Vehicle trajectory filtering based on optimal control and simulator-based reconstruction

Vjekoslav Diklić*, Ivan Marković †,
* dSPACE engineering d. o. o. Zagreb, Croatia vdiklic@dspace.hr
† Fakultet elektrotehnike i računarstva, Zagreb, Croatia ivan.markovic@fer.hr

***Abstract*—In the automotive industry, simulations driven by real-world data are desirable for testing and development, enabling realistic and accurate performance evaluation and system optimization. However, real-world data is often plagued by disturbances due to factors such as sensor noise, outliers, varying driving conditions, recording delays, and inconsistencies that affect data reliability and limit simulation effectiveness. A common practice for mitigating these disturbances is to employ low-pass filters, Kalman filters, smoothers, or a combination of these methods to improve the quality of recorded data. While these techniques can yield accurate results, they may inhibit the use of more complex vehicle physical models or oversmooth critical moments from the traffic scenario. In this paper, we propose to rethink vehicle trajectory filtering and treat it as a control problem where the vehicle needs to traverse a set of reference points based on a physically realistic vehicle model. The final output of this approach is noise- and outlier-free, resampled vehicle trajectory, and states such as velocity, heading angle, and vehicle steering wheel position. The proposed method is evaluated in simulation scenarios with data plagued by various disturbances. Common solution for such data is the Rauch-Tung-Striebel smoother which was utilized for comparison.**

***Keywords*—vehicle trajectory, Kalman filter, Rauch-Tung-Striebel smoother, state estimation**

I. Introduction

In modern vehicle development, testing is primarily conducted through simulations due to the speed, repeatability, and reduced costs of simulation-based tests. The value of these simulations is significantly enhanced when based on real-world driving data, as it increases diversity and provides unique scenarios. However, the generation of such scenarios is highly susceptible to variations in vehicle trajectory noise and other external disturbances. Existing approaches to filtering real-world driving data often oversmooth critical moments, reducing simulation quality to the extent that the scenario becomes unusable for the intended testing purpose.

Post-processing of vehicle trajectory data has been extensively researched, particularly focusing on methods to reduce noise and improve accuracy. The *Kalman filter* [1] is widely recognized for its ability to optimally estimate vehicle states from noisy data [2]–[4]. Another important method is the *Rauch-Tung-Striebel (RTS) smoother* [5], which enhances Kalman filtering by considering both past and future data points, ensuring smoother reconstructions [6]. Additionally, *Lagrange interpolation* is commonly used alongside Kalman filtering to address trajectory outliers and maintain data fidelity during post-processing [2]. This hybrid approach has demonstrated significant improvements in reducing root mean square error and increasing the signal-to-noise ratio. It is common practice to employ sensor fusion techniques on recorded real-world driving data, either during the recording phase, in post-processing, or both, to combine complementary data from multiple sensors and improve the accuracy of vehicle state estimation [7]–[9]. Although post-processing of vehicle trajectory data using methods such as the Kalman filter [4], [10], low-pass filters, and particle filtering [11] can yield accurate results, these techniques can sometimes produce inadequate outcomes for generating precise driving scenarios, such as oversmoothing critical actions. This can lead to errors in recreated scenarios, such as lateral offsets or inadequate braking, potentially causing vehicle collisions that did not occur during the real-life data capture.

Vehicle trajectories recorded from real-world driving scenarios often contain various disturbances. Based on our analysis and prior studies of available datasets, such as [12], [13], these disturbances can generally be categorized into three types: general white or colored noise, random outliers, and uneven sampling or missing data points. Such disturbances may arise from unsynchronized or unequal sensor sampling rates, environmental factors, road infrastructure effects, weather conditions, or other sources. Additionally, resulting vehicle trajectory data often include only the vehicle position over time, leaving critical parameters such as velocity, heading angle, and steering wheel position to be estimated during post-processing. To create accurate and credible driving scenarios, all vehicle states, depending on the required level of detail, need to be available besides the trajectory data. The commonly used vehicle states for driving scenario reconstruction are position, velocity, steering angle, and heading angle. These variables are components of a simplified vehicle model, commonly referred to as the bicycle model [14]. Given that, the problem at hand is to reconstruct the vehicle trajectory and estimate the aforementioned vehicle states at an even sample rate, from noisy and incomplete trajectory data. The goal is to ensure that the resulting trajectory aligns with the input trajectory, accurately reconstructs vehicle states, and produces a trajectory that, while not identical to the original, is similar and free from noise, distortions, and other artifacts.

