Design and Application of a Multirate Sensor ADAS system for IWM-EVs

Luuk van Sundert*, Yuki Hosomi†, BinhMinh Nguyen†, Tom Oomen*, Hiroshi Fujimoto†
*Mechanical Engineering, Control Systems Technology ,Eindhoven University of Technology, Eindhoven, Netherlands
†Department of Advanced Energy, The University of Tokyo, Kashiwa, Japan

***Abstract*—In-wheel motor electric vehicles (IWM-EVs) enable high-bandwidth control, but integrating these capabilities with adaptive cruise control (ACC) remains challenging due to sparse and delayed ADAS perception measurements. This paper presents a practical multirate sensing and control framework for LiDAR-based ACC in IWM-EVs. A lightweight LiDAR processing pipeline detects the lead vehicle at 10 Hz, after which a dual-rate Kalman filter with constant-delay compensation increases the effective estimation update rate and provides corrected relative-state estimates. These estimates are used by a discrete-time model predictive controller for longitudinal distance control. Real-vehicle experiments demonstrate stable and smooth vehicle-following behavior in both static and dynamic ACC scenarios.**

***Index Terms*—LiDAR perception, Delay compensation, Dual-rate Kalman filter, Model Predictive Control (MPC)**

I. Introduction

A. Background of this study

Electric vehicles (EVs) are widely regarded as a key technology for future sustainable transportation systems. In recent years, in-wheel motor electric vehicles (IWM-EVs) have attracted considerable attention thanks to their capability for independent wheel torque control with fast response and high precision. To fully exploit these advantages, advanced control strategies are required to guarantee safety, stability, and energy efficiency. Well studied control strategies are for example traction control [1], yaw moment control [2], and braking force distribution control [3]. However, integration with advanced driver assistance systems (ADAS), and in particular adaptive cruise control (ACC), remains challenging. The difficulty arises from the mismatch between the high feedback rate of the in-wheel motor encoders (1 kHz) and the relatively low rate of the perception sensors of the ADAS system (e.g. LiDAR and camera). Additionally, the ADAS pipeline typically introduces a substantial computation delay, resulting in measurements that are low-rate and delayed. Both the mismatch in control rate and the delay compensation occur from different sources and should be addressed with different mechanisms.

B. Two challenges to be solved by this study

In order to address these challenges, two key elements are required. First, a dedicated sensor processing and estimation framework should be designed that increases the effective feedback rate and compensates for the measurement delay by predicting the current relative states of preceding vehicles from low-rate, delayed detections. Secondly, the control architecture should be adjusted such that the ACC control can be seamlessly combined with other control strategies, such as driving force control. While the estimation and delay-compensation techniques used in this study are based on established methods from prior work [5], this paper extends and applies these methods to a practical IWM-EV adaptive cruise control scenario with sparse LiDAR sensing, fixed delay compensation and real-time longitudinal control on a real IWM-EV. The key contribution of this study therefore lies in the practical system-level integration and experimental validation of sensing, estimation, delay-compensation, and control methods for LiDAR-based ACC in IWM-EVs. To the authors' knowledge, such an experimentally validated integration for LiDAR-based IWM-EV ACC has not previously been reported.

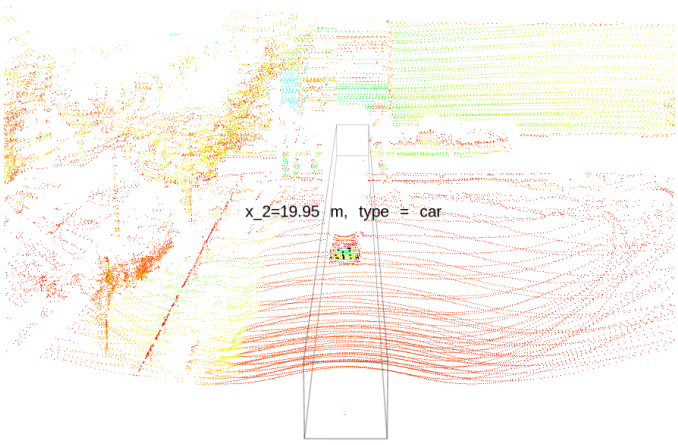


Fig. 1: Visualization of intensity of LiDAR data and detections resulting from detection algorithm, using [4].

