A Framework for Robust Remote Driving Strategy Selection

Michael Klöppel-Gersdorf[a] and Thomas Otto
Fraunhofer IVI, Fraunhofer Institute for Transportation and Infrastructure Systems, Dresden, Germany

Keywords: Driving Strategy Selection, Yard Automation, Particle Filter, Robust Optimization, Valet Parking, V2X, IEEE 802.11p.

Abstract: In this paper, a framework for assisting Connected Vehicle (CV) is proposed, with the goal of generating optimal parameters for existing driving functions, e.g., parking assistant or Adaptive Cruise Control (ACC), to allow the CV to move autonomously in restricted scenarios. Such scenarios encompass yard automation as well as valet parking. The framework combines Model predictive control (MPC) with particle filter estimators and robust optimization.

1 INTRODUCTION

Remote control of Connected Vehicles (CVs) is an active topic of research, spanning questions from generating feasible movements, e.g., in yard automation (Belov et al., 2021), to the efficient transmission of the calculated maneuvers (Tsukada et al., 2020) as well as, of course, generating non-intersecting (in space-time) trajectories (Scheffe et al., 2021). The goal of this paper is to take a step back from the newest developments and to implement automatic driving capabilities on an existing vehicle using existing technology, which requires to retrofit the vehicle with communication capabilities as well as gaining access to driving functions. A centralized approach is used as most vehicles will not provide the necessary computation power. Here, the goal is not to completely remote control the vehicle (this might be difficult due to latency reasons and missing sensors), but rather to generate useful set points for the local assistance functions, called driving strategies (Manzinger et al., 2017). Following these strategies is then left to the vehicle. A robust optimization approach is used to account for the inherent uncertainties when predicting the future movement of vehicles. The framework is not intended to provide general automated driving capabilities, rather it is mainly concerned with restricted areas like parking lots or yards, where the driver can drop of the vehicle at an entry point. As such, we are commonly dealing with uninhabited vehicles.

The paper is organized as follows: In the next section preliminaries and requirements for employing the presented approach are discussed, whereas the third section introduces the framework, which is used in a case study in Section 4. The final section concludes the paper.

[a] https://orcid.org/0000-0001-9382-3062

2 PRELIMINARIES

In this section, the necessary inputs, their corresponding latencies and minimum requirements for the proposed framework are discussed.

2.1 Minimum Requirements on the Vehicle under Test (VuT) and Operating Area

In order to obtain driving strategy advice from the proposed framework, the vehicle under consideration (called VuT throughout the remainder of the paper) needs to be connected, using either 802.11p, Cellular V2X (C-V2X) or other means of connection, e.g., using Long Term Evolution (LTE) or 5G. Any other traffic participant (called target hereafter) needs either to be able to use the same communication means as the VuT to broadcast its position or needs to be detected by the VuT or by infrastructure sensors. This limits the approach mainly to restricted areas, like industrial yards or parking lots, since it is much easier to monitor such areas. Furthermore, the maximum number of participants is also much more limited. For driving strategy advice, the vehicle must be able to carry out at least one of longitudinal (e.g., Adaptive Cruise

Klöppel-Gersdorf, M. and Otto, T.
A Framework for Robust Remote Driving Strategy Selection.
DOI: 10.5220/0011088900003191
In *Proceedings of the 8th International Conference on Vehicle Technology and Intelligent Transport Systems (VEHITS 2022)*, pages 418-424
ISBN: 978-989-758-573-9; ISSN: 2184-495X

Control (ACC)) or lateral (e.g., lane keeping) control or both (e.g., parking assistant). Of course, an external interface to these functions must exist, at least temporarily (based, e.g., on location). One way of realizing this access is the usage of openpilot[1]. In addition, the VuT should have sufficient sensors to detect situations of imminent danger (e.g., front crash assistant) as the framework might be too slow to react under such circumstances due to message latency.