In this paper, a novel method for vehicle trajectory filtering is proposed. We focus exclusively on post-processing

of trajectories and do not cover filtering methods that are used during vehicle data recording. The proposed approach is based on simulating vehicle driving using a modified Model Predictive Controller (MPC) [15] that follows the input trajectory as a reference path. The performance of the novel method is compared to the existing Rauch-Tung-Striebel (RTS) smoother. Data for testing in this work is based on synthetic data altered by applying various types of disturbances. The performance of the proposed method and RTS is evaluated against the original ground-truth trajectory based on metrics such as the mean square error and standard deviation.

The paper is organized as follows: The proposed filtering method is presented in Sec. II, followed by Sec. III-B, which covers data acquisition, disturbance simulation, and processing. Evaluation and results are detailed in Sec. IV, and the conclusion is provided in Sec. V.

II. Proposed Postprocessing Method

This chapter presents a novel solution based on model simulation. Simulating any system model provides distortion-free data with a sampling time greater than or equal to the integration step size. The novel method introduced in this work employs a modified MPC that uses data requiring filtering as a list of desired states through which the model is driven in simulation. The ability of the MPC to operate in this form is achieved by adding an additional member to its criterion function. The standard criterion function J_o is defined as:

$$J_o = \sum_{k=0}^{N-1} \left[e_k^T Q e_k + \Delta u_k^T R \Delta u_k \right] + e_N^T S_N e_N. \quad (1)$$

In this equation, e_k represents the state error at step k, Q is the weighting matrix for state errors, Δu_k represents the control input deviation, R is the weighting matrix for control inputs, e_N is the terminal state error, and S_N is the weighting matrix for terminal state errors. The criterion function is modified by adding an additional term:

$$J_N = J_o + \sum_{j=0}^{M} [(x_{N+j} - X'_j)^T \frac{\mathbf{FN}}{f_z(j)} (x_{N+j} - X'_j)]. \quad (2)$$

In the standard application of MPC there is a current model state and a goal state. Solving the criterion function for a given horizon size yields a set of control values that will drive the system to the desired goal state. Such an application of MPC for filtering, when a list of states is used, may encounter issues if the criterion function does not include terms for states that follow the goal state. The additional term in the criterion function enables the inclusion of states beyond the goal state, effectively anticipating the next goal state. The extended part is providieng just that ability, to take the so-called *future states* into calculation. The inspiration for this modification of MPC came from navigating a vehicle through a cone obstacle course. While the driver steers around the current cone, they simultaneously anticipate and prepare for the next obstacle. In other words, their focus on the immediate next cone is dominant, but they are also anticipating the cone that follows. **FN** is a weight matrix for the future states, while $f_z(j)$ is a decay function that reduces the weight of states as they move further from the current point. $f_z(j)$ can be defined as a simple exponential function, $f_z(j) = z^j$, or any other function that decreases weight based on the distance from the current goal state. The elements $x_{N+j} - X'_j$, similar to e_N, represent the error between the predicted final points and the future states.

A. Patching and Reconstruction of Vehicle States

The previously described method requires vehicle states as input data. Since the input data consists only of vehicle positions over time, the vehicle states were reconstructed using simple formulas. This process involves two main steps: patching missing samples and reconstructing the vehicle states using basic vehicle kinematics. The process of patching missing samples consists of two main steps: detecting the missing samples and applying patches by averaging the previous and subsequent samples in the data. The first step involves analyzing the time intervals between consecutive data points. A reference time interval is determined by identifying the smallest consistent time difference across the filter input data. This reference, denoted as Δt_{ref}, serves as the basis for detecting missing data points:

$$\Delta t_{\text{ref}} = \min(T_{i+1} - T_i). \quad (3)$$

When the time gap between two adjacent points exceeds the reference value, it indicates missing samples requiring patching. The number of missing samples is estimated as:

$$\text{Number of missing samples} = \left\lfloor \frac{\Delta t}{\Delta t_{\text{ref}}} \right\rfloor - 1, \quad (4)$$

where $\Delta t = T_{i+1} - T_i$.

