** image: objects sensed by the vehicle; **Bottom-right** image: Real-time digital twin generated by the central system.

8. Conclusions and Future Work

We have proposed a cooperative perception system capable of generating and maintaining the digital twin of the traffic environment in real-time by fusing higher level data of multiple sensors (deployed either in the infrastructure or in intelligent vehicles), thus providing object detections of higher reliability and at the same time extending the sensing range. Besides giving a general idea on cooperative perception we have also introduced the key building blocks of this system including the calibration, 3D detection, tracking, fusion, data synchronization, communication and visualization. In case of time-critical components we have also presented the underlying algorithms and pointed out the relevant implementation details, as well. The functional prototype of the proposed system has also been created and tested under real circumstances on-line. We have demonstrated a single service of the proposed perception system, namely the real-time visualization of the generated digital twin of the environment including pedestrians as dynamic objects of interest communicated using standard SENSORIS message formats over 5G or DSRC to the central server. The system can further be extended to support other type of objects, as well such as cars, bicyclist, etc. Besides the digital twin generation a broad range of new derivative services can be facilitated, as well, including cloud-assisted and cloud-controlled ADAS functions, various analytics for traffic control, etc., which are subjects of further research. We have also shown that the proposed perception system is able to operate in real-time, meaning that an overall latency of less than 100ms has been achieved. As already stated, we envision this prototype system as part of a larger network of local information processing and integration nodes, where the logically centralized digital twin is maintained in a physically distributed edge cloud in real-time.

We have encountered several noteworthy practical questions and lessons during the implementation which led to the establishment of certain best practices that can not be treated adequately within the bounds of this paper, but can be at least mentioned. Some of them include considering sensor latencies, triggering simultaneous snapshots and associating data from sensors with different frequencies. There are effects related to vehicle movement during a full LiDAR rotation. There are problems with creating a perfectly flat and orthogonal calibration points layout in the field. As already mentioned in some detail, GPS time based inter-platform synchronization was a cardinal issue. Detection can suffer from all the problems inherent in deep learning systems: unfamiliar lighting, background, or anything that takes the input image beyond the domain and distribution the neural network was trained for can influence the algorithm adversely. Of course, deep learning models are also susceptible to deliberate adversarial attacks like "invisibility T-shirts" [28] on pedestrians, etc. Foreground clutter in chest height (even a stretched out hand) can destabilize our LiDAR-reliant raw fusion method. DSRC communication tends to break down in the presence of obscuring objects: the installation height and placement of on-board/road-side communication units is crucial, communication hand-off between moving vehicle platforms and stationary road side units has to be solved. On the server side, managing the spatially distributed digital twin across several edge computing nodes and their overlapping areas of responsibility is a theoretical problem we are currently investigating. Practical considerations like system security, authentication, authorization and information integrity are undeniably safety critical issues that must be tackled before industrial application. So is the adherence to automotive standards like ASIL D and the use of provably real-time hardware and software systems that come with industrial-grade guarantees. Despite numerous challenges, technological enablers like cheap LiDAR-s, powerful deep learning and ubiquitous 5G are making the road towards cooperative perception services more attainable by the day.

Author Contributions: Conceptualization, Z.S. and V.T.; methodology, V.T., A.R. and V.R.; sensor calibration, A.R., M.C. and A.K.; platform setup: A.R., V.R., Z.V. and M.C.; platform software: data acquisition, object detection, A.R. and M.C.; platform software: tracking, Z.V.; communication: Z.P. and V.R.; central software: stream processing and local area fusion, V.R.; simulation and visualization, M.S. and B.V.; writing—original draft preparation, A.R. and V.R.; writing—eview and editing, A.R. and V.R.; supervision, project administration, V.T.; funding acquisition V.T. and Z.S. All authors have read and agreed to the published version of the manuscript.

Funding: The research reported in this paper and carried out at the Budapest University Technology and Economics has been supported by the National Research Development and Innovation Fund (TKP2020 National Challenges Subprogram, Grant No. BME-NC) based on the charter of bolster issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology. In addition the research was supported by the National Research, Development and Innovation Office through the project "National Laboratory for Autonomous Systems" under Grant NKFIH-869/2020. Funder NKFIH Grant Nos.: TKP2020 BME-NC and NKFIH-869/2020.

Acknowledgments: The research reported in this paper and carried out at the Budapest University of Technology and Economics has been supported by the National Research Development and Innovation Fund (TKP2020 National Challenges Subprogram, Grant No. BME-NC) based on the charter of bolster issued by the National Research Development and Innovation Office under the auspices of the Ministry for Innovation and Technology. In addition the research was supported by the Ministry of Innovation and Technology NRDI Office within the framework of the Autonomous Systems National Laboratory Program.