2.2 Vehicle-to-Everything (V2X) Communication

The main source of position information is position data broadcasted via V2X using the Cooperative Awareness Message (CAM) (ETSI EN 302 637-2 V1.3.2 (2014-11), 2014) or alternatively the Basic Safety Message (BSM). The following discussion is based on ETSI ITS-G5 communication, but similar values also hold true for other standards. Depending on the dynamics of the vehicle, the CAM will be generated every $0.1-1$ s. According to the specification, the position given in the message cannot be older than 50 ms. Transferring the message using our implementation of the V2X stack (Jacob et al., 2020) takes an additional $1-13$ ms, with an average of 9.1 ms (Strobl et al., 2019). One additional method of gaining position information is via the newly proposed Collective Perception Message (CPM) (ETSI TR 103 562 V2.1.1 (2019-12), 2019), which can be used to transmit lists of detected objects. It is important to keep in mind that measurements sent via CPM are relative to the sending station, i.e., besides the accuracy of the sensors, also the position accuracy of the sending station has to be taken into account.

2.3 Video Detection

The main approach to detect non-connected participants in the proposed framework relies on RGB and thermal video data (Klöppel-Gersdorf et al., 2021). Based on YoloV3 (Redmon and Farhadi, 2018), processing times between $0.3-2.5$ s for the whole chain from taking the image to the calculated position are achieved on a constrained computation device, the NVIDIA Jetson AGX Xavier. The former number corresponds to a simple projection of one single image of the detected object to the ground plane, whereas the latter corresponds to the fusion of multiple images and a 3D reconstruction. Other authors report similar numbers, e.g., (Ortiz Castelló et al., 2020), where the detection process for a single images takes a little less than 0.1 s. Depending on the scenario, more accurate processing methods can be applied, e.g., the algorithm presented in (Ihrke, 2018).

[1]https://github.com/commaai/openpilot

2.4 Additional Means of Object Detection

Although not considered here, several other information sources could be included in the framework, e.g., Lidar detection (Barea et al., 2018) or the usage of location tags based on, e.g., Zigbee (Kuo et al., 2010). In this case, objection fusion as described in (Barea et al., 2018; Liang et al., 2019) might be necessary.

3 FRAMEWORK

The framework introduced here shares the ideas of (Stahl and Hauth, 2011) and (Blackmore, 2006), using Model predictive control (MPC) and the particle filters, not only to determine the current state but also to predict future states. But instead of calculating different optimal control values depending on the evolution of the underlying process only one control variable per time step is determined, which satisfies the given constraints for all sampled realizations of the given process, which is an application of the scenario approach to robust optimization (Calafiore and Campi, 2006). On the other hand, the problem solved at every step of the MPC scheme can also be considered a special case of (Blackmore, 2006), where the probability of violating the constraints is required to be zero.

3.1 Particle Filter

A particle filter (Gordon et al., 1993) with systematic resampling (Douc and Cappé, 2005) is used to estimate the current location of the VuT as well as the target vehicles. A separate particle filter is employed for each tracked entity. The first time an entity is detected, an initial distribution to sample from has to be chosen. As this choice depends on the internal state used within the particle filter, only few details can be given here. The minimum information tracked by the particle filter is the current position of the vehicle, therefore, it is useful to initialize the position information part of the particles with the first measured position plus some Gaussian noise with zero mean and variance according to the accuracy of the sensor. Particles are then propagated using a predefined movement model with some additional random noise. In the case of the VuT, this propagating function also considers control actions determined before,

i.e., for a single particle x_n^{VuT}

$$x_{n+1}^{VuT} = f(x_n^{VuT}, u_n, \xi_n, \Delta t_n^{track,VuT}), \quad (1)$$

where $u_k \in \mathcal{U} \subset \mathbb{R}^{n_u}$ are the control variables, $x_n^{VUT} \in \mathbb{R}^{n_x}$ are the state variables of the VuT, ξ_n are stochastic components with known distribution, $\Delta t_n^{track,VuT}$ is the time step since the last update and the function f describes the movement. Note, that there are no smoothness assumptions on this function. The propagation function for particles belonging to the additional targets $t \in \mathcal{T}_n$, where $\mathcal{T}$ is an index set of the currently existing target vehicles at time step n, is similar to the definition above but does not include controls