II. Problem setting

For a LiDAR based ACC to work properly, accurate information of the relative dynamics between the host vehicle, hereafter referred to as the ego vehicle, and the lead vehicle is required. In this paper these relative motions in discrete time are described as in (1).

$$x_k = \begin{bmatrix} x_{\mathrm{k,rel}} \\ v_{\mathrm{k,rel}} \\ a_{\mathrm{k,rel}} \end{bmatrix} = \begin{bmatrix} x_{k,2} - x_{k,1} \\ v_{k,2} - v_{k,1} \\ a_{k,2} - a_{k,1} \end{bmatrix} \tag{1}$$

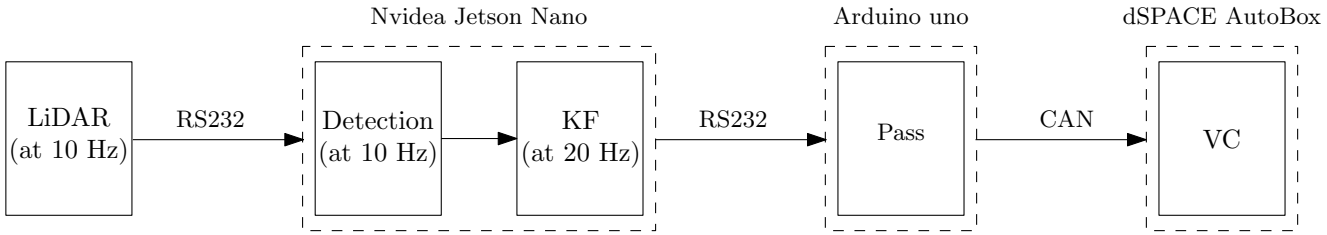


Fig. 2: Overview of the proposed sensor system including hardware, communication protocols, and publishing rates per process

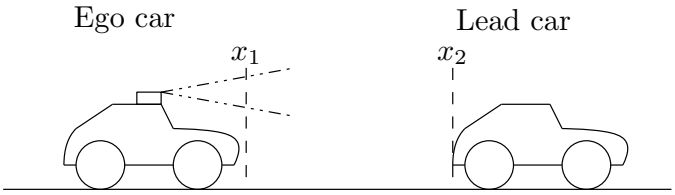


Fig. 3: Schematic drawing showing problem setting including ego and lead car

Where x_1, represents the front bumper of the ego car and x_2 the back of the leading car, as shown in Fig. 3. v_1, v_2, a_1, a_2 represent the absolute velocity and acceleration of both cars. For simplicity, only the longitudinal motion on a flat high-friction surface is considered.

III. System design and detection algorithm

A. Outline of sensor system design

In order to estimate the relative dynamics between both cars the sensor system schematically shown in Fig. 2 is designed. It consists of a Livox HAP LiDAR [6], an NVIDIA Jetson Nano Developer Kit [7], an Arduino UNO R4, and a dSPACE AutoBox [8]. The LiDAR driver publishes a ROS 2 point cloud topic [9] at 10 Hz, which serves as the input to the detection algorithm running on the Jetson.

B. Detection algorithm

The detection algorithm is responsible for detecting leading vehicles. For each incoming point cloud, the algorithm first extracts the x, y, and z coordinates and removes all points outside the region of interest (ROI), which in the ACC application is defined as the driving lane in front of the ego vehicle. To reduce computational load, the remaining points are downsampled using voxel downsampling [10]. The voxel size acts as a tunable parameter to balance computational cost and spatial resolution: smaller voxels keep more geometric detail at higher computational cost, while larger voxels improve processing speed at the cost of fine detail. After downsampling, ground points are removed using RANSAC plane segmentation [11], limited to 30 iterations to cap computation time. The remaining non-ground points are treated as candidate objects. Object extraction is performed using Euclidean clustering implemented in the Point Cloud Library (PCL) [10]. This method groups points that lie within a specified clustering tolerance of one another. A smaller tolerance yields finer segmentation and can separate smaller objects, whereas a larger tolerance merges nearby structures into fewer, larger clusters. Clusters containing fewer points than a predefined threshold are discarded as noise. For each remaining cluster, a 3D bounding box is estimated to approximate the object's dimensions and centroid. If the bounding box dimensions fall within predefined bounds corresponding to typical passenger car sizes, the cluster is classified as a vehicle. For each detected vehicle, the distance to the ego car is obtained from the closest point in the cluster. To mitigate spurious measurements due to stray points, the closest $5\,\%$ of points (filtered by x-coordinate) is discarded before computing this distance. The vehicle with the smallest resulting distance is selected as the lead vehicle. The measured distance to the lead vehicle, together with a status flag indicating whether a valid vehicle is detected or the measurement is out of range, is transmitted via RS-232 to an Arduino. The Arduino runs a simple pass-through program that forwards the data in two CAN messages to the dSPACE Autobox, where it is used as input by the controller, which is further discussed in section V. However, due to the low LiDAR frame rate and the significant delay, control reference tracking performance will be limited. To improve the performance an observer is designed.