The next step involves reconstructing the data at missing points using linear interpolation. For a missing sample at index k, the interpolated values for the X and Y coordinates are calculated as:

$$X_k = X_i + \frac{k-i}{n}(X_{i+1} - X_i), \quad (5)$$

$$Y_k = Y_i + \frac{k-i}{n}(Y_{i+1} - Y_i), \quad (6)$$

where n is the total number of missing samples plus one, and k runs from 1 to n. The time value for the interpolated points is:

$$T_k = T_i + k \cdot \Delta t_{\text{ref}}. \quad (7)$$

This linear interpolation technique generates new values for missing samples, ensuring a smooth and continuous transition across the gap. Patching these samples is essential because the subsequent stage of vehicle state reconstruction involves data differentiation, which is highly sensitive to missing data. After patching the missing samples, the next stage reconstructs the vehicle's state using basic vehicle kinematics. This process begins with calculating the vehicle's velocity from the patched spatial and temporal data. Velocity is calculated as the rate of

change of the vehicle's position over time, which requires accurate and consistent time intervals between data points. This is expressed as:

$$v(t) = \frac{\sqrt{(x(t) - x(t-\Delta t))^2 + (y(t) - y(t-\Delta t))^2}}{\Delta t}. \tag{8}$$

Here $x(t)$ and $y(t)$ denote the spatial coordinates at time t, and Δt represents the time interval between consecutive samples. The vehicle's heading angle is computed from the directional changes observed in its trajectory. This angle indicates the vehicle's orientation relative to its previous position:

$$\theta(t) = \text{atan2}(y(t) - y(t-\Delta t), x(t) - x(t-\Delta t)), \tag{9}$$

while the steering angle of the vehicle is calculated as:

$$\delta(t) = \text{atan2}\left(\frac{L \cdot \left(\frac{\theta(t) - \theta(t-\Delta t)}{\Delta t}\right)}{v(t)}\right) \tag{10}$$

Here $\delta(t)$ denotes the steering angle at time t, L is the wheelbase (the distance between the front and rear axles), and $\dot{\theta}(t)$ represents the yaw rate (rate of change of the heading angle). Additionally, the change in heading angle, often referred to as the angular rate or yaw rate, is calculated to provide further insights into the vehicle's steering and turning behavior. To ensure smoothness and accuracy in estimating these vehicle state variables, a low-pass filter is applied to the data. This filter reduces high-frequency noise that could distort velocity and heading angle calculations, enabling a more reliable reconstruction of the vehicle's dynamic states. The result of this stage is a comprehensive vehicle state profile, including the vehicle's position, velocity, heading angle, and changes in heading angle over time. This profile serves as a critical input for subsequent modeling and analysis of vehicle dynamics.

B. Vehicle Model

Both the RTS smoother and the described MPC-based filtering method require a system model for their operation. For this purpose, the vehicle bicycle model [14] was chosen. Although simplified, this model provides reliable future state estimations while maintaining computational efficiency. The bicycle model represents the vehicle's state-space with five components $\mathbf{x}$ and the input control vector with two components $\mathbf{u}$:

$$\mathbf{x} = \begin{bmatrix} p_x & p_y & v & \psi & \delta \end{bmatrix}^\top = \begin{bmatrix} x_1 & x_2 & x_3 & x_4 & x_5 \end{bmatrix}^\top \tag{11}$$

$$\mathbf{u} = \begin{bmatrix} \dot{\delta} & a \end{bmatrix}^\top = \begin{bmatrix} u_1 & u_2 \end{bmatrix}^\top. \tag{12}$$

In this model, p_x and p_y represent the global vehicle position, v is the longitudinal velocity, ψ denotes the heading angle, and δ represents the steering angle. The control inputs consist of the acceleration a, which can be positive for acceleration or negative for braking, and the steering wheel rate $\dot{\delta}$.

TABLE I: DLC and SLC parameters

Parameter	*shape*	d_{x_1}	d_{x_2}	d_{y_1}	d_{y_2}	X_{s_1}	X_{s_2}
DLC	2.4	25	21.95	4.05	5.7	27.19	56.46
SLC	2	25	21.95	5	5.7	60	1000