$$y_{n+1}^t = g(y_{n+1}^t, \eta_n^t, \Delta t_n^{track,t}), \quad (2)$$

where η_n^t are stochastic components with known distribution. The particle filter employed follows the usual steps of the particle filtering algorithm, i.e., sampling the new population of particles using the previous sampled particles and the movement model, adapting the weights given new position measurements and a possible resampling. The number of particles used per filter and time step is denoted by $n_{particles}$ and typically ranges between 100 –1000, depending on the underlying model. If only a restricted driving area is given (as in the case study below, with large non-drivable areas), this information can also be included when adapting the weights. This is even possible when using the particle filter for propagation into the future. As mentioned above, the process of tracking the single participants is completely asynchronous. Therefore, as starting point for the optimization process, the particles are propagated to one single future time point, which is defined by the minimum amount of time required to generate the solution and transmit it to the VuT. Here, particles are only reweighed by the drivable area criterion as no measurements exist. As mentioned before, different detectors need different computation times to generate a position information for a given object, e.g., GPS measurements are faster than information gained via video detection. This can lead to asynchronous and non-sequential data acquisition. Especially if the slower sensor gives a very accurate position information, it might be beneficial to recalculate the particle filter chain including the newly obtained information. Therefore, older states of the filter as well as the already obtained measurements have to be saved.

3.2 Robust Optimization

Borrowing the notation from (Stahl and Hauth, 2011), the following stochastic optimization problem is to be solved:

$$\begin{aligned}
\min_{u_k,\ldots,u_{k+T_p}} \quad & J(x_k, u_k, \ldots, u_{k+T_p}, T_p), && (3)\\
s.t. \quad & X_{n+1}^{VuT} = f(X_n^{VuT}, u_n, \xi_n), && (4)\\
& Y_{n+1}^t = g(Y_{n+1}^t, \eta_n^t), && (5)\\
& OS(x_n^{VuT}) \cap OS(y_n^t) = \emptyset, && \\
& x_n^{VuT} \in X_n^{VuT}, \quad y_n^t \in Y_n^t && (6)\\
& t \in \mathcal{T}, && (7)\\
& u_n \in \mathcal{U}, && (8)\\
& n = k, \ldots, k+T_p, && (9)
\end{aligned}$$

where the variables are defined as above in the particle filter definition and X^{VuT} and Y^t describe the sets of all particles belonging to either the VuT or the targets $t \in \mathcal{T}$, respectively. The function $OS(\cdot)$ describes the space occupied by a vehicle with a given state, i.e., equation (6) describes a no crash condition. The functions f and g, representing the dynamics of the VuT as well as the target vehicles, are the same as in the description of the particle filter above, with the only difference that the time step Δt^{prop} is now the same for all vehicles as we are propagating into the future. Finally, the objective function J has the form

$$\begin{aligned}
J(x_k, u_k, \ldots, u_{k+T_p}, T_p) \quad = \quad & \sum_{n=k}^{k+T_p} \|u_n - u_{n-1}\|_Q^2 \\
+ \quad & \sum_{n=k+1}^{k+T_p} \|s_n - x_n\|_R^2, \quad (10)
\end{aligned}$$

where the norms are weighted Euclidian norms with weighting matrices R and Q, and $s_n \in \mathbb{R}^{n_x}$ are desired set points. Evaluating the constraints (6) requires $|\mathcal{T}| n_{particles}^2$ computations, which might be computationally challenging. For an easier computation with only $|\mathcal{T}|$ checks,

$$\forall t \in \mathcal{T} : conv(OS(X_n^{VuT})) \cap conv(OS(Y_n^t)) = \emptyset \quad (11)$$

can be used instead of (6). Here, $conv(\cdot)$ describes the convex hull. Although this reduces the number of constraints to be checked significantly, this might lead to a more conservative approach.