IV. Observer design and delay compensation

A. Kalman Filter design

In order to deal with the sparse LiDAR data and delay, a dual rate KF with delay compensation framework is used, as described in [5]. To apply the framework to ACC a one dimensional constant-acceleration model described in (2) is used.

$$x_{k+1} = Fx_k + w_k, \qquad w_k \sim \mathcal{N}(0, Q), \tag{2}$$

$$z_k = Hx_k + v_k, \qquad v_k \sim \mathcal{N}(0, R), \tag{3}$$

Where z_k is equal to the measurement update, $H = \begin{bmatrix}1 & 0 & 0\end{bmatrix}$ and F is described by (4)

$$F = e^{A_c T} = \begin{bmatrix}1 & T & \frac{1}{2}T^2 \\ 0 & 1 & T \\ 0 & 0 & 1\end{bmatrix}, \tag{4}$$

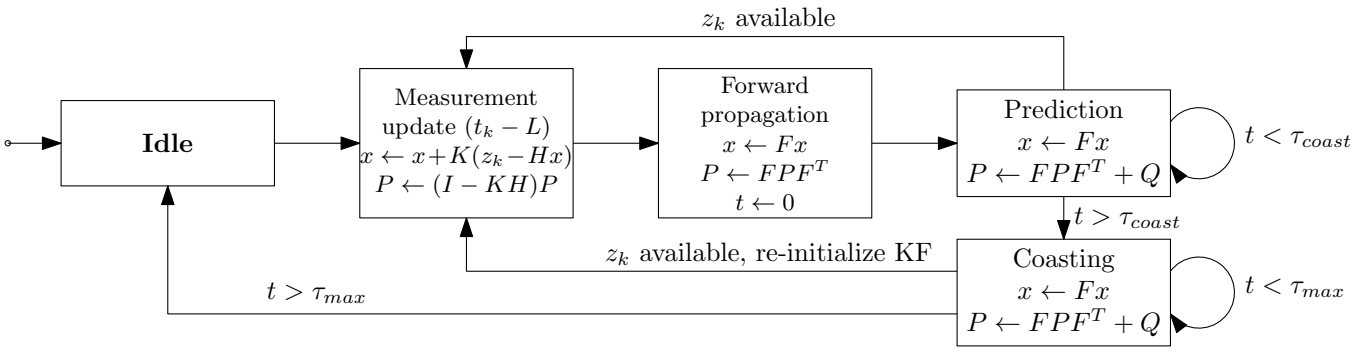


Fig. 4: Schematic overview of KF architecture including delayed updates and a time based state machine to re-initialize KF.

At each discrete step with no new LiDAR measurement, the filter predicts.

$$\hat{x}_{k|k-1} = F\hat{x}_{k-1|k-1}, \tag{5}$$

$$P_{k|k-1} = FP_{k-1|k-1}F^\top + Q, \tag{6}$$

and, when a measurement is available, updates as:

$$\tilde{z}_k = z_k - H\hat{x}_{k|k-1} \tag{7}$$

$$S_k = HP_{k|k-1}H^\top + R \tag{8}$$

$$K_k = P_{k|k-1}H^\top S_k^{-1} \tag{9}$$

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k\tilde{z}_k \tag{10}$$

$$P_{k|k} = (I - K_kH)P_{k|k-1}(I - K_kH)^\top + K_kRK_k^\top \tag{11}$$

The estimation is initialized with $\hat{x}_{0|0} = [\,z_0,\ 0,\ 0\,]^\top$ and $P_{0|0} = \mathrm{diag}([\,0.04,\ 9,\ 4\,])$, corresponding to $\sigma_{d,0} = 0.2\,\mathrm{m}$, $\sigma_{v,0} = 3\,\mathrm{m/s}$, $\sigma_{a,0} = 2\,\mathrm{m/s^2}$. Measurement noise is set to $R = \sigma_r^2$ with $\sigma_r = 0.15\,\mathrm{m}$. Where the measurement noise is based on the convenience measurement using a static lead car at a distance of 50 m. Additionally, to prevent spurious LiDAR detections to affect the control a gating technique is used to reject outliers.