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript, or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

4G/5G	Fourth/fifth generation technology standard for broadband cellular networks
ADAS	Advanced Driver-Assistance Systems
API	Application Programming Interface
CAV	Connected and Autonomous Vehicle
CPU	Central Processing Unit
dGPS	Differential Global Positioning System
DSRC	Dedicated Short-Range Communications
FoV	Field of View
GNN	Global Nearest Neighbor
GNSS	Global Navigation Satellite System
GPS	Global Positioning System
GPU	Graphics Processing Unit
HAD	Highly Automated Driving
HD map	High Definition map
I2V	Infrastructure to Vehicle
IMM filter	Interacting Multiple Model filter
IMU	Inertial Measurement Unit
ITS	Intelligent Transportation System
JVM	Java Virtual Machine
LiDAR	Light Detection and Ranging (sensor)
MAC	Medium Access Control (sublayer of the data link layer in the OSI networking model)
NTP	Network Time Protocol
PPS	Pulse Per Second (synchronization signal)
PTP	Precision Time Protocol
R&D	Research and Development
RAID	Redundant Array of Independent Disks
S2T	Central tracking, aka. sensor-to-track fusion or source-to-track fusion
SciL	Scenario in the Loop
SUID	System-wide Unique Identifier
T2T	Track-to-track fusion
TCP	Transmission Control Protocol
UTM	Universal Transverse Mercator
UUID	Universally Unique Identifier
V2V	Vehicle to Vehicle
WGS84	World Geodetic System 1984
WiFi	Wireless Fidelity (network protocol family)
YOLO	You Only Look Once (object detection system)

References

1. Krämmer, A.; Schöller, C.; Gulati, D.; Knoll, A. Providentia—A large scale sensing system for the assistance of autonomous vehicles. *arXiv* **2019**, arXiv:1906.06789.
2. Gabb, M.; Digel, H.; Muller, T.; Henn, R.W. Infrastructure-supported perception and track-level fusion using edge computing. In Proceedings of the IEEE Intelligent Vehicles Symposium, Paris, France, 9–12 June 2019; Institute of Electrical and Electronics Engineers Inc.: Piscataway, NJ, USA, 2019; pp. 1739–1745. [CrossRef]
3. Tsukada, M.; Oi, T.; Kitazawa, M.; Esaki, H. Networked roadside perception units for autonomous driving. *Sensors* **2020**, *20*, 5320. [CrossRef] [PubMed]
4. Wang, Z.; Liao, X.; Zhao, X.; Han, K.; Tiwari, P.; Barth, M.J.; Wu, G. A Digital Twin Paradigm: Vehicle-to-Cloud Based Advanced Driver Assistance Systems. In Proceedings of the IEEE Vehicular Technology Conference, Antwerp, Belgium, 25–28 May 2020; [CrossRef]
5. Kobayashi, H.; Han, K.; Kim, B. Vehicle-to-Vehicle Message Sender Identification for Co-Operative Driver Assistance Systems. In Proceedings of the 2019 IEEE 89th Vehicular Technology Conference (VTC2019-Spring), Kuala Lumpur, Malaysia, 28 April–1 May 2019; pp. 1–5. [CrossRef]
6. Liu, Y.; Wang, Z.; Han, K.; Shou, Z.; Tiwari, P.; Hansen, J.H. Sensor Fusion of Camera and Cloud Digital Twin Information for Intelligent Vehicles. *arXiv* **2020**, arXiv:2007.04350.