4 CASE STUDY

This use case is inspired by a real-world example. Therefore, we want to mimic these circumstances as much as possible, i.e., choosing the same venue, adapting noise parameters to real-world sensor data and reflecting the existing control actions. In this special case, this means that we can only influence the velocity of the VuT via Maneuver Coordination

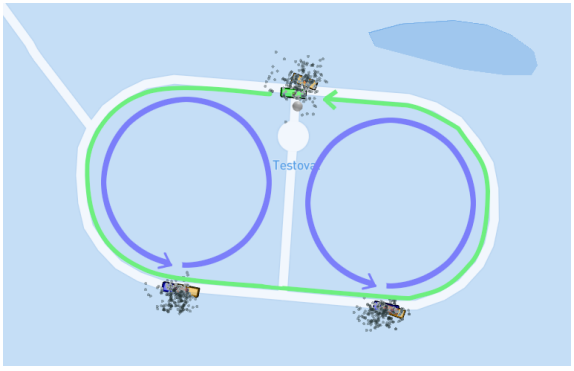

Figure 1: Simulation setup of the case study extracted from Streetscape.gl (Uber ATG and VIS.GL, 2020) visualization. Shown are the VuT in green and the two target vehicles in purple as well as their corresponding driving trajectories. Additionally, in orange are the position as given by the simulated GPS sensor. The point clouds represent position hypotheses generated by the particle filter.

Message (MCM) (Auerswald et al., 2019) but cannot change the trajectory. The whole simulation scenario is shown in Fig. 1 and consists of the VuT and two other target vehicles. The VuT initially moves at 50 km/h, whereas the target vehicles move with 18 km/h. Trajectories as used in the scenario are also shown in Fig. 1. These trajectories are derived using a bicycle model (Althoff et al., 2017), but are not known to the optimization algorithm.

For all three vehicles the states of the particle filter consist of $\bar{x} = (x, y, v, \theta)$, where x, y are coordinates in a local Cartesian coordinate system measured in meters, v is the velocity measured in meters per second and θ is the heading measured in radians. For the initialization of the particle filter, x and y are sampled from a Gaussian distribution with the first measurement received used as mean and $\Sigma = 4.5\,\mathrm{m} \times I_{2x2}$, which corresponds to the Circular Error Probable (CEP) of the GPS chip used (u blox, 2019). The velocity is sampled from a normal distribution with zero mean and standard deviation $\sigma = 15\,\mathrm{m/s}$. Finally, the heading θ is sampled from a uniform distribution from 0 to 2π, i.e., no information about the initial direction of the vehicles is known. Similar to (Scheffe et al., 2021), quantized control values are employed, i.e., the acceleration of the VuT is controlled with $u_k \in \{-6\,\mathrm{m/s^2}, -3\,\mathrm{m/s^2}, 0\,\mathrm{m/s^2}, 3\,\mathrm{m/s^2}, 6\,\mathrm{m/s^2}\}$. In addition, constraints on the maximum and minimum speed of the VuT were considered, i.e., $0\,\mathrm{m/s} \leq v \leq 13.88\,\mathrm{m/s}$. To propagate the particles, a simple linear movement model is used, i.e.,

$$f(\bar{x}, u, \xi) = \bar{x} + \Delta t\,(v\cos(\theta), v\sin(\theta), u, 0) + \xi \quad (12)$$

and

$$g(\bar{y}, \eta) = \bar{y} + \Delta t\,(v\cos(\theta), v\sin(\theta), 0, 0) + \eta, \quad (13)$$