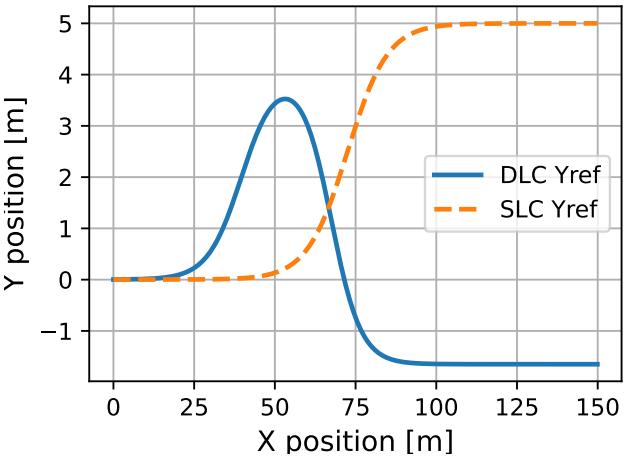


Fig. 1: Reference path

III. Data Generation, Disturbances, and Processing

A. Data Generation and Processing

The data used as ground truth was acquired by simulating a multi-body vehicle model from the Python-based CommonRoad Vehicle Library [16]. The simulation included a double lane change (DLC) maneuver, as proposed in [17], and a similar single lane change (SLC) maneuver with parameters shown in Table I, while trajectories are illustrated in Fig. 1 with the vehicle velocity set to 5 m/s. The acquired data consisted of 29 model state variables sampled at a rate of 0.025 s. Path following was achieved using an MPC with a nonlinear bicycle model, which used only five state variables for prediction. After data generation, five state variables were selected from the model's original 29: x and y positions, vehicle velocity, heading angle, and steering wheel angle. Hereafter, this reduced dataset is referred to as the Ground Truth (GT).

B. Adding Disturbances

This subsection describes how commonly encountered disturbance types were mimicked and introduced in the dataset. From the GT, the x and y positions, along with a time vector, were selected for adding the distortions. Three types of distortions were introduced to the data: noise, random sample removal, and random outlier injection. Noise was generated by applying a specific noise-to-signal ratio (NSR) to the GT data, ensuring controlled degradation of signal quality. Additionally, the noise could vary in type, either white or colored.The process of adding noise to the signal begins with calculating the signal power. The average power of the signal, P_{signal}, is computed using the formula:

$$P_{\text{signal}} = \frac{1}{n}\sum_{i=1}^{n}(x_i)^2, \tag{13}$$

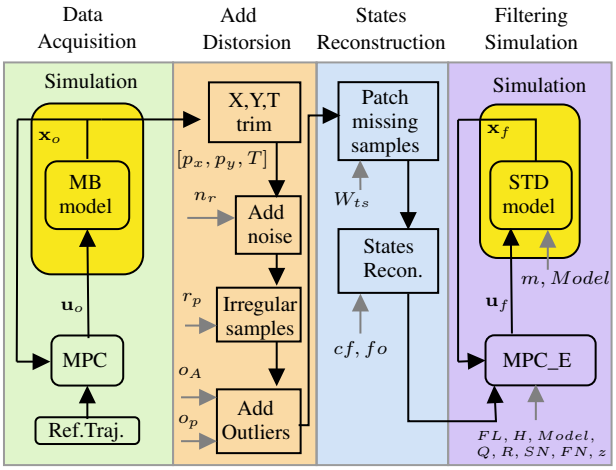


Fig. 2: Evaluation configuration overview diagram

where x_i represents the samples of the signal, and n is the total number of samples. Using the noise-to-signal ratio NSR, the noise power P_{noise} is calculated as:

$$P_{\text{noise}} = P_{\text{signal}} \times 10^{\left(\frac{\text{NSR}_{\text{dB}}}{10}\right)}. \tag{14}$$

White noise is then generated as a normal distribution with zero mean and standard deviation $\sqrt{P_{\text{noise}}}$.Finally, the generated noise was added to the original signal to create the noisy version.

Outliers were introduced to simulate extreme deviations in the data, which are commonly encountered in real-world signals due to sensor errors or unexpected events. The outliers added to the GT data were generated by adding values several times larger than the average signal value at randomly selected samples. The number of outliers, N_o, was determined by multiplying a specified percentage O_p with the total number of samples N_s and rounding the result to the nearest whole number. The amplification of the added values is controlled by a factor O_A, which scales the average signal value. Each outlier was introduced by adding a multiple of the average value of the corresponding data entry. The sign of the outliers was randomized to simulate both positive and negative deviations. Additionally, the values of the outliers varied to prevent uniform patterns, ensuring the deviations mimicked realistic scenarios.

Irregular sampling was simulated to mimic data gaps or uneven sampling rates, which commonly occur due to communication failures, low sensor resolution, or incomplete data acquisition. This was achieved by removing a specified percentage r_p of samples from each signal. The number of samples to be removed, N_r, was calculated by multiplying the specified percentage r_p by the total sample count N_s and rounding to the nearest whole number. Random indices were then selected to identify which samples to remove, ensuring that no sample was selected more than once. This method introduced gaps in the data, effectively replicating missing samples or irregularly spaced data points observed in real-world scenarios.