B. Delay compensation

Due to the delay, a new measurement z_k that arrives, corresponds to a past state x_{k-L}. Where L corresponds to a fixed amount of samples delay $L = \mathrm{round}(\tau_d/T)$. To compensate for the delay first the delay is measured. The measurement is performed by comparing the timestamp of the LiDAR with the timestamp of the current time before sending the measurement z_k to the controller. For a 10 s interval this results in a mean delay of 171 ms and a standard deviation of 0.81 ms, shown in Fig. 6. The low standard deviation indicates that the delay remains highly consistent during operation, justifying the use of a fixed-delay compensation approach. fixed-delay compensation can be implemented using the following approach. The filter maintains a buffer of the last $L+1$ predicted priors $(\hat{x}_{j|j-1}, P_{j|j-1})$. The buffer of predicted priors, makes it possible to apply the update not to the last prior, but to the prior to which it belongs, as shown in (12)

$$y = z_k - H\hat{x}_{k-L|k-L-1} \tag{12}$$

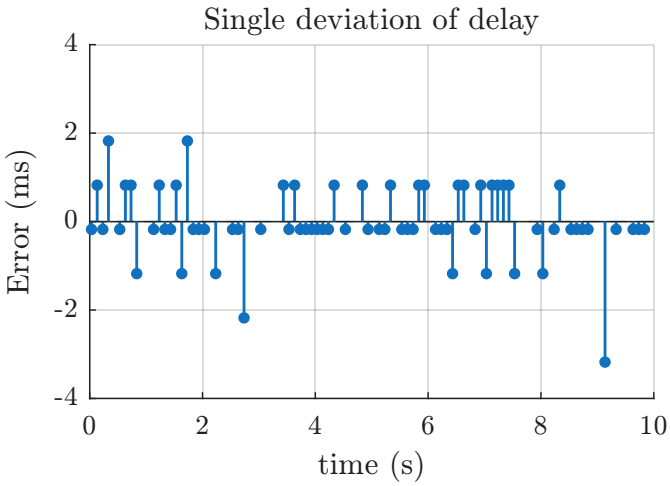


Fig. 5: Error of the delay measured over time, created using timestamp data of LiDAR and ROS2 time.

$$\hat{x}_{k-L|k-L} = \hat{x}_{k-L|k-L-1} + Ky \tag{13}$$

$$P_{k-L|k-L} = (I - KH)P_{k-L|k-L-1}(I - KH)^\top + KRK^\top. \tag{14}$$

With the updated state at x_{k-L}, the results is propagated forward through the model to the current time, as in (15)

$$\hat{x}_{j+1|k} = F\hat{x}_{j|k}, \tag{15}$$

$$P_{j+1|k} = FP_{j|k}F^\top + Q, \tag{16}$$

This yields a delay-compensated estimate $\hat{x}_{k|k} = [\hat{x}_{\mathrm{rel}},\ \hat{v}_{\mathrm{rel}},\ \hat{a}_{\mathrm{rel}}]^\top$ that represents the best estimate of the current system state, even though the measurement itself was acquired τ_d seconds earlier.

C. Robustness against Lane switching and cross traffic

Finally, the KF should be adjusted to work in scenarios with switching lead vehicles and non idealized situations. In other words it should stop predicting or re-initialize in case a leading vehicle moves out of lane or switch to a new leading

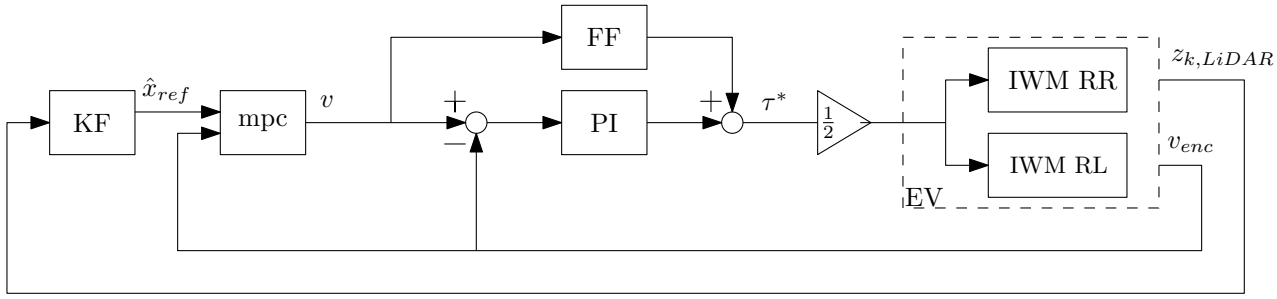


Fig. 6: Schematic overview of control architecture, consisting of MPC loop and Velocity control loop. RR represents the rear right and RL represents rear left