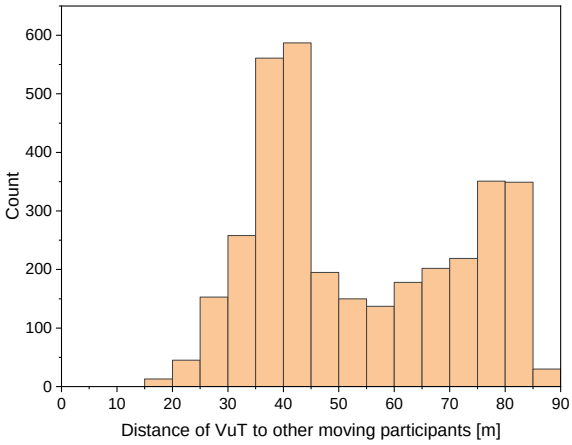

Figure 2: Histogram showing the distribution of distances of the VuT to the other participants over a simulation time of 171.4 s, with a 0.1 s resolution, i.e., 1714 total time steps. At no time, the distance is less than 15 m.

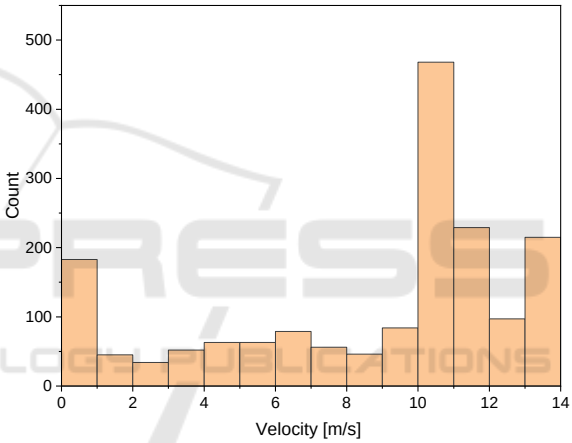

Figure 3: Histogram showing the velocity of the VuT over a simulation time of 171.4 s, with a 0.1 s resolution, i.e., 1714 total time steps. At several time steps the velocity drops to 0 m/s. Average velocity is 8.7 m/s.

where $\xi, \eta \sim \mathcal{N}(0, diag(0.02, 0.02, 1, 0.2))$. As the vehicles follow circular motions, their heading changes constantly. Therefore, CAM are generated at a frequency of 10 Hz. Correspondingly, $\Delta t^{track} \approx 0.1\,\mathrm{s}$.

A prediction horizon of $T_p = 3$ with $\Delta t^{prop} = 1\,\mathrm{s}$ is used. In (10), we choose $R = 1$ and $Q = diag(0, 0, 1, 0)$, i.e., only a set point for the target velocity of $s_j = s = (0, 0, 13.88\,\mathrm{m/s}, 0)$ (corresponding to 50 km/h) is given. The simulation and optimization uses a proof-of-concept implementation in Python using pfilter[2] with $n_{particles} = 200$ for each entity and time step, without any optimization like parallelization. To still achieve real-time performance, an aggressive branch-and-bound scheme (Frese et al.,

[2]https://github.com/johnhw/pfilter

Figure 4: Vehicle configuration at $t = 20\,\mathrm{s}$. Both vehicles are moving anti-clockwise as shown. At this time instance, the green VuT moves with 50 km/h whereas the violet participant moves at 18 km/h.

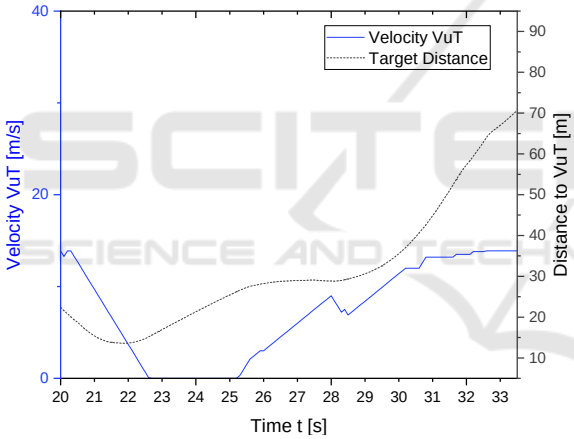

Figure 5: Distance to one of the targets and resulting velocity of the VuT after optimization. The initial configuration corresponding to $t = 20\,\mathrm{s}$ is shown in Fig. 4.