IV. Evaluation

The evaluation was conducted in two steps. In the first step, the goal was to identify the optimal configuration for the given parameters: FL, Z, and horizon size. The second step examined how variations in vehicle mass in the simulation influenced the results. Additionally, the same trajectories were filtered with RTS for performance comparison.

Filtering simulations were performed using the Single Track Drift (STD) model [16], a modified single-track model with an introduced mass. Tire forces were calculated using the Pacejka tire model [18]. The STD model was used for filtering purposes because it offers the best compromise between computational efficiency and simulation accuracy compared more complex vehicle models. A general overview of the evaluation configuration is shown in the diagram in Fig. 2. The configuration and simulation were implemented in Python. The *CasADi* symbolic framework for Python [19] was used to define the optimization problems, while the Interior Point Optimizer *Ipopt* [20] was employed to solve them. Identical configuration for vehicle simulation was employed in out earlier work [21], where the focus was on path following and the implementation of MPC, while in this paper we expand the existing MPC to allow it to function as a filter and apply computed control variables on a more complex vehicle model to make the vehicle "drive" through the data points and effectively filter the data. Moreover, therein reference path was clearly defined, while in this case we are reconstructing trajectories from noisy data with missing samples.

With the stated goal of finding the best parameters, multiple experimental runs were conducted using both SLC and DLC trajectories, each with varying levels of applied distortion. The values of parameter FL were varied from 0 to 25, the Z parameter was tested at values of 0.9, 0.97, 1, 1.05, 1.1, 1.2, 1.5, and 2, while the prediction horizon was evaluated for three sizes: 6, 10, and 15. As a control, a distortion-free trajectory was also filtered to determine the extent of distortion introduced by the filtering method itself. Vehicle mass was set to be the same as MB model in reference simulation $m = 1000\ kg$. Distortion parameters and results are presented in Tables III and II. This table contains results for the three best parameter sets for each configuration, along with RTS. The mean square error between the reference vehicle trajectory and the filtering results is denoted by e_{pos}, while the standard deviation of the same values is represented as δ_{pos}. A visual representation of the distorted trajectories used as the filter input can be seen in Figs. 3a and 3b. Filtering performance in terms of MSE position, in relation to parameters FL, Z, and the horizon, can be seen in Fig. 4. Filtering results, together with the RTS results, are shown in Figs. 5 and 6. The three best overall configurations for the combined SLC and DLC cases were achieved with $FL = 1$ and a horizon size of 15. The main difference was in the Z values, with the best result at $Z = 1.2$, the

TABLE II: Configuration parameters and results: SLC

ID		1	2	3	4	5
Parameter	n_r	0.1	0.1	0.2	0.5	0
	r_p	5%	5%	20%	20%	0%
	O_A	2%	10%	10%	20%	0
	O_p	2	5	10	10	0
Best	FL, z, h	8, 0.9, 15	11, 1.2, 15	2, 0.9, 15	4, 0.95, 6	18, 1.1, 6
	e_{pos}	**0.00503**	**0.03340**	**0.17960**	**0.28142**	0.00046
	σ_{pos}	**0.00106**	**0.00621**	**0.00629**	**0.01487**	**0.00001**
RTS	e_{pos}	0.00677	0.13212	1.30350	1.42003	**0.00003**
	σ_{pos}	0.01077	0.32903	2.30069	2.09224	0.00006

TABLE III: Configuration parameters and results: DLC

ID		1	2	3	4	5
Parameter	n_r	0.1	0.1	0.2	0.5	0
	r_p	5%	5%	20%	20%	0%
	O_A	2%	10%	10%	20%	0
	O_p	2	5	10	10	0
Best	FL, z, h	8, 1.1, 10	11, 1.05, 6	5, 1.1, 6	6, 1, 6	13, 1.2, 15
	e_{pos}	0.02546	0.01110	0.02104	0.02438	0.02579
	σ_{pos}	**0.0027**	0.00226	**0.00451**	**0.00551**	0.00155
RTS	e_{pos}	**0.00117**	**0.00194**	**0.00498**	**0.00751**	**0.00018**
	σ_{pos}	0.00155	**0.00212**	0.00684	0.00915	**0.00025**