7. Shan, M.; Narula, K.; Wong, R.; Worrall, S.; Khan, M.; Alexander, P.; Nebot, E. Demonstrations of cooperative perception: Safety and robustness in connected and automated vehicle operations. *Sensors* **2020**, *21*, 200. [CrossRef] [PubMed]
8. Tihanyi, V.; Tettamanti, T.; Csonthó, M.; Eichberger, A.; Ficzere, D.; Gangel, K.; Hörmann, L.B.; Klaffenböck, M.A.; Knauder, C.; Luley, P.; et al. Motorway Measurement Campaign to Support R&D Activities in the Field of Automated Driving Technologies. *Sensors* **2021**, *21*, 2169. [CrossRef] [PubMed]
9. Szalay, Z.; Szalai, M.; Tóth, B.; Tettamanti, T.; Tihanyi, V. Proof of concept for Scenario-in-the-Loop (SciL) testing for autonomous vehicle technology. In Proceedings of the 2019 IEEE International Conference on Connected Vehicles and Expo (ICCVE), Graz, Austria, 4–8 November 2019; pp. 1–5.
10. Szalay, Z.; Hamar, Z.; Simon, P. A multi-layer autonomous vehicle and simulation validation ecosystem axis: Zalazone. In *International Conference on Intelligent Autonomous Systems*; Springer: Berlin/Heidelberg, Germany, 2018, pp. 954–963.
11. Zhao, M.; Mammeri, A.; Boukerche, A. Distance measurement system for smart vehicles. In Proceedings of the 2015 7th International Conference on New Technologies, Mobility and Security (NTMS), Paris, France, 27–29 July 2015; pp. 1–5. [CrossRef]
12. Zhang, Z. A flexible new technique for camera calibration. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1330–1334. [CrossRef]
13. Wang, W.; Sakurada, K.; Kawaguchi, N. Reflectance Intensity Assisted Automatic and Accurate Extrinsic Calibration of 3D LiDAR and Panoramic Camera Using a Printed Chessboard. *Remote Sens.* **2017**, *9*, 851. [CrossRef]
14. Daniilidis, K. Hand-Eye Calibration Using Dual Quaternions. *Int. J. Robot. Res.* **1999**, *18*, 286–298. [CrossRef]
15. Vyacheslav, I.V.; Illya, E.K.; Irina, P.C. Accurate Time Synchronization for Digital Communication Network. In Proceedings of the 2007 17th International Crimean Conference—Microwave Telecommunication Technology, Sevastopol, Ukraine, 10–14 September 2007; pp. 259–260. [CrossRef]
16. Bochkovskiy, A.; Wang, C.Y.; Liao, H.Y.M. YOLOv4: Optimal Speed and Accuracy of Object Detection. *arXiv* **2020**, arXiv:2004.10934.
17. Hartley, R.; Zisserman, A. *Multiple View Geometry in Computer Vision*, 2nd ed.; Cambridge University Press: Cambridge, UK, 2004; [CrossRef]
18. Yilmaz, A.; Javed, O.; Shah, M. Object tracking: A survey'ACM computing surveys (CSUR). *ACM Comput. Surv.* **2006**, *38*, 13. [CrossRef]
19. Labbe, R. Kalman and bayesian filters in python. *Chap* **2014**, *7*, 246.
20. Chong, C.Y. Tracking and data fusion: A handbook of algorithms (bar-shalom, y. et al; 2011)[bookshelf]. *IEEE Control. Syst. Mag.* **2012**, *32*, 114–116.
21. Blom, H.; Bar-Shalom, Y. The interacting multiple model algorithm for systems with Markovian switching coefficients. *IEEE Trans. Autom. Control* **1988**, *33*, 780–783. [CrossRef]
22. Blackman, S.S.; Popoli, R. *Design and Analysis of Modern Tracking Systems*; Artech House Radar Library: Boston, MA, USA; London, UK, 1999.
23. Munkres, J. Algorithms for the assignment and transportation problems. *J. Soc. Ind. Appl. Math.* **1957**, *5*, 32–38. [CrossRef]
24. Kuhn, H.W. The Hungarian method for the assignment problem. *Nav. Res. Logist. Q.* **1955**, *2*, 83–97. [CrossRef]
25. Abd El-Gawad, M.A.; Elsharief, M.; Kim, H. A comparative experimental analysis of channel access protocols in vehicular networks. *IEEE Access* **2019**, *7*, 149433–149443. [CrossRef]
26. Dinar, A.E.; Merabet, B.; Ghouali, S. NTP Server Clock Adjustment with Chrony. In *Applications of Internet of Things*; Springer: Berlin/Heidelberg, Germany, 2021; pp. 177–185.
27. Szalai, M.; Varga, B.; Tettamanti, T.; Tihanyi, V. Mixed reality test environment for autonomous cars using Unity 3D and SUMO. In Proceedings of the 2020 IEEE 18th World Symposium on Applied Machine Intelligence and Informatics (SAMI), Herlany, Slovakia, 23–25 January 2020; pp. 73–78. [CrossRef]
28. Xu, K.; Zhang, G.; Liu, S.; Fan, Q.; Sun, M.; Chen, H.; Chen, P.Y.; Wang, Y.; Lin, X. Adversarial T-shirt! Evading Person Detectors in A Physical World. *arXiv* **2020**, arXiv:1910.11099.