2010) is used. Additionally, variant (11) is used to ease the computational burden. Simulation results for a time horizon of 171.4 s can be found in Fig. 2–5. In Fig. 2, a histogram of the Euclidian distance of the VuT's center to the target vehicles centers is shown. Given the chosen vehicle dimensions, any distance less than 6 m could be considered hazardous and would possibly lead to a collision. As can be seen, the distances do not fall under the critical value, i.e., the algorithm was able to provide feasible driving strategies. As the histogram in Fig. 3 shows, the algorithm leads to lower VuT velocities over large parts of the scenario and even stopped the vehicle for a total of nearly 20 seconds. One instance of this happening is shown in Fig. 4 and 5. Here, Fig. 4 shows the vehicle configuration at $t = 20\,\mathrm{s}$ and Fig. 5 shows the VuT's velocity as well as the distance to one of the two participants. As the VuT gets closer to the other participant, the optimization framework advises sharp deceleration to avoid a collision, bringing the VuT to a full stop at $t = 22.6\,\mathrm{s}$. As the distance to the target grows larger again, the algorithm responds with accelerating the VuT at $t = 25.2\,\mathrm{s}$.

Overall, it can be found that the derived accelerations lead to a rather conservative behavior. Even in the worst case, occurring at $t = 21.4\,\mathrm{s}$ (compare Fig. 5), the distance to the target is 13.71 m, which is more than double of the critical distance. Furthermore, the VuT came to a halt at several instances. This is partly caused by the rather large CEP of the GPS chip when used without real-time kinematics, which leads to large variations when propagating the particles and is partly also a result of the chosen optimization scheme. An approach based on chance constraints (Blackmore, 2006) would possibly lead to a less conservative result. Nonetheless, one should keep in mind that no trajectory information was available to the algorithm, allowing for rather large deviations in the propagation algorithm. If these were available, e.g., via a future version of the MCM, and would be incorporated in the algorithm, better optimization results could be expected as the particle filter penalizes hypotheses deviating from the given trajectory and not only penalize hypotheses outside the drivable area as in the given case.

4.1 Scaling the Scenario

As mentioned above, the presented show case ran in real-time, i.e., every computation step took strictly less than 0.1 s on an Intel i7-10850H with 32 GiByte RAM using only one of the existing six processor cores (ignoring hyperthreading). In the following, we will discuss how enlarging the scenario (more targets and/or more VuTs) would influence computation time and how real-time performance could be achieved in these cases.

When increasing the number of targets, the computational load increases linearly, i.e., using parallel computing on the CPU would allow to handle up to 12 targets using the given hardware. Higher number of targets could be dealt with employing more capable hardware. In such cases it might be also beneficial to detect relevant targets, e.g., based on Euclidian distance or road geometry, and only include these into the optimization algorithm. While non-relevant targets still have to be tracked (as they could become relevant later on), no collision checks have to be car-

ried out in the optimization step. Generally, the propagation of particles is very suited for processing on graphic cards, since the same transformations is applied to a large number of different particles. Depending on the hardware employed, up to several hundred targets could be tracked in the same time it now takes to track the participants in the scenario presented above.

When increasing the number of VuTs, computational load very much depends on the specific scenario. In the best case, there is no interaction between VuTs, e.g., because of distance, and the optimization can be carried out for each VuT separately, leading only to a linear increase in computational demand. If there are interactions, two approaches come to mind. First, one could still optimize the VuTs separately and treat the other VuTs as targets. This leads to a super-linear increase in computational demand, since there are still n^{VuT} optimization problems to be solved, where n^{VuT} is the number of VuTs, but the number of targets is also increased by $n^{VuT} - 1$. While this could still be handled by parallelization, some situations might require joint optimization, i.e., cooperation between the vehicles, e.g., if two VuTs are approaching each other head on. In joint optimization, computation time very much depends on the optimization algorithm and the specific problem formulation. Above, a discrete optimization problem is formulated. Naively scaling this to several VuTs, would lead to an exponential increase in possible control values. While parallelization might allow to deal with up to three VuTs in joint optimization, more aggressive optimization routines would have to be employed (Tang, 1998), at the cost of finding only a reasonable but not the optimal solution.