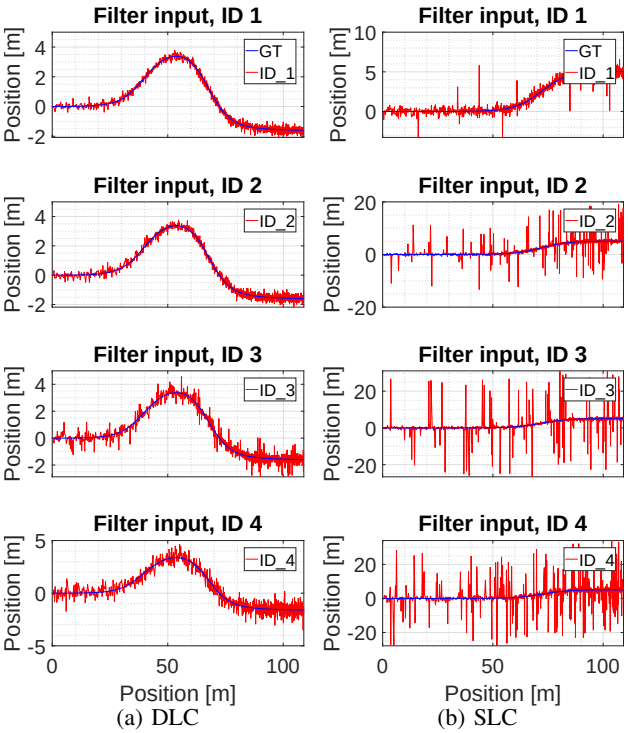


Fig. 3: Input Trajectories

second-best at $Z = 1.5$, and the third-best at $Z = 0.95$.

The three configurations that demonstrated the best overall performance in the first part were used in the second part of the evaluation, with variable vehicle masses in the filtering simulation. For each filtering simulation, the vehicle mass was varied with values of 500, 1000, 2000, 3000, 4000, 5000, and 6000 kg. Distortion parameters were set to configuration ID one from Tables II and III.

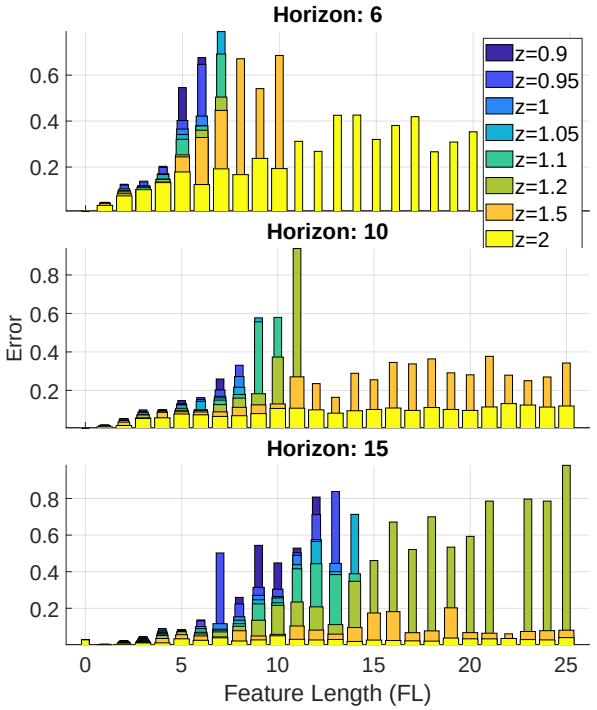


Fig. 4: Pos. MSE for different FL, Z, and horizon values.

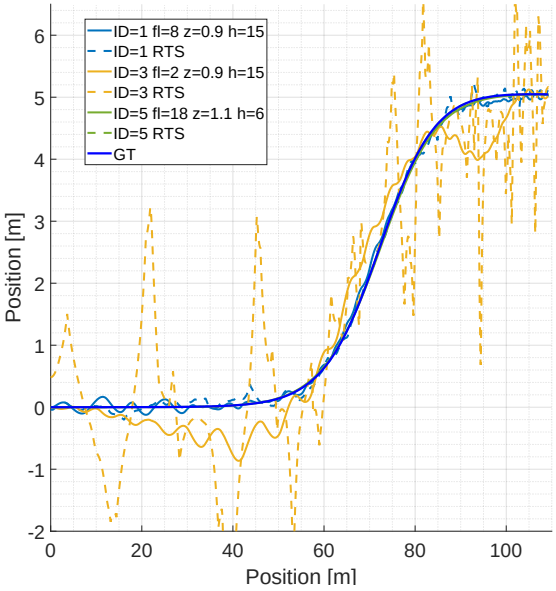


Fig. 5: SLC results.