vehicle. To account for these changes the state machine shown in Fig. 4 is used. When initializing the KF, it starts in the Idle state, meaning that there is no lead car present. Once a car is detected it initializes the KF using the first measurement. After, it moves to the prediction state where it will predict until a new measurement comes. However, if no new measurement arrives within the maximum coasting time (τ_{coast}), it will move to the coasting state. In the coasting state it will keep predicting, however once it receives a new measurement it re-initialize the KF, instead of performing an update. The re-initialization prevents the update from being rejected due to the set bound. If instead, there is no new update within the maximum valid measurement time (τ_{max}), it will move again to the idle state, meaning that there is no car in front. In this way the state machine ensures that if a car moves out of the region of interest only for a short time, it will keep tracking it, however if it moves out completely it will move back to the idle state due to the time constraints.

V. Controller design for ACC application

In order to combine well known traction control techniques together with ACC, the control architecture is split in two loop. The first loop is a conventional velocity control loop with a feedforward and feedback controller, which is based on [12]. In this loop the DFC can be easily implemented by adding the DFC in front of the in-wheel motor, as described in [13]. The second loop, is responsible for generating the reference signal for the velocity control loop. The reference is generated using a MPC, that uses the KF estimate x_{rel} as input. The MPC scheme is implemented in discrete time with a sampling period of $T_s = 0.05$ s and a prediction horizon of N_p. The control input u_k represents the commanded longitudinal acceleration of the ego vehicle. The constant acceleration model is described in (17)

$$x_k = \begin{bmatrix} x_{relk} \\ v_{relk} \end{bmatrix}, \quad x_{k+1} = Ax_k + Bu_k, \tag{17}$$

with

$$A = \begin{bmatrix} 1 & T_s \\ 0 & 1 \end{bmatrix}, \quad B = \begin{bmatrix} \frac{1}{2}T_s^2 \\ T_s \end{bmatrix}. \tag{18}$$

To achieve ACC the desired distance is defined using a constant time-gap policy:

$$d_{\text{ref}} = d_0 + T_h\, v_{\text{ego}}, \tag{19}$$

where d_0 is the standstill distance and T_h the desired time gap. To achieve the desired state the optimization problem described in (20), is used.

$$\begin{aligned} \min_{\mathbf{u}} \quad & \sum_{i=0}^{N_p-1} \left(\|x_{k+i|k} - d_{\text{ref}}\|_Q^2 + \|u_{k+i|k} - u_{\text{ref}}\|_R^2 \right) \\ \text{s.t.} \quad & x_{k+i+1|k} = Ax_{k+i|k} + Bu_{k+i|k}, \\ & u_{\min} \le u_{k+i|k} \le u_{\max}, \\ & x_{k+i|k} \ge d_{\text{safe}}, \\ & x_{k|k} = \hat{x}_k. \end{aligned} \tag{20}$$

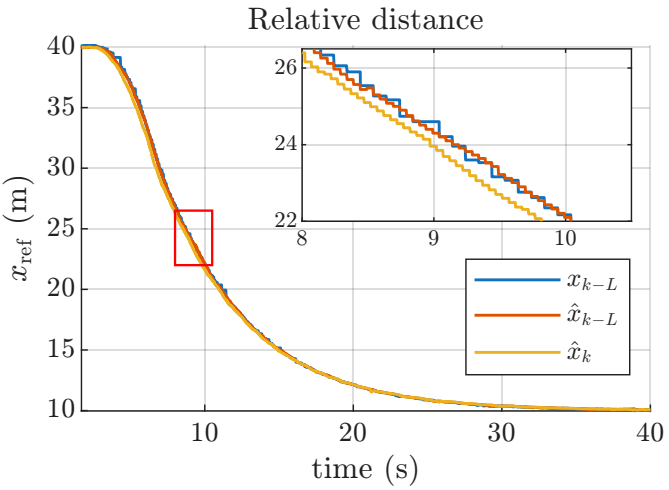
Here, $\hat{x}_k$ is the estimated state provided by the multi-rate Kalman observer, d_{safe} enforces a minimum inter-vehicle distance, and $u_{\text{ref}} = 0$ represents nominal cruising acceleration. The weighting matrices Q and R penalize deviation from the reference and excessive control effort, respectively. The optimization problem is formulated as a Quadratic Program (QP) and solved online at a rate of $50\,\text{Hz}$ using MATLAB's built-in MPC toolbox [14].