The current implementation is in Python, further speed ups could be expected from switching to C++, although it is hard to quantify which speed ups can be achieved as some of the Python libraries already employ C/C++ code themselves.

5 CONCLUSIONS AND FUTURE WORK

In this paper, a framework for assisting CV with at least on driving function, e.g., ACC, was proposed. The approach only relies on position information derived via V2X messages or through external sensors like cameras and, therefore, can incorporate even a certain number of conventional vehicles or persons. Use case are mainly in yard automation or on other restricted zones, e.g., automatically moving freshly build vehicles from the conveyor belt to a parking area without the need for a human driver. A simulation based on a real-live scenario showed the feasibility of the framework. While the scenario ran in real-time without much need for optimization, larger scale problems could benefit from parallelization.

Applications in live traffic are also possible, but given the conservativeness of the presented approach, it would possibly take a long time to cross, for example, a non-signalized intersection. Approaches based on chance-constraints (Blackmore, 2006) could be a better choice in this case. One additional drawback of using the method in live traffic is the requirement to detect every traffic participant, which could require a large number of detectors. Finally, some situations could also be difficult to track with the particle filter, e.g., a person entering a vehicle or a passenger vehicle being loaded on a truck. Such scenarios would require additional information from the sensor system.

The simulations, so far, considered only sparsely populated scenarios. While the conservativeness of the algorithm would lead to longer driving times in comparison to a human driver, this is not a problem in such cases. Future work includes simulations of more densely populated uses cases to allow a measurement how performance is impacted by using the algorithm instead of a human driver.

ACKNOWLEDGEMENTS

This research is financially supported by the German Federal Ministry of Transport and Digital Infrastructure (BMVI) under grant numbers FKZ 45FGU141_B (DiSpoGo) and co-financed by the Connecting Europe, Facility of the European Union (C-ROADS Urban Nodes). We would like to thank Ina Partzsch for her valuable comments and suggestions.

REFERENCES

Althoff, M., Koschi, M., and Manzinger, S. (2017). Commonroad: Composable benchmarks for motion planning on roads. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 719–726. IEEE.

Auerswald, R., Dod, M., Franke, L., Fritzsche, R., Haberjahn, M., Jungmann, A., Klöppel-Gersdorf, M., Krems, J. F., Lorenz, S., Kreißig, I., et al. (2019). Heterogeneous infrastructure for cooperative driving of automated and non-automated connected vehicles. In *Smart Cities, Green Technologies and Intelligent Transport Systems*, pages 270–296. Springer.

Barea, R., Pérez, C., Bergasa, L. M., López-Guillén, E., Romera, E., Molinos, E., Ocana, M., and López, J. (2018). Vehicle detection and localization using 3d

lidar point cloud and image semantic segmentation. In *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, pages 3481–3486.

Belov, N., Barbosa, C. E. V., Keppler, F., Kolb, J., Nitzsche, G., and Wagner, S. (2021). Trucktrix® path-planning in the helyos operating system for yard automation. In *2021 IEEE 19th International Conference on Industrial Informatics (INDIN)*, pages 1–6. IEEE.

Blackmore, L. (2006). A probabilistic particle control approach to optimal, robust predictive control. In *AIAA Guidance, Navigation, and Control Conference and Exhibit*, page 6240.

Calafiore, G. and Campi, M. (2006). The scenario approach to robust control design. *IEEE Transactions on Automatic Control*, 51(5):742–753.

Douc, R. and Cappé, O. (2005). Comparison of resampling schemes for particle filtering. In *ISPA 2005. Proceedings of the 4th International Symposium on Image and Signal Processing and Analysis, 2005.*, pages 64–69. IEEE.