TABLE IV: Comparison of mass influence on MSE and STD position metrics.

mass	z=0.95		z=1.2		z=1.5	
$[kg]$	e_{pos}	σ_{pos}	e_{pos}	σ_{pos}	e_{pos}	σ_{pos}
500	0.03465	0.06094	**0.03241**	0.05837	**0.03205**	0.05621
1000	**0.034**	**0.05974**	0.03338	**0.05801**	0.03283	**0.05581**
1500	0.03826	0.06757	0.03952	0.06768	0.04059	0.06809
2000	0.03869	0.06983	0.03991	0.07073	0.04153	0.072
3000	0.03853	0.07177	0.03998	0.07387	0.0412	0.07541
4000	0.03737	0.06999	0.03853	0.07176	0.03986	0.073
5000	0.03625	0.06875	0.03707	0.07	0.0379	0.07111
6000	0.03526	0.06806	0.0358	0.06916	0.0365	0.07017

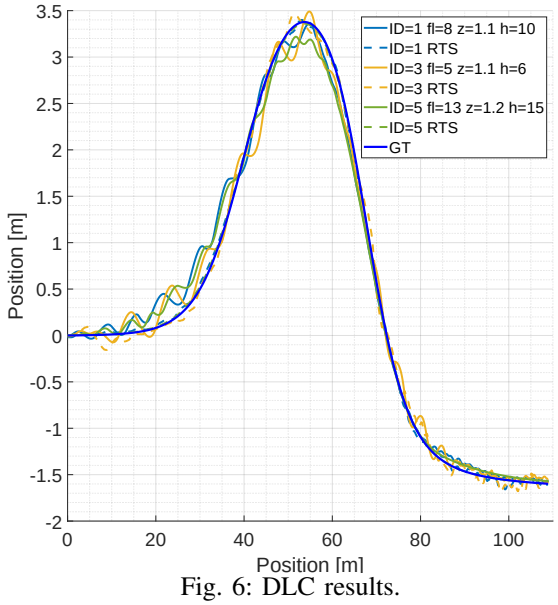


Fig. 6: DLC results.

Filtered trajectories are shown in Figs. 5 and 6, and the corresponding values of error and deviation can be seen in Table IV. The execution time of the proposed method depends on the FL variable, used simulator, and the horizon size, ranging from approximately 100 s to longer, whereas the RTS smoother processes the aforementioned data in a bit less than 0.3 s. However, note that the proposed method is intended to be used offline in a single run to preprocess the data.

V. Conclusion

In the paper we have presented a method for filtering vehicle trajectories based on controlling a simulated vehicle via MPC through the missing-sample and noisy dataset. Results obtained in the evaluation indicate that the introduced method has the capability to filter out a vehicle trajectory plagued with considerable levels of noise. In comparison to RTS, it showed more accurate performance in higher-noise regimes for the single-lane change case, while the double lane change case RTS had on average better results, although the proposed method exhibited smaller deviation in some cases. However, the idea of the proposed method is to be used offline, thus more advanced vehicle simulators can be used in lieu of the current STD model. Furthemore, several parameters and configurations for the introduced method were not tested within the scope of this paper. For example, road types, weather conditions, vehicle longitudinal behavior, and different types of road vehicles. This can be additionally tested in future work.

References

[1] R. E. Kalman, "A new approach to linear filtering and prediction problems," *Journal of Basic Engineering*, vol. 82, no. 1, pp. 35–45, 1960.

[2] J. Wang, Y. Liang, J. Tang, and Z. Wu, "Vehicle trajectory reconstruction using lagrange-interpolation-based framework," *Applied Sciences*, vol. 14, no. 3, 2024. [Online]. Available: https://www.mdpi.com/2076-3417/14/3/1173

[3] S. Sun, J. Bi, and C. Ding, "Cleaning and processing on the electric vehicle telematics data," in *Smart Service Systems, Operations Management, and Analytics*, H. Yang, R. Qiu, and W. Chen, Eds. Cham: Springer International Publishing, 2020, pp. 1–6.

[4] C. Barrios and Y. Motai, "Improving estimation of vehicle's trajectory using the latest global positioning system with kalman filtering," *IEEE Transactions on Instrumentation and Measurement*, vol. 60, no. 12, pp. 3747–3755, 2011.