VI. Experimental results

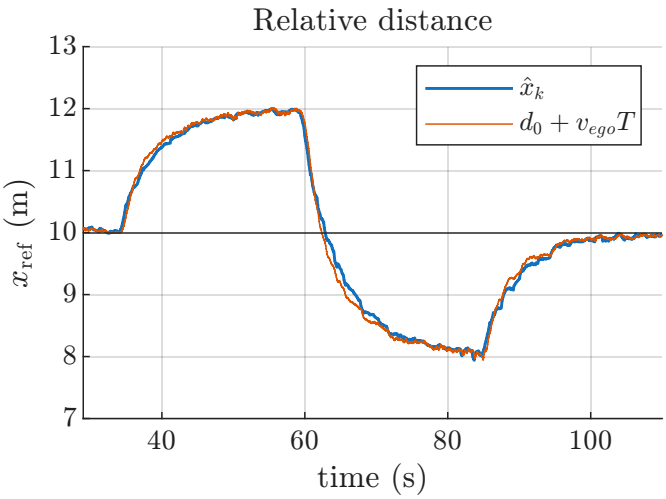
To validate the performance of the sensor system, the system is implemented on an experimental vehicle [12] and connected as described in section III. The implementation parameters of the experiment are shown in Table I. Note here that T_h has been selected conservatively to guaranty experimental safety.

A. ACC performance static lead car

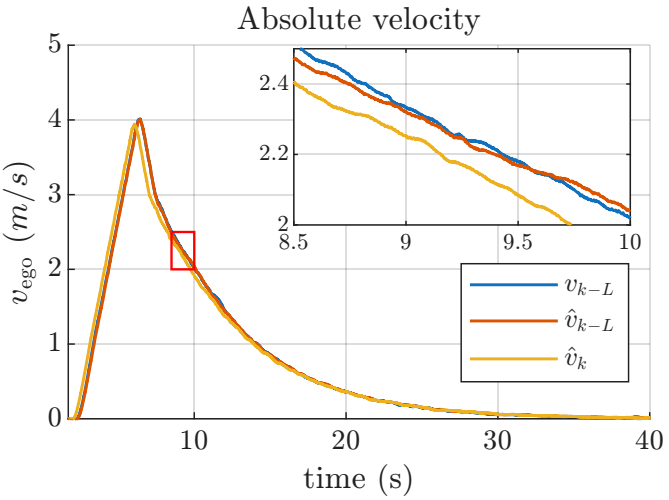
Second, the full sensor system is implemented including the MPC based vehicle controller. To test the performance the static lead car is placed at a 50 m distance. After initializing the sensor system the MPC controller is activated and because the follow distance is 10 m the ego car will drive towards the lead car and stop 50 m in front of the lead car. In order to show the performance improvement the results using no KF, using Kf and using KF and delay compensation are compared



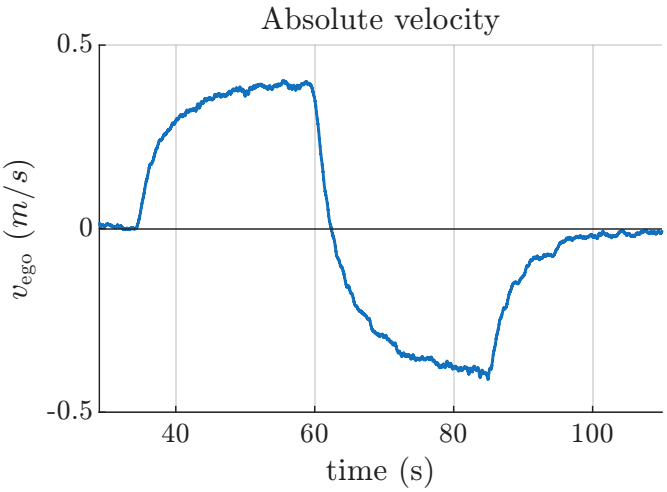
((a)) Relative distance between static lead car and ego car, where x_{k-L} represents no KF, $\hat{x}_{k-L}$ represents KF without delay compensation and $\hat{x}_k$ represents KF with delay compensation