ETSI EN 302 637-2 V1.3.2 (2014-11) (2014). ETSI EN 302 637-2 V1.3.2 (2014-11) Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Part 2: Specification of Cooperative Awareness Basic Service. Standard, ETSI.

ETSI TR 103 562 V2.1.1 (2019-12) (2019). ETSI TR 103 562 V2.1.1 (2019-12) Intelligent Transport Systems (ITS); Vehicular Communications; Basic Set of Applications; Analysis of the Collective Perception Service (CPS); Release 2. Standard, ETSI.

Frese, C., Beyerer, J., and Huber, M. (2010). Cooperative motion planning using branch and bound methods. In *Proc. of the 2009 Joint Workshop of Fraunhofer IOSB and Institute for Anthropomatics, Vision and Fusion Laboratory (J. Beyerer and M. Huber, eds.), no. IES-2009-13*, pages 187–201.

Gordon, N. J., Salmond, D. J., and Smith, A. F. (1993). Novel approach to nonlinear/non-gaussian bayesian state estimation. In *IEE Proceedings F-radar and signal processing*, volume 140, pages 107–113. IET.

Ihrke, S. (2018). Precise edge tracking of vehicles using a static camera setup. In *2018 International Symposium on Intelligent Signal Processing and Communication Systems (ISPACS)*, pages 12–17.

Jacob, R., Anwar, W., Schwarzenberg, N., Franchi, N., and Fettweis, G. (2020). System-level performance comparison of ieee 802.11p and 802.11bd draft in highway scenarios. In *2020 27th International Conference on Telecommunications (ICT)*, pages 1–6.

Klöppel-Gersdorf, M., Trauzettel, F., Koslowski, K., Peter, M., and Otto, T. (2021). The fraunhofer ccit smart intersection. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1797–1802. IEEE.

Kuo, W.-H., Chen, Y.-S., Jen, G.-T., and Lu, T.-W. (2010). An intelligent positioning approach: Rssi-based indoor and outdoor localization scheme in zigbee networks. In *2010 International Conference on Machine Learning and Cybernetics*, volume 6, pages 2754–2759.

Liang, M., Yang, B., Chen, Y., Hu, R., and Urtasun, R. (2019). Multi-task multi-sensor fusion for 3d object detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Manzinger, S., Leibold, M., and Althoff, M. (2017). Driving strategy selection for cooperative vehicles using maneuver templates. In *2017 IEEE Intelligent Vehicles Symposium (IV)*, pages 647–654.

Ortiz Castelló, V., Salvador Igual, I., del Tejo Catalá, O., and Perez-Cortes, J.-C. (2020). High-profile vru detection on resource-constrained hardware using yolov3/v4 on bdd100k. *Journal of Imaging*, 6(12).

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Scheffe, P., Pedrosa, M. V., Flaßkamp, K., and Alrifaee, B. (2021). Receding horizon control using graph search for multi-agent trajectory planning.

Stahl, D. and Hauth, J. (2011). Pf-mpc: Particle filter-model predictive control. *Systems & Control Letters*, 60(8):632–643.

Strobl, S., Klöppel-Gersdorf, M., Otto, T., and Grimm, J. (2019). C-its pilot in dresden – designing a modular c-its architecture. In *2019 6th International Conference on Models and Technologies for Intelligent Transportation Systems (MT-ITS)*, pages 1–8.

Tang, Z. (1998). Time constrained optimization. In *Proceedings of the 37th IEEE Conference on Decision and Control (Cat. No.98CH36171)*, volume 4, pages 3899–3906 vol.4.

Tsukada, M., Oi, T., Kitazawa, M., and Esaki, H. (2020). Networked roadside perception units for autonomous driving. *Sensors*, 20(18).

u blox (2019). Product summary NEO-M8P series u-blox M8 high precision GNSS modules. https://www.u-blox.com/en/docs/UBX-15015836. Accessed: 2022-01-14.

Uber ATG and VIS.GL (2020). AVS project page. https://avs.auto. Accessed: 2020-12-15.