[5] H. E. Rauch, F. Tung, and C. T. Striebel, "Maximum likelihood estimates of linear dynamic systems," *AIAA Journal*, vol. 3, no. 8, pp. 1445–1450, 1965.

[6] G. Su, Z. Zeng, A. Song, C. Zhao, F. Shen, L. Yuan, and X. Li, "A general framework for reconstructing full-sample continuous vehicle trajectories using roadside sensing data," *Applied Sciences*, vol. 13, no. 5, 2023. [Online]. Available: https://www.mdpi.com/2076-3417/13/5/3141

[7] B. Khaleghi, A. Khamis, F. O. Karray, and S. N. Razavi, "Multisensor data fusion: A review of the state-of-the-art," *Information Fusion*, vol. 14, no. 1, pp. 28–44, 2013. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S1566253511000558

[8] F. Castanedo, "A review of data fusion techniques," *The Scientific World Journal*, vol. 2013, no. 1, p. 704504, 2013. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1155/2013/704504

[9] S. Li and H.-S. Yoon, "Sensor fusion-based vehicle detection and tracking using a single camera and radar at a traffic intersection," *Sensors*, vol. 23, no. 10, 2023. [Online]. Available: https://www.mdpi.com/1424-8220/23/10/4888

[10] J. Sun, X. Sun, Z. Zhan, and J. Zhou, "A method of vehicle trajectory restoration based on kalman filter," in *2022 IEEE 8th International Conference on Cloud Computing and Intelligent Systems (CCIS)*, 2022, pp. 662–665.

[11] Y.-J. Liu, C.-H. Dou, F. Shen, and Q.-Y. Sun, "Vehicle state estimation based on unscented kalman filtering and a genetic-particle swarm algorithm," *Journal of The Institution of Engineers (India): Series C*, vol. 102, no. 2, p. 447–469, Jan. 2021. [Online]. Available: http://dx.doi.org/10.1007/s40032-021-00663-1

[12] R. Krajewski, J. Bock, L. Kloeker, and L. Eckstein, "The highd dataset: A drone dataset of naturalistic vehicle trajectories on german highways for validation of highly automated driving systems," in *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*, 2018, pp. 2118–2125.

[13] Y. He, B. Cao, and C.-Y. Chan, "Wut-ngsim: A high-precision and trustworthy vehicle trajectory dataset," 2022. [Online]. Available: https://dx.doi.org/10.21227/hmsb-ka76

[14] P. Polack, F. Altche, B. d'Andrea Novel, and A. de La Fortelle, "The kinematic bicycle model: A consistent model for planning feasible trajectories for autonomous vehicles?" in *2017 IEEE Intelligent Vehicles Symposium (IV)*. IEEE, Jun. 2017. [Online]. Available: http://dx.doi.org/10.1109/IVS.2017.7995816

[15] J. Richalet, A. Rault, J. Testud, and J. Papon, "Model predictive heuristic control: Applications to industrial processes," *Automatica*, vol. 14, no. 5, pp. 413–428, 1978.

[16] M. Althoff, M. Koschi, and S. Manzinger, "Commonroad: Composable benchmarks for motion planning on roads," in *Proc. of the IEEE Intelligent Vehicles Symposium*, 2017.

[17] F. Borrelli, P. Falcone, T. Keviczky, J. Asgari, and D. H. , "Mpc-based approach to active steering for autonomous vehicle systems," *International Journal of Vehicle Autonomous Systems*, vol. 3, no. 2-4, pp. 265–291, 2005.

[18] H. Pacejka and I. Besselink, *Tire and Vehicle Dynamics*, ser. Electronic publications. Elsevier Science, 2012. [Online]. Available: https://books.google.hr/books?id=ETnzam6qS2oC

[19] J. A. E. Andersson, J. Gillis, G. Horn, J. B. Rawlings, and M. Diehl, "CasADi – A software framework for nonlinear optimization and optimal control," *Mathematical Programming Computation*, vol. 11, no. 1, pp. 1–36, 2019.

[20] A. Wächter and L. T. Biegler, "On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming," *Mathematical Programming*, vol. 106, no. 1, p. 25–57, Apr. 2005. [Online]. Available: http://dx.doi.org/10.1007/s10107-004-0559-y

[21] V. Diklić and B. Novoselnik, "Comparison of linear and nonlinear model predictive control for vehicle path following," in *2024 47th MIPRO ICT and Electronics Convention (MIPRO)*, 2024, pp. 1794–1799.