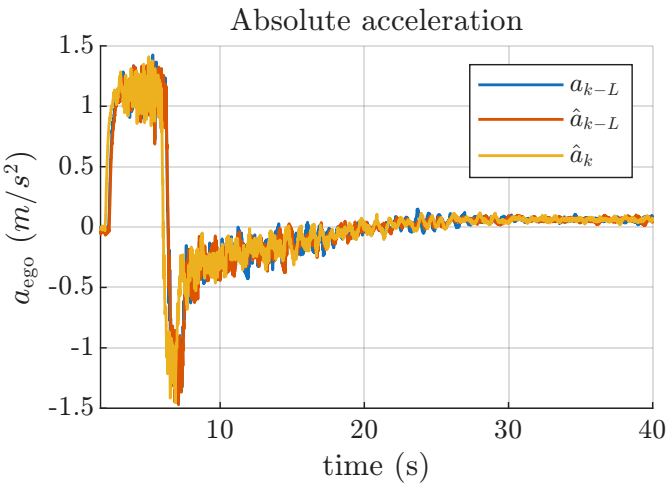
((b)) Relative distance between dynamic lead car and ego car compared to reference, using KF and delay compensation compared with reference



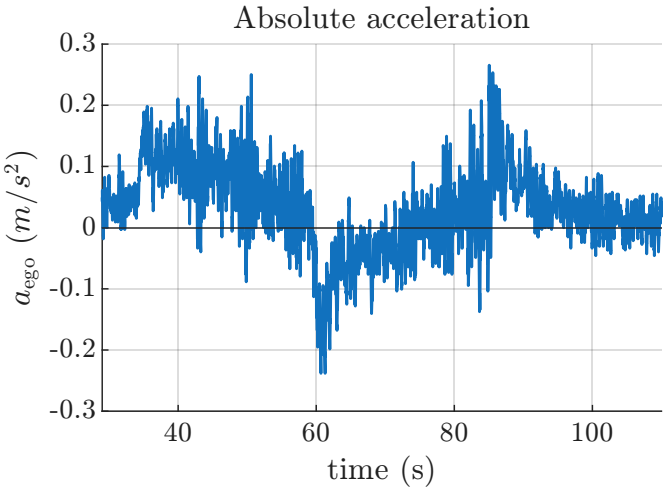
((c)) Absolute velocity between static lead car and ego car



((d)) Absolute velocity of ego car following the dynamic lead car



((e)) Absolute acceleration between static lead car and ego car



((f)) Absolute acceleration of ego car following the dynamic lead car

Fig. 7: Experimental results for ACC using a static lead car (left column) and using a dynamic lead car (right column), static scenario shows the improved control rate and effective delay compensation. The dynamic scenario shows effective trajectory tracking.

in Fig. 7(a), Fig. 7(b) and Fig. 7(c). The result show that the KF increases the effective estimation rate to 20 Hz, thereby increasing also the overall control frequency. Additionally, the delay compensation reduces the average deviation between the estimated and true position.

B. ACC performance with dynamic lead car

Finally, the ACC is tested with a dynamic lead vehicle. First, the relative distance between the ego vehicle and lead vehicle is brought to the safety distance of 10 m using the procedure described in subsection VI-A. If both cars are static, the lead car will move 10 m forward at 0.4 m/s, after which it will return again 10 m backwards at the same speed. Thanks to the MPC the ego car will follow this trajectory at a reference distance, which is dependent on the ego car speed. The result is shown in Fig. 7(d), Fig. 7(e) and Fig. 7(f). The ego car tracks the reference position with a RMSE of 0.3 m during constant velocity and tracks with a RMSE of 0.7 m during the full trajectory. After the lead car is stationary again the following distance converges to the safety distance.

VII. Conclusion

This paper presented a practical multirate sensing and control framework for adaptive cruise control in IWM-EVs under sparse and delayed LiDAR measurements. The proposed system combines lightweight point-cloud processing, delay-compensated state estimation, and MPC-based longitudinal control in a real-vehicle implementation. Experimental results demonstrate that the framework increases the effective estimation update rate from the original 10 Hz LiDAR measurements to 20 Hz, thereby reducing the effective sensing interval and improving the smoothness and consistency of the estimated distance signals, while remaining within the real-time computational constraints of the vehicle controller. Although the estimation rate remains below the bandwidth of the motor controller, the achieved 20 Hz update rate proved sufficient for stable and smooth low-speed ACC maneuvers in the considered experimental scenarios. In both static and dynamic ACC experiments, the proposed framework achieved stable and smooth vehicle-following behavior. During the dynamic tracking experiment, the ego vehicle achieved a tracking RMSE of 0.3 m during constant-velocity motion. These results demonstrate that the proposed system can effectively compensate for sparse and delayed LiDAR measurements in practical IWM-EV ACC scenarios. The main contribution of this study lies in the practical system-level integration and experimental validation of sensing, estimation, delay-compensation, and control methods for LiDAR-based ACC in IWM-EVs. Furthermore, the proposed two-loop architecture enables straightforward integration with existing driving-force-control strategies without modification of the low-level wheel control structure. Future work includes improving the latency identification and compensation framework, reducing computational complexity through lower-complexity control strategies, and accelerating communication between the perception and vehicle controller. In addition, tracking performance could be further improved through sensor fusion with IMU or radar measurements to improve acceleration estimation accuracy.

TABLE I: Implementation parameters for detection and MPC

Detection	**Description**	**Value**
ROI_x	Region Of Interest x direction	2 m : 60 m
ROI_y	Region Of Interest y direction	-0.75 m : 0.75 m
ROI_z	Region Of Interest z direction	-2 m : 1 m
V_{size}	Voxel size	0.4 m
ϵ	Neighborhood radius	0.9 m
p_{min}	Minimum points of object	8
τ_{coast}	Maximum coast time	0.5 s
τ_{max}	Maximum estimate time	1 s

MPC	**Description**	**Value**
T_s	MPC sample time	0.05 s
N_p	Prediction horizon	30
d_0	Standstill distance	10.0 m
T_h	Time gap	5 s
$u_{\min}, u_{\max}$	Acceleration limits	-2 m/s^2 : 2 m/s^2

References

[1] H. Fujimoto, J. Amada, and K. Maeda, "Review of traction and braking control for electric vehicle," in *2012 IEEE Vehicle Power and Propulsion Conference*, 2012, pp. 1292–1299.

[2] N. Ando and H. Fujimoto, "Yaw-rate control for electric vehicle with active front/rear steering and driving/braking force distribution of rear wheels," in *2010 11th IEEE International Workshop on Advanced Motion Control (AMC)*, 2010, pp. 726–731.

[3] H. Fujimoto and S. Harada, "Model-based range extension control system for electric vehicles with front and rear driving–braking force distributions," *IEEE Transactions on Industrial Electronics*, vol. 62, no. 5, pp. 3245–3254, 2015.

[4] H. Kam, S.-H. Lee, T. Park, and C.-H. Kim, "Rviz: a toolkit for real domain data visualization," *Telecommunication Systems*, vol. 60, pp. 1–9, 10 2015.

[5] B. M. Nguyen, W. Ohnishi, Y. Wang, H. Fujimoto, Y. Hori, K. Ito, M. Odai, H. Ogawa, E. Takano, T. Inoue, and M. Koyama, "Dual rate kalman filter considering delayed measurement and its application in visual servo," in *2014 IEEE 13th International Workshop on Advanced Motion Control (AMC)*, 2014, pp. 494–499.

[6] Livox Technology Company, *Livox HAP LiDAR User Manual*, 2023. [Online]. Available: https://www.livoxtech.com/hap

[7] NVIDIA Corporation, *NVIDIA Jetson Nano Developer Kit*, 2022. [Online]. Available: https://developer.nvidia.com/embedded/jetson-nano-developer-kit

[8] dSPACE GmbH, *dSPACE MicroLabBox*, 2023. [Online]. Available: https://www.dspace.com/en/pub/home/products/hw/microlabbox.cfm

[9] S. Macenski, T. Foote, B. Gerkey, C. Lalancette, and W. Woodall, "Robot operating system 2: Design, architecture, and uses in the wild," *Science Robotics*, vol. 7, no. 66, p. eabm6074, 2022. [Online]. Available: https://www.science.org/doi/abs/10.1126/scirobotics.abm6074

[10] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1–4, 2011.

[11] Y. Ling, Y. Wang, and T. Chan, "Ransac-based planar point cloud segmentation enhanced by normal vector and maximum principal curvature clustering," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. X-1-2024, pp. 145–151, 05 2024.

[12] Y. Hosomi, B.-M. Nguyen, H. Fujimoto, H. Ikeda, and T. Nohara, "Driving force control for on-board motor electric vehicles with adaptive drivetrain friction and phase stabilization speed controller," in *2024 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, 2024, pp. 1456–1461.

[13] T. Ueno, B.-M. Nguyen, S. Nagai, and H. Fujimoto, "Wheel velocity based cascade driving force control for electric vehicles," *IEEE/ASME Transactions on Mechatronics*, vol. 30, no. 4, pp. 2620–2631, 2025.

[14] MathWorks, *Model Predictive Control Toolbox User's Guide*, The MathWorks, Inc., Natick, Massachusetts, USA, 2024, version R2023b, https://www.mathworks.com/products/mpc